User Interaction Context

Studying and Enhancing Automatic User Task Detection on the Computer Desktop via an Ontology-based User Interaction Context Model

Dissertation submitted to the
Graz University of Technology,
Faculty of Computer Science,
for the attainment of the degree of
Doctor of Engineering Sciences (Dr. techn.)

by

Dipl.-Ing. Dipl.-Ing. Andreas S. Rath
andreas.rath@tugraz.at
Knowledge Management Institute,
Graz University of Technology

First Assessor and Advisor: Univ.-Prof. Dr. Klaus Tochtermann
Second Assessor: Univ.-Prof. Dr. Ronald Maier
Advisor: Dr. Stefanie N. Lindstaedt

Graz, March 8, 2010

Abstract

User context detection has recently gained momentum, thanks to the interest shown by several established research areas, such as information retrieval, personal information management, technology enhanced learning as well as task and process management. In the information retrieval area user context is exploited for personalizing search, in personal information management for relating tasks, processes, persons, documents and projects, in task and process mining for discovering task and process flows as well as in the fields of technology enhanced learning and knowledge work support for generating rich user profiles in order to provide appropriate learning and work material. Automatic task detection is one important challenge in the area of user context detection because once the user task is known it is possible to support her better. Automatic task detection on the computer desktop is classically seen as a machine learning problem, but has only been explored with text-based and switching sequence based user context features. Furthermore no public standard datasets from laboratory or real-world experiments for investigating task detection are available.

The goal of this dissertation research is to automatically detect the task of a user on her computer desktop in order to enable task-specific support which is especially important for knowledge workers facing complex tasks and information overload in today's knowledge economy. More specifically, this research strives to study and enhance automatic task detection on the computer desktop via an ontology-based user interaction context model (UICO). The user interaction context is a subset of the user context and is defined as *"all interactions of the user with resources, applications and the operating system on the computer desktop"*. The proposed ontology including the automatically observed user interaction context data is utilized for the feature engineering for automatic task detection, more specifically automatic task classification.

This thesis introduces a novel task detection approach referred to as the *"ontology-based task detection approach"* and evaluates it on three independent datasets containing over 500 tasks from over 40 users of two different domains. These datasets are from three laboratory user experiments designed and performed as part of this research effort. The most important insights gained from these evaluations are: (i) combinations of features engineered from the UICO almost always outperform feature combinations suggested by existing task detection approaches, (ii) the J48 decision tree and Naïve Bayes classifiers perform globally better than the k-Nearest Neighbor and the Linear Support Vector Machine algorithms, (iii) six single features showing a good discriminative power for classifying tasks as well as a stable performance across the evaluation datasets (the *acc. obj. name* feature, the *window title* feature, the *used res. metadata* feature, the *acc. obj. value* feature, the *datatype properties* feature and the *acc. obj. role* feature), (iv) the best overall task detection results are achieved by the UICO feature category *Application Cat.* and by the combination of all 50 features (*All Categories*) and (v) knowledge-intensive tasks can be classified as well as routine tasks.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, March 8, 2010	Andreas S. Rath
....	....
Date	Andreas S. Rath

Contents

List of Figures

List of Tables

List of Algorithms

Preface and Acknowledgments

This PhD thesis represents the key results of my research performed at the Know-Center in the Knowledge Services division during the last four years. The Know-Center, Austria's Competence Center for knowledge-based Applications and Systems, and its associated Knowledge Management Institute, Graz University of Technology, represented an inspiring, idea generating and open-minded environment for me to pursue my dissertation research. Based on the work on technology-oriented knowledge management [Maurer & Tochtermann, 2002; Tochtermann, 2003] and how to increase the productivity of knowledge-workers through work-integrated learning [Lindstaedt *et al.*, 2008a], this dissertation investigates in mechanisms for automatically detecting the task of a user on the computer desktop through user observations. The knowledge discovery framework from Granitzer [2008], the *KnowMiner* framework, also provided a valuable base for indexing and retrieval of digital resources as well as applying information extraction techniques for enriching the resource dimension of the user context (see Section 3.5.2). In the Know-Center I had the great opportunity to be part of the national funded research project *DYONIPOS* [DYONIPOS, 2006] and the *KnowSe* project [KnowSe, 2009] which is one of the research projects regarding user context detection of the *Knowledge Services* division of the Know-Center. At this point I would like to thank Prof. Klaus Tochtermann and Dr. Stefanie N. Lindstaedt for establishing and providing this fruitful environment.

I would like to acknowledge and thank many people for their valuable support during my research work. Especially, I would like to thank Prof. Klaus Tochtermann, my Ph.D. adviser and professor, as well as Dr. Stefanie N. Lindstaedt, my advisor and division manager, for their willingness to support me in my discovery into the evolving areas of user context detection and task detection. They encouraged me to explore and to try out new ways as well as helped me to keep the big picture in mind. I benefited from their continuous constructive feedback and their valuable support during my research work and during the writing process of this thesis. I would also like to thank Prof. Ronald Maier for willing to serve as a second reader.

Without my wonderful collaborators of the Know-Center and the Knowledge Management

Institute I couldn't have made it this far: Dr. Tobias Ley and Barbara Kump for their feedback about the design of the task detection experiments. Dr. Stefanie N. Lindstaedt and Dr. Michael Granitzer for providing me with interesting perspectives and insights as well as for paving the way to the execution of the user experiments in terms of time, hardware, experiment environment and supportive students. Dr. Michael Granitzer, Mark Kröll, Christin Seifert, Peter Prettenhofer and Markus Muhr for teaching me the beauty of machine learning, their discussions and fruitful input about feature engineering, supervised machine learning and algorithm evaluation as well as for kindly answering my questions. Markus Leitner for investigating and testing several user context detection tools as part of his bachelor thesis [Leitner, 2007]. Georg Kompacher, Daniel Resanovic, and Didier Devaurs for their impressive work in the design and development of the KnowSe prototype as well as their support in helping me carrying out the experiments. Didier Devaurs for the interesting discussions and the new perspectives on user context detection as well as the great time during publishing the research results. Nicolas Weber, Mark Kröll and Dr. Michael Granitzer for their good collaboration in engineering the first Dyonipos prototype. Andreas Rechberger and Thomas Pichler for extending the list of available user interaction context sensors by developing one for Mozilla Thunderbird, Novell GroupWise and a Network Stream Sensor. Stefanie Wechtitch for her work on user interruptibility aspects for KnowSe. Nicolas Weber for his feedback on the chapter about the user interaction context ontology approach. Dr. Markus Strohmaier for reading the expose of my research proposal and his comments. Many thanks to the PhD. students of the "dissertation circle" of the Know-Center and the Knowledge Management Institute. In general, I would like to thank all the people from the Know-Center and the students from Graz University of Technology who participated in the experiments and supported me in carrying out my research work.

Parts of this dissertation research has also been published elsewhere. The conceptual model about the user interaction context, the Semantic Pyramid, is introduced in [Rath *et al.*, 2006]. The developed user interaction context sensors are described in [Pichler, 2007; Rath *et al.*, 2007; Rechberger, 2007]. The automatic detection and fulfillment of the user's information need based on the user context is elaborated in [Kröll *et al.*, 2006; Rath, 2007; Rath *et al.*, 2007]. The provision of context-aware knowledge services for personal information management is presented based on use cases in [Rath *et al.*, 2008]. [Granitzer *et al.*, 2009a, 2008; Kröll *et al.*, 2007] represent the joint work about automatic task detection during the Dyonipos project on which this dissertation research builds. The ontology-based user context model, the user interaction context ontology (UICO), is shown and described in [Rath *et al.*, 2009d]. In this paper also the ontology-based task detection approach is introduced. Furthermore this paper gives the first results of this novel task detection approach based on the experiment's dataset collected in the domain of the Know-Center (see Section 7.4.2). The detection of real user tasks by training on user interaction context data observed in a laboratory setting is explored in [Rath *et al.*, 2009a] (see Section 7.4.3). KnowSe services for fostering "awareness" in computer supported collaborative work environments are presented in [Rath *et al.*, 2009c]. In [Rath *et al.*, 2009b] an overview of the knowledge services for personalized learner support provided by the KnowSe framework is given.

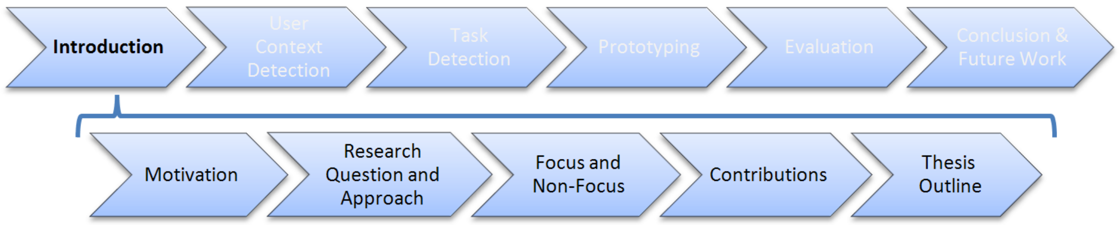
Finally, I would like to thank my family. They have provided tremendous support during my university years and helped me organizing my non-university life during intense research periods. Especially, I would like to say thank you to my generous, wise and beloved parents. This thesis is dedicated to them.

Graz, March 8, 2010

Andreas S. Rath

Andreas S. Rath

1 Introduction



Knowledge plays an increasingly important role in organization of all types in today's knowledge economy. The management of knowledge can decide upon the success or failure of an organization. This has been acknowledged by the interdisciplinary research field knowledge management that involves several research areas such as computer science, economics and psychology. In order to stay competitive within a global market an *"effective knowledge management has become imperative for organizations"* [Tochtermann, 2003]. The shift of the society towards a knowledge economy has also resulted into a change in the type of work that is in focus by organization executives to gain a competitive advantage. Today the challenge is to increase the productivity of knowledge work and knowledge workers [Drucker, 1999] in order to speed up the innovation cycles and the development of new products. Knowledge work can be characterized as *"creative work solving unstructured problems that require exploration or creation of knowledge"* [Maier, 2005] as opposed to manual work in which the task and how it should be performed is clear. This was different a few decades ago when the focus was put on increasing the productivity of manual workers in manufacturing [Drucker, 1999]. Since knowledge workers, people who perform knowledge work, are getting more important for organizations, organizations strive to support them as much as possible in order to increase their productivity.

Nowadays a lot of knowledge work is done with the help of computer systems which enable the access to information of various sources from inside and outside the organization, e.g., the Internet, the organizations intranet, organizational information systems, or databases. Since the amount of information available is continuously growing, these knowledge workers are more and more facing an *information overload*. A literature survey about information overload research from Edmunds & Morris [2000] recognizes the theme that although enough information is available it is difficult to obtain relevant, useful information when it is needed. Feldman [2004] reported about IDC[1] studies unveiling that (i) knowledge workers spend between 15% to 35% of their time searching for information and that (ii) 40% of corporate users reported that they can not find the information they need to do their work on their intranets. Possible solutions to information overload Edmunds & Morris [2000] identified are: a reduction in the duplication of information, the adaption of personal information management strategies together with intelligent software solutions and the provisioning of value-added information.

These findings among others motivated this dissertation research, which goal is to automatically identify the user task on a desktop computer system in order to enable a task-specific support. Task-specific support can be of various forms, such as providing useful, relevant material for work and learn situations, suggesting topical experts, finding collaborators as well as the management and the access of task-specific information.

Technology-oriented Knowledge Management

Supporting knowledge workers with the knowledge required to efficiently perform a specific task is the objective of technology-oriented knowledge management [Tochtermann, 2003]. The *Maurer-Tochtermann Model (MT-Model) for Knowledge Management* [Maurer & Tochtermann, 2002] visualized in Figure 1.1 describes knowledge management from an information technology perspective. Although the focus of the MT-Model is on the communication between people and computer systems (KM-system), they also included the organizational perspective. As can be observed in Figure 1.1, a large amount of knowledge exchange happens over a KM-system.

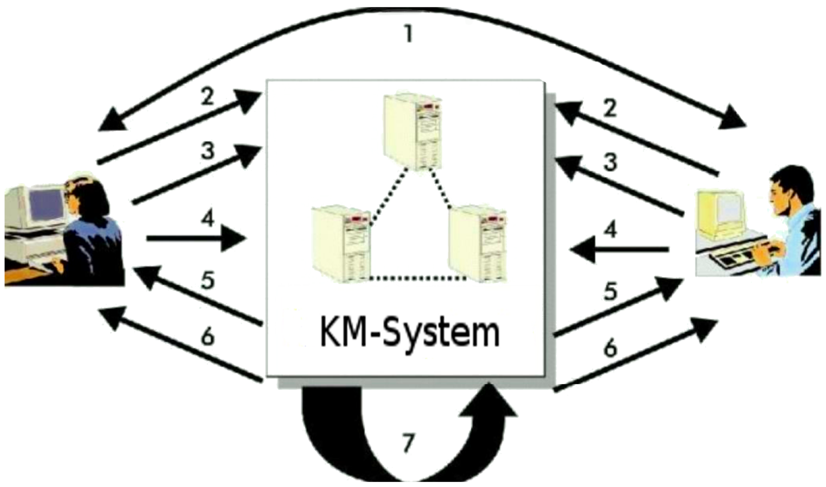


Figure 1.1: The Maurer-Tochtermann Model for Knowledge Management [Maurer & Tochtermann, 2002]

Description: All organizational aspects of knowledge management are subsumed by arrow 1. Arrow 2 and 3 symbolizes the various ways of explicit and implicit input of information into the KM-system respectively. Explicit input is the direct input of data by a user, e.g, the user enters data in a web form. Implicit input is such kind

[1]International Data Corporation (IDC): http://www.idc.com/

of information and knowledge entering into the KM-system that is produced as a by-product of the user's activities and hence not requiring any user effort. An example they give for implicit input is the sending of an announcement of an event to a group of persons by email. By sending this email over a KM-system, the information about the event is also put into a folder "upcoming events" and is automatically moved to a "past events" folder by the KM-system when the event is over.

Arrow 4 stands for the creation of new knowledge by the KM-system through user observations. The idea is to utilize the input of different sources, e.g., databases and employees, to derive general rules and procedures that can be suggested in similar situations. An example here is the observation of the interactions of a user with resources and applications to infer her tasks [Granitzer *et al.*, 2009a] as done in the research project DYONIPOS [Tochtermann *et al.*, 2006]. Further examples for task detection are given in Chapter 4. The research field *process mining* even works on the identification of process flows and the extraction of process models through user observations [Aalst & Weijters, 2004].

Arrow 5 indicates the classical query mechanism for information systems and databases. The user has to explicitly formulate a query for retrieving information from the KM-system. Here, information retrieval [Rijsbergen, 1979] is one of the main research areas that work on finding a good match between the query and the available information.

Arrow 6 indicates that the KM-system can generate and offer knowledge proactively, i.e., without an explicit request from the user. A sub-research area of information retrieval called context-aware information retrieval [Fuhr, 2005], also referred to as just-in-time information retrieval [Rhodes, 2000], focuses on exploiting the context of the user for retrieving information relevant to the user's current situation. In the research project APOSDLE learn and work support is provided based on the current task and the current topic the user is working on [Lindstaedt *et al.*, 2008b, 2009a].

Arrow 7 symbolizes that a KM-system has the possibility to generate new knowledge based on existing knowledge autonomously. Examples for the autonomous knowledge creation are an automatic interlinking of resources (e.g., documents, persons, emails web pages etc.), ontology learning and ontology alignment, inferencing with semantic web technologies [Sirin *et al.*, 2007]. An example for a knowledge discovery framework is the *KnowMiner* [Granitzer, 2008] that offers knowledge discovery services such as clustering, information extraction and information retrieval.

A KM-system is distinguished from a classical information system by the additional actions described through the arrows 3, 4, 6, and 7.

The actions symbolized by the arrows 4, 6 and 7 are of special relevance to this dissertation research, because this dissertation also shows that such kind of actions can indeed be implemented today. The goal of this dissertation research is to study and enhance automatic user task detection on the computer desktop through user observations (cf. arrow 4) in order to get to know the user and her tasks better. Detecting the user task is one important step towards task-specific support. For achieving this goal this dissertation proposes an approach that brings together machine learning and semantic web technology. More specifically a *user interaction context ontology (UICO)* is proposed for engineering novel features for the machine learning problem *"task detection"* (see Chapter 4 and 5). While user observations according to the MT-model can be the input from different sources, this dissertation research focuses only on a subset of possible user observations, the *"user context"*. User context is *"any information that can be used to characterize the situation of entities that are considered relevant to the interaction between a user and an application, including the user and the application themselves"* [Dey *et al.*, 2001]. The *user interaction context* is a subset of this user context and is defined as *"all interactions of the user with resources, applications and the operating system on the computer desktop. Resources are digital artifacts on the computer desktop, e.g., documents, web pages, emails, persons, appointments and notes"*. It strongly focuses on user interactions.

For the representation of the user interaction context semantic web technology is utilized. More specifically, an *"ontology"* is created. An ontology is *"an explicit specification of a conceptualization."* [Gruber, 1993]. It is a knowledge representation that defines what objects and what relationships between those objects exist. In user context detection research the representation of the user context as an ontology is referred to as an ontology-based user context model [Ötztürck & Amodt, 1997]. The ontology-based user context modeling approach has been advocated as being the most promising one [Baldauf *et al.*, 2007; Strang & Linnhoff-Popien, 2004] mainly because of its dynamicity, expressiveness and extensibility. Furthermore reasoning mechanisms [Sirin *et al.*, 2007] can be applied to an ontology-based user context model to infer new knowledge based on existing one (cf. arrow 7).

In order to do automatic task detection the user interaction context has to be automatically observed on the computer desktop. For this, context sensors which enable an unobtrusive capturing of all interactions of a user with her computer desktop are developed as part of this dissertation research. The captured user interaction context is then used to automatically populate the highly connected UICO. Automatic population here means an autonomous instantiation of concepts and creation of properties between concept instances of the UICO based on the observed and the automatically inferred user interaction context. The automatic population exploits the structure of user interface elements of standard office applications as well as preserves data types and relationships between the data through a combination of rule-based, information extraction and supervised learning techniques (cf. arrow 7). The UICO is a much richer representation of the user interaction context than is typically stored in attention metadata sensor streams [Wolpers *et al.*, 2007] since it preserves relationships that otherwise are lost. Furthermore, the UICO not only stores low-level data but also high level concepts. These high level concepts constitute connection points to other semantic desktop ontologies (see Section 2.2.1). Semantic desktop ontologies gain access to low-level user interaction context data through these high-level UICO concepts.

The highly connected UICO is naturally enabling a variety of context-aware applications (see Section 6.5) and *"mining"* activities for in-depth analyzes of for example user characteristics, actions, preferences, interests and goals. The results of such mining activities is new knowledge about the user (cf. arrow 7). One of the possible mining activities is task detection. Task detection in this dissertation research is seen as a classification problem which exploits the UICO for feature engineering (see Chapter 5). Once a task is detected based on the observed user interaction context, the knowledge about the detected task is added to the UICO (cf. arrow 7) and hence enables the provisioning of knowledge specific to this task to the user (cf. arrow 6).

The purpose of this chapter is to motivate this research, to give an overview of the research approach taken and to clarify what is "in focus" and "not in focus". The contributions of this research effort are also highlighted and the structure of the thesis including a short overview of the content of each chapter presented.

1.1 Motivation

Researchers from various fields have already started investing into getting a deeper understanding about the user's context. A selection of research areas are information retrieval that utilizes the user context for personalizing search queries [Budzik *et al.*, 2001; Fuhr, 2005; INTELLEXT Inc., 2007], knowledge maturing [Maier & Schmidt, 2007; Schmidt, 2005b] for providing a glue between the knowledge maturing phases as well as for the *decontextualization* and *recontextualization* during a maturing process, personal information management [Bellotti & Smith, 2000; Jones, 2007; Lansdale, 1988] to relate the user context to projects [Jones *et al.*, 2008; Schwarz, 2006], tasks [Catarci *et al.*, 2007], processes [Fenstermacher, 2005], topics [Sauermann *et al.*, 2005] and notes [Chirita *et al.*, 2006; Kleek *et al.*, 2007; Sauermann *et al.*, 2006a], technology enhanced learning [Schmidt, 2005a; Wolpers *et al.*, 2007] and work-integrated learning [Lindstaedt *et al.*, 2005, 2008a; Ulbrich *et al.*, 2006] to provide the appropriate learning material, as well as task and process management for discovering tasks [Dragunov *et al.*, 2005; Granitzer *et al.*, 2008; Kleek & Shrobe, 2007; Kröll *et al.*, 2006; Oliver *et al.*, 2006; Shen *et al.*, 2007] and process flows [Aalst & Weijters, 2004; Medeiros *et al.*, 2003; Weijters & Aalst, 2001].

The next sections motivate specifically the investigated research areas of this dissertation effort which are (i) *automatic user context detection* and (ii) *automatic task detection* and points to open questions in these areas.

1.1.1 Automatic User Context Detection

Several research areas strive to get to know the user better and to gather more insights about the user's current situation in order to enable better and more accurate support. Since the user cannot be bothered to manually enter information about her current situation, automatic means providing this information are required. Among these research areas two are information retrieval and information management.

As recently discussed in the information retrieval community, the emphasis of future information retrieval applications ought to be put on exploiting the user's context in order to increase the accuracy of retrieval results [Callan *et al.*, 2007]. For formulating a context-aware query to an information retrieval system (i) the context of the user has to be detected beforehand and (ii) the features utilized for a contextualized search and ranking procedure have to be selected. A manual specification of the user's context for automatic query formulation or manually constructing a context-aware query by the user is not an optimal solution since it requires expert knowledge as well as user effort. In the personal and organizational information management area research works on solving the *information fragmentation* problem [Bellotti & Smith, 2000] by relating resources on the computer desktop [Jones, 2007]) to projects, tasks, topics, processes etc. as well as to each other, i.e., capturing the user context of personal information management objects [Sauermann *et al.*, 2007]. Building these relations manually is cumbersome and requires a lot of effort from the user side. This additionally introduced user effort often leads to missing relations and to an incomplete picture about different *aspects of context* [Schwarz, 2006].

In both described areas above unobtrusive automatic user context detection mechanisms which observe the user and her interactions as well as algorithms for aggregating, abstracting and analyzing the observed data are highly appreciated. Furthermore sophisticated user context models are required to represent and relate different aspects of the user's context. Strang & Linnhoff-Popien [2004] distinguish in their survey between key-value models, markup scheme models, graphical models, object oriented models, logic-based models and ontology-based models. A more recent survey from [Baldauf *et al.*, 2007] agrees with the conclusions of Strang & Linnhoff-Popien [2004] and Dourish [2004] to favor an ontology-based context model. The main reasons mentioned are its dynamicity, expressiveness and extensibility. An ontology modeled with the ontology web language (OWL) [OWL, 2007] can provide this easy access requirement through its querying possibility with the query language SPARQL [Prud'Hommeaux & Seaborne, 2008]. A further advantage of using an ontology-based user context model is that an ontology can be validated through reasoners [Sirin *et al.*, 2007], i.e., checked for consistency. From a context-aware application and service perspective a user context model ideally allows an easy access to up-to-date contextual information to leverage this information for improving today's personal and organizational information management, intelligent retrieval systems as well as task and process management systems.

Three important open questions in the area of user context detection are (i) *What is the appropriate level of detail for sensing contextual information?*, (ii) *Which techniques should be used to aggregate, relate and reason about contextual information?* and (iii) *How to share and exchange contextual information?*. The answers to these questions depend on the application requiring the contextual information as well as on the capabilities of the computer environment on which the sensing process takes place (e.g., mobile or desktop environment).

1.1.2 Automatic Task Detection

Understanding the *"user context"*, more specifically the task of the user, in which resources are used, produced and distributed is of great interest, because once the links between the user's tasks and utilized resources is clear, research can start developing automatic means to detect the user's task. Automatically detecting the task of a user is valuable because once the task is known the user can be better supported with relevant information such as learning and work resources as well as with task guidance.

Task detection belongs to the field of *activity recognition* [Andrews *et al.*, 2004; Horvitz *et al.*, 1998, 1999; Philipose *et al.*, 2004]. In activity recognition the system observes a sequence of events, tries to determine and understand the goals of the user, and responds to them. Focusing on automatic task detection on the computer desktop a classical approach in recent research is to model task detection as a machine learning problem, more specifically as a classification problem. The need for automating the task detection process comes from the unrealistic cost of developing and maintaining a detailed knowledge base containing information about the user's different activities including the involved people, meetings, emails etc. [Mitchell *et al.*, 2006].

However, the focus so far was on using only text-based features and switching sequences [Chernov, 2008; Dredze *et al.*, 2006; Granitzer *et al.*, 2008; Horvitz *et al.*, 1998; Kushmerick & Lau, 2005; Lokaiczyk *et al.*, 2007; Oliver *et al.*, 2006; Shen, 2009] for detecting the user's task. None of the approaches so far rely on ontology models which are seen to be advantageous and beneficial in comparison to other user context modeling approaches in the user context detection research field [Baldauf *et al.*, 2007; Strang & Linnhoff-Popien, 2004]. Next to the advantages of an ontology-based user context model for user context detection, the advantages for automatic task detection are (i) the possibility to construct various types of features based on the (populated) ontology ranging from text-based, ontology structure-based, time-based to graph-based features, (ii) the easy access of the data representation with the rich query language SPARQL and (iii) the flexible data representation that allows to easily adapt to the requirements of various domains.

In the area of task detection freely available standard datasets for studying task detection approaches are still missing. Furthermore the representative number of field studies and laboratory experiments performed to gather task detection datasets is low. Some reasons for this are (i) the tedious, labor-intensive and expensive process for designing and carrying out user studies and (large-scale) experiments for the data collection and the data labeling [Oliver *et al.*, 2006], (ii) the amount of time required for developing user context observation mechanisms, and (iii) the difficulty of making the recorded usage data anonymous [Chernov *et al.*, 2008].

Three important open questions in the area of automatic task detection modeled as a machine learning problem are (i) *What kind of tasks can be automatically detected?* (ii) *Which context features are most discriminative between tasks?* and (iii) *Which machine learning algorithm should be used?*. The answers to these questions have only been investigated in limited ways and require further research efforts, both in a single and in different domains as well as in experimental and real world settings. These questions are investigated and addressed in this dissertation research through the approach described bellow.

1.2 Research Question and Approach

The goal of this dissertation research is to automatically detect the task of a user on her computer desktop in order to enable task-specific support. More specifically, this research strives to study and enhance automatic task detection on the computer desktop via an ontology-based user interaction context model. Hereby, the main research hypothesis of this dissertation effort is:

> *"The accuracy of automatic task detection can be enhanced by features engineered from an ontology-based user interaction context model in comparison to features and feature combinations of existing approaches."*

In Figure 1.2 the complete *user interaction context ontology task detection pipeline* (UICO pipeline) that sums up the steps of the task detection process is presented. It starts from observing the user's interactions and ends at the detected user task. The *automatic user context detection* approach and the *automatic task detection* approach are displayed in the left and right area of Figure 1.2 respectively. The UICO pipeline is used for testing the main research hypothesis as well as for answering the sub-research questions in the areas of automatic user context detection and automatic task detection presented bellow.

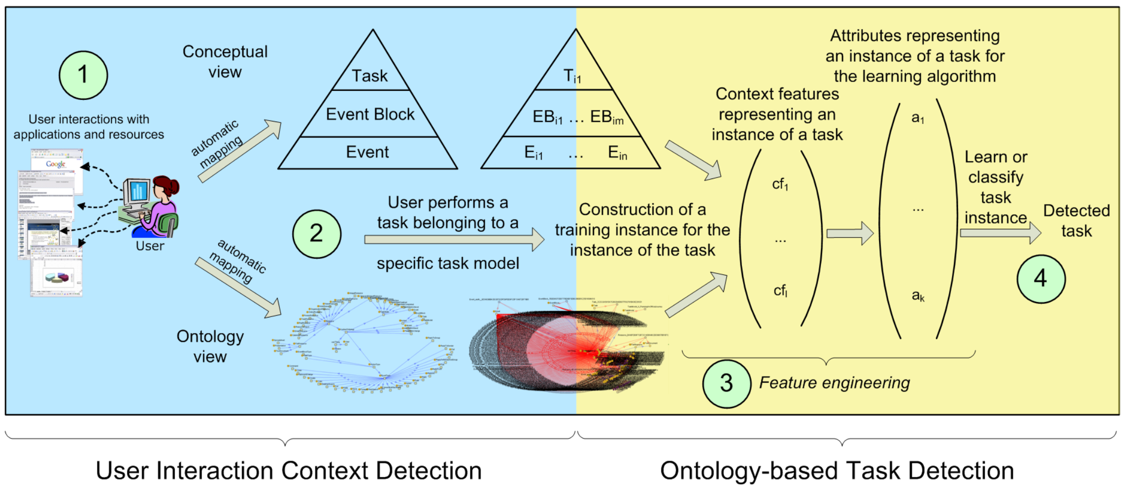

1
User interactions with applications and resources
User
Conceptual view
automatic mapping
Task
Event Block
Event
T_{i1}
EB_{i1} … EB_{im}
E_{i1} … E_{in}
2
User performs a task belonging to a specific task model
automatic mapping
Ontology view
Construction of a training instance for the instance of the task
Context features representing an instance of a task
cf_1
…
cf_l
Attributes representing an instance of a task for the learning algorithm
a_1
…
a_k
Learn or classify task instance
Detected task
4
3
Feature engineering
User Interaction Context Detection
Ontology-based Task Detection

Figure 1.2: This figure visualizes the complete user interaction context ontology task detection pipeline(UICO pipeline) starting from (1) the automatic unobtrusive user interaction observation mechanisms to (4) detecting the user's task. The automatic population of the user interaction context model (2) is displayed in two ways (i) the instantiation of entities in the conceptual model (conceptual view) and (ii) in the ontology model (ontology view). In (3) the feature engineering process for transforming a task instance into a training instance is shown. The training instance is further fed to attribute selection and learning algorithms for (4) detecting the task of the user.

1.2.1 Automatic User Context Detection

A conceptual user interaction context model including its realization as an ontology-based user interaction context model (UICO) is proposed for automatically constructing the relations between user interactions, resources and applications by unobtrusively capturing low-level contextual attention metadata [Wolpers *et al.*, 2007]. Automatic user interaction context observation mechanisms, referred to as *context sensors*, were developed for automatically capturing the user's interactions with resources, applications and the operating system on the computer desktop. Based on the observed user interaction context automatic population mechanisms for the UICO comprising of static rules, information extraction and machine learning techniques were developed and integrated. The concepts and relations present in the UICO were created via a bottom up approach. New concepts and relations were added to the UICO based on (i) new data and metadata about the user interaction context captured by context sensors and (ii) based on the results of the user interaction context analysis algorithms. An important requirement was that the state, the relations and the entities of the model are synchronous on a single user interaction basis. This means that each user interaction with a resource and an application is directly reflected in the UICO and hence up-to-date at all times.

User context observation experiments in three laboratory settings were conducted to evaluate the applicability of the proposed user context detection approach for enhancing automatic task detection.

Following sub-research questions were investigated:

- To what extend is it possible to automatically and unobtrusively observe how users interact with resources and applications available on their computer desktops?
- Which aggregation levels can be reached with what kind of techniques when utilizing the automatically captured contextual information about the user's interactions with resources, applications and the operating system on the user's desktop?

1.2.2 Automatic Task Detection

The research approach to automatic task detection is based on the user context detection approach described in Section 1.2.1 above. Novel context features for automatic task detection were engineered based on the populated UICO and tested for their discriminative power among users' tasks and task types. A variety of standard classifiers were evaluated with the engineered UICO features and UICO feature combinations on three different datasets. Among the classifiers were the Naïve Bayes, the Linear Support Vector Machine (SVM) with various cost parameter settings, the J48 decision tree and the k-Nearest Neighbor (KNN-k) algorithm with different k settings. Since no freely-available standard datasets for evaluating automatic task detection are available, the approach was to create new datasets for evaluating this research's automatic task detection approach. This research reports on the design and execution of three large-scale laboratory task detection experiments in which over 500 task executions were recorded. The sum of all collected datasets were about 1500 megabytes. The task detection performances of the UICO features/feature combinations evaluated on the recorded datasets were compared to ones of feature/feature combinations of already existing approaches. Significance tests were performed for ranking features and feature combinations according to their discriminative power between tasks as well as ranking classifiers according to their achieved task detection performance for each laboratory experiment.

Following sub-research questions were investigated:

- What kind of tasks can be detected?
- Are there user interaction context features/feature combinations automatically observable by context sensors on the computer desktop that influence the performance of automatic task detection more than others?
- Which classifiers should be used for automatic task detection?

- Does the computer desktop environment influence the performance of automatic task detection?
- Can a single expert train a classifier in advance such that it detects other users' tasks?
- Can a group of experts train a classifier such that it recognizes the tasks of another user?
- Can a classifier be trained with predefined standard task executions classify personal tasks correctly?

1.2.3 Realization

The research for this dissertation was performed during the DYONIPOS and the KnowSe project, both carried out as part of the authors work at the Know-Center[2], Austria's Competence Center for Knowledge Management.

The *DYONIPOS* (DYnamic ONtology based Integrated Process OptimiSation)[3] project [Tochtermann *et al.*, 2006] was a two-year national funded research project financed by the Austrian Research Promotion Agency[4] within the strategic objective FIT-IT. The DYONIPOS project partners were from the Know-Center, Hewlett Packard Austria[5], the Institute for Information Systems and Computer Media of Graz University of Technology[6], m2n - consulting and development gmbh[7] and Austria Federal Ministry of Finance[8] as a use case partner. DYONIPOS targeted to resolve the dilemma of the organizational need for standardization and control on the one hand and the creative freedom required by knowledge workers on the other hand. The idea was to support both, the process executer and the process engineer. Process executers were provided with supportive resources that were useful and relevant to their current tasks and processes. Hereby, the user interactions with the computer desktop were automatically recorded in order to infer the current task and process. Process engineers were supported in reviewing and analyzing the recorded task and process executions in order to validate and to enhance existing standard processes as well as to derive new ones. The DYONIPOS prototypes were evaluated in the business environment of the use case partner, the Austria Federal Ministry of Finance.

The conceptual model, the Semantic Pyramid (see Section 3.2), most of the user interaction context sensors (see Section 3.4.1) and aggregation mechanism described in Section 3.5 originated from the work of the author in the DYONIPOS project. A screenshot of the first prototype is given in Figure 6.1.

The *KnowSe* (Knowledge Services) project[9] is one of the research projects of the *Knowledge Services* division of the Know-Center including user context detection and attention metadata extraction. As part of this project the equally named service-oriented knowledge

[2]`http://www.know-center.at`
[3]`http://www.dyonipos.at`
[4]`http://www.ffg.at`
[5]`http://www.hp.com/at`
[6]`http://www.iicm.tugraz.at`
[7]`http://www.m2n.at`
[8]`https://www.bmf.gv.at`
[9]`http://en.know-center.at/forschung/knowledge_services`

services framework (*KnowSe*) has been developed which forms the basis for dynamically orchestrating a large variety of intelligent knowledge services. KnowSe's goal is to provide knowledge services, highly contextualized to a persons work context, interconnected with a multitude of knowledge sources as well as is able to detect patterns and make inferences based on uncertain information. The user interaction context detection and the ontology-based task detection approach has been integrated as services in the KnowSe framework. The user interaction context ontology (UICO) is a key ingredient of KnowSe, by providing a coherent view on and a single access point to the data and information stored about the user's interaction context. The user interaction context detection mechanisms developed in the DYONIPOS project were refined during this dissertation in order to automatically populate the UICO by discovering new concepts, and deriving inter-concepts relations. Hereby also the knowledge discovery framework of the Know-Center, the KnowMiner [Granitzer, 2008], is utilized for information extraction and information retrieval. The ontology-based task detection approach proposed in this dissertation further enriches the available information about the user through automatically identifying the user's tasks. This allows KnowSe to make hidden information and connections between tasks, resources, and people visible and usable. Examples of knowledge services provided by KnowSe are context-aware information retrieval, user interruptibility and visual reflection services. Further details about the knowledge services including screenshots are presented in Section 6.5.

1.3 Focus and Non-Focus

This sections gives a compressed overview of what is in focus and not in focus of this research.

Focus:

- Studying the user interaction context in relation to the user's tasks on the computer desktop.
- Development of user interaction context sensors, more specially application and windows operating system sensors for commonly used applications of office workers.
- Design of a conceptual model of the user interaction context including its realization as an ontology-based user interaction context model.
- Bottom up creation of a user interaction context model that stores the data and meta-data observed by context sensors as well as the relations and concept instances discovered by user context analysis algorithms.
- Design and development of algorithms and mechanisms for automatically populating the ontology-based user interaction context model.
- Design and execution of three laboratory experiments for collecting datasets with tasks form different domains, different users and different tasks as well as task types.
- Discovery of highly discriminative user context features for automatic task detection based on the datasets of three laboratory experiments.
- Studying the task detection performance of the standard classifiers: the Naïve Bayes, the Linear Support Vector Machine (SVM) with various cost parameter settings, the J48 decision tree and k-Nearest Neighbor (KNN-k) with different k settings. The performance

of the classifiers were evaluated on the collected datasets with both novel features/feature combinations engineered from the user interaction context ontology as well as well-known features/feature combinations of already existing approaches.

Non-Focus:

- *"Context"* and *"user context"* in other research areas than in computer science.
- Developing new machine learning or attribute selection algorithms or methods.
- Tackling the task switch detection problem or the online learning of tasks.
- The evaluation of context-aware proactive information delivery based on the populated user interaction context model.
- Proposing a new context modeling technique. This research utilizes the ontology-based context modeling approach which several researchers noted as advantageous [Baldauf *et al.*, 2007; Strang & Linnhoff-Popien, 2004].
- Design nor development nor evaluation of context-aware knowledge-work or learn support applications or systems.
- The detection of the user's context other than the user interaction context on the computer desktop.
- Privacy issues regarding user context detection or task detection.

1.4 Contributions

This dissertation investigates user context detection and task detection in order to enhance automatic task detection on the computer desktop for enabling task-specific support. Since user context detection and task detection are relevant for several research areas such as technology-oriented knowledge management, personal information management, human-computer interaction, work-integrated learning as well as information retrieval this section describes the contributions of this research effort in greater detail. The contributions of this thesis are described bellow in the order they are presented and discussed in this dissertation:

First, this thesis introduces the *"user interaction context"* as a subset of Dey *et al.* [2001]'s definition of the user context. The user interaction context is defined as *"all interactions of the user with resources, applications and the operating system on the computer desktop. Resources are digital artifacts on the computer desktop, e.g., documents, web pages, emails, persons, appointments and notes"*. For this user interaction context a conceptual model referred to as the *semantic pyramid* is presented. Furthermore, a bottom up approach is shown how to realize this conceptual model as an ontology-based user context model, referred to as the *user interaction context ontology* (UICO).

The UICO is a richer representation of the user interaction context than normally stored in context sensor streams (see Section 2.3) since it preserves relationships that otherwise are lost. This rich representation can be utilized by several types of applications requiring user interaction context data as shown in Section 6.5. Examples for such applications are context-aware information retrieval, measuring user interruptibility, or visually representing the relationships between tasks, users and resources.

In user context detection research several context sensor data formats, such as Attention.Xml, Contextualized Attention Metadata (CAM), APOSDLE or DYONIPOS event log XML exist which make it difficult to exchange user context data. These data formats very much depend on the context sensors and do not include aggregated or inferred contextual information. The UICO is built bottom-up based on the data and metadata delivered by context sensors and includes aggregated and inferred contextual information. It could serve as a starting point for the discussion about a shared representation of the user context and hence pave the way to a shared user context data exchange format.

Second, this thesis presents mechanisms and techniques to automatically and unobtrusively observe the user interaction context on the computer desktop which automatically populate the user interaction context ontology in real-time. These include (i) context sensors that capture the user interaction context in a variety of standard office and desktop applications, (ii) algorithms and information extraction techniques for discovering resources (e.g., documents, files, folder, persons, web links, etc.) in the raw sensor stream and automatically relating resources, (iii) heuristic algorithms for aggregating and abstracting low-level context sensor *events* to blocks of user interactions on resources (*event blocks*).

The *Maurer-Tochtermann Model for Knowledge Management (MT-Model)* [Maurer & Tochtermann, 2002] includes the creation of new knowledge based on user observations (cf. MT-Model arrow 4). This thesis shows that the creation of new knowledge about the user interaction context based on unobtrusive user observations is possible. The knowledge about the user interaction context is represented as an ontology, more specifically in the automatically populated UICO. The populated, highly-connected UICO is naturally enabling a variety of *"mining"* activities for in-depth analyzes of user characteristics, actions, preferences, interests, goals, application usage, social network, user competences, access and usage of learning materials and tasks. The results of such mining activities is new knowledge about the user (cf. MT-Model arrow 7).

Through combining several populated UICO's mining activities on an organizational level are enabled for creating new knowledge about the organization and its daily business. Examples are information and process flows, topic expert identification, social networks, cooperation between groups and divisions, topical trends as well as access and usage of organizational resources such as paid subscription services, licensed applications or servers.

The relationships preserved in the UICO are also valuable for personal and organizational information management research that strives to relate projects, tasks, processes, persons and other digital resources in order to target the information fragmentation problem [Bellotti & Smith, 2000]. The mechanisms for automatically constructing these relationships reduce the time required by users to manually create them.

For human-computer interaction research the user interaction context data is interesting in order to get to know how users interact with applications, i.e., the application usage. An example is the identification of navigational paths of novice users for identifying errors in the user interface design [Fenstermacher & Ginsburg, 2002]. In adaptive systems research, a sub-area of human-computer interaction, the user interaction context could be utilized to automatically adapt the user interface, i.e., remove/hide unimportant user interface elements, in order to ease the execution of the current task of the user.

In work-integrated learning the user interaction context can serve for building rich user/learner profiles [Lindstaedt *et al.*, 2009a], the automatic annotation of learning resources about how these resources are used by the learner, for enhancing the retrieval process [Ochoa & Duval, 2006], and for computing the user's competences [Ley *et al.*, 2008] based on the access and usage of learning resources as well as the performed tasks.

Third, a flexible, reusable and service-oriented application, referred to as *Context OBservation Service* (COBS) is one of the technical outcomes of this dissertation research. It can be utilized to enrich applications and systems with real-time user interaction context information for various purposes (e.g., context-aware information retrieval, task detection or user interruptions) as well as for various domains (e.g., technology enhanced learning and work-integrated learning, computer supported collaborative work, personal information management or semantic desktop systems). Examples for proof of concept implementations are described in Section 6.5.

Fourth, this thesis introduces the *ontology-based task detection approach* and evaluates it on three datasets containing over 500 tasks from over 40 users. These datasets stem from three laboratory experiments designed and performed part of this dissertation research. The evaluation results confirm that automatic task detection can be successfully performed by applying machine learning techniques. They show that routine as well as knowledge-intensive tasks can be classified with a high accuracy based on the automatically captured user interaction context with the ontology-based task detection approach.

A comparison with existing task detection approaches highlights that the features engineered based on the user interaction context ontology for training the machine learning algorithms significantly outperformed features and feature combinations proposed by existing approaches in the settings explored here. Furthermore novel context features were discovered that significantly enhance automatic task detection. The best performing single features were the *acc. obj. name* feature, the *window title* feature, the *used res. metadata* feature, the *acc. obj. value* feature, the *datatype properties* feature and the *acc. obj. role* feature. The best overall feature combinations were the combinations of features of the *application category* and the combination of all 50 features (*All Categories*). The best results for a specific task detection problem were achieved by combining the top k single performing features.

Task detection is classically seen as a machine learning problem. In machine learning feature engineering is a key aspect to achieve good results. The features and feature combinations with a high discriminating power for classifying different types of tasks identified by this dissertation research are an important contribution to task detection research, because good features are difficult to find. Furthermore, the ontology-based task detection approach includes 50 distinguished features engineered from the user interaction context ontology that cover various aspects of the available user interaction context data and thus could prove to be adaptive for various tasks and domains.

Fifth, this thesis offers evidence of influencing factors for the task detection performance (see Section 7.3). In particular it investigates the following factors: (i) types of tasks (routine and knowledge-intensive), (ii) computer environment, (iii) *specific goals* of tasks, (iv) *"offline"* and *"expert user(s) task"* training, (v) context features and context feature combinations and (vi) machine learning algorithms.

The identification and investigation of influencing factors to the task detection problem enriches our understanding of the *task detection* phenomenon. The better we understand this phenomenon, the better we can identify settings in which task detection can be applied and the better we can tune our algorithms for automatically detecting the user's tasks.

1.5 Thesis Outline

This thesis consists of 8 chapters. The structure of this thesis is displayed in Figure 1.3 and further described in the following:

Chapter 1 outlines the motivation for this work, gives an overview of the investigated research questions and the followed approaches. A listing of the contributions of this work to the area of human-computer interaction and information systems as well as an overview of the structure of this thesis round off the first chapter.

Chapter 2 gives an overview of the related work in the areas of *"context"* and *"context-awareness"*. Past and current research about user context modeling, user context observation, user context utilization and user context exploitation as well as application areas for user context are presented. A special focus is put on approaches after 2000 that have not already been presented as part of Dey's dissertation [Dey, 2000] or in the survey papers by Strang & Linnhoff-Popien [2004] and Baldauf *et al.* [2007]. A selection of highly relevant approaches to the user context detection approach of this dissertation research is discussed in further detail. The chapter finishes by highlighting application areas for user context.

Chapter 3 presents the *user interaction context* approach starting by introducing the term *"user interaction context"* as a subset of the user context as defined by Dey *et al.* [2001]. The *"user interaction context"* is defined as *"all interactions of the user with resources, applications and the operating system on the computer desktop. Resources are digital artifacts on the computer desktop, e.g., documents, web pages, emails, persons, appointments and notes"*. The conceptual model, referred to as the *Semantic Pyramid*, as well as a realization of this model as an ontology-based user context model, referred to as the *user interaction context ontology* (UICO), are also explained in this chapter. After presenting the UICO as a semantic representation of the user interaction context this chapter elaborates on automatic UICO population mechanisms. Hereby it gives an overview of the context sensors that were implemented in order to automatically and unobtrusively observe the user interaction context on the computer desktop. Furthermore it shows the techniques and the methods used to aggregate and to transform the raw context sensor data into concept instances and relations of the UICO. Next to the automatic population of the ontology which is based on the automatically observed user interaction context also the computation and creation of the relations between the concept instances are explained.

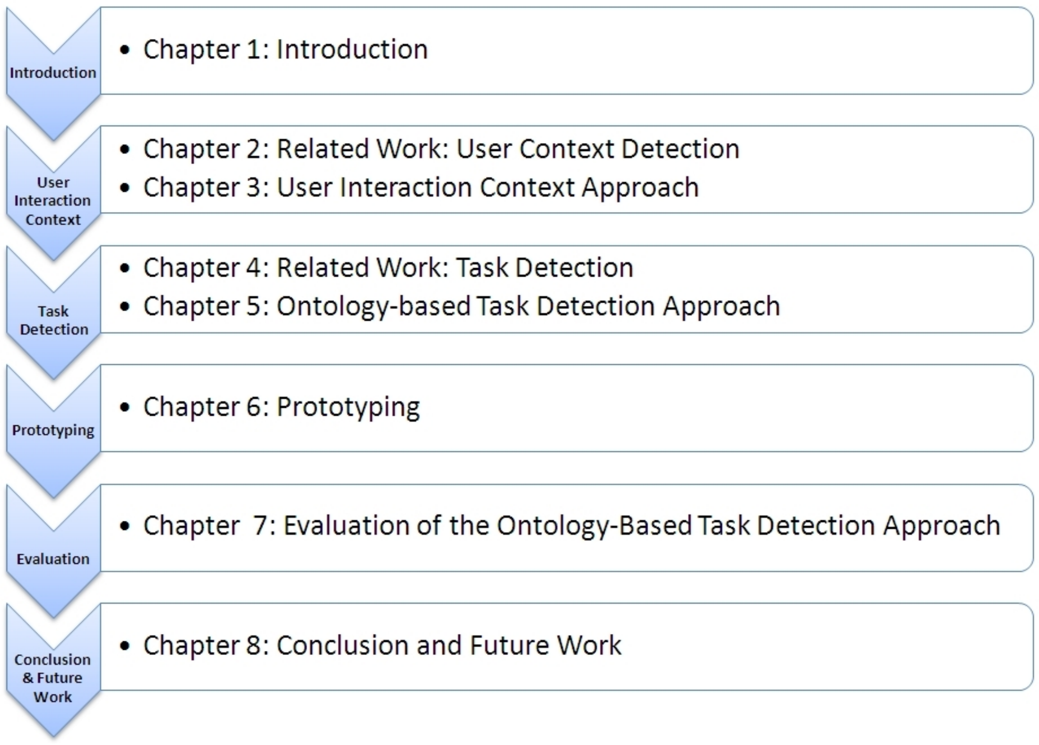

Introduction
• Chapter 1: Introduction
User Interaction Context
• Chapter 2: Related Work: User Context Detection
• Chapter 3: User Interaction Context Approach
Task Detection
• Chapter 4: Related Work: Task Detection
• Chapter 5: Ontology-based Task Detection Approach
Prototyping
• Chapter 6: Prototyping
Evaluation
• Chapter 7: Evaluation of the Ontology-Based Task Detection Approach
Conclusion & Future Work
• Chapter 8: Conclusion and Future Work

Figure 1.3: Structure of this thesis

Chapter 4 discusses the related work on task detection. A selection of relevant approaches to this research regarding automatic task detection in emails, in web browsers and on the complete computer desktop are highlighted in greater detail. A discussion of the relevant literature in respect to the research goals of this dissertation rounds off this chapter.

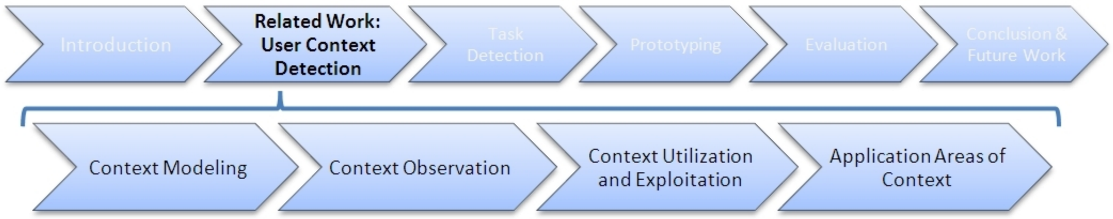
Chapter 5 introduces the *ontology-based task detection approach* which is a novel approach to task detection that combines semantic technologies with machine learning in order to improve task detection. In this approach the user interaction context ontology (UICO) is utilized to engineer novel features and feature combinations for doing automatic task detection. 50 features classifiable into 5 feature categories were extracted from the UICO. On behalf of the *ontology-based task detection pipeline* this chapter explains the transformation of the user interaction context stored in the UICO to class/training instances of machine learning algorithms (*training instance construction*) and elaborates on the construction and preprocessing of each feature (*feature engineering*).

Chapter 6 is intended to give an overview of the prototypes built for observing the user interaction context. Next to the architecture also the graphical user interfaces of the prototypes are presented. In this chapter also the architecture of the designed and implemented toolkit for studying and evaluating the task detection performance on the gathered task usage datasets is given. Application prototypes in the area of user interaction context visualization, context-aware information retrieval and user interruptions based on the user interaction context ontology are described and visualized.

Chapter 7 presents the evaluation of the ontology-based task detection approach. It gives insides into the methodology used to evaluate the approach and explains the performance measures used to assess task detection performance. Three large-scale laboratory experiments conducted as part of this thesis research for evaluating different aspects of the task detection performance of the ontology-based task detection approach in comparison to existing approaches. Aspects regarding automatic task detection studied in this chapter are: (i) types of tasks (routine and knowledge-intensive), (ii) computer environment, (iii) *specific goals* of tasks, (iv) "offline" and "expert task" training, (v) context features and context feature combinations and (vi) machine learning algorithms. The design and execution of the experiments for collecting the task usage datasets as well as the conducted task detection evaluations are presented and discussed for each laboratory experiment. This chapter closes with an attempt to generalize the findings of the experiments' and a listing of open questions.

Chapter 8 concludes this thesis It reflects on the set goals (self-assessment) and presents a number of interesting possibilities for future research based on the outcomes of this dissertation.

2 Related Work: User Context Detection



This chapter describes the related work regarding *"context"* and *"context-awareness"* as used in computer science. Past and current research is described for context modeling in Section 2.2, for context observation in Section 2.3 and for context utilization and context exploitation in Section 2.4. A special focus in the presentation of the related work is put on research after 2000 that has not already been discussed as part of Dey's dissertation [Dey, 2000] or in the survey papers by Strang & Linnhoff-Popien [2004] and Baldauf *et al.* [2007]. A selection of highly relevant approaches to the user context detection approach of this dissertation research is discussed in further detail. This chapter finishes by highlighting application areas for user context in Section 2.5.

2.1 Introduction

Massive amounts of digital information are available to to a computer user today. It is important to find the information that is relevant to a user's context and also to put information into context, i.e., to clarify in which setting information is used and produced. This research work focuses on a subset of the user context which is introduced as the user interaction context. For representing contextual information about the user a model is required. Various approaches to model the user context have been surveyed and discussed in [Baldauf *et al.*, 2007; Strang & Linnhoff-Popien, 2004]. Both surveys conclude to favor the ontology-based approach mainly because of its dynamicity, expressiveness and extensibility. In 1994 the term *context-aware* was first defined by [Schilit *et al.*, 1994] and referred to location, identities of nearby people and objects and changes to those objects. In 2001 Dey *et al.* [2001] provided a well elaborated list of historical definitions of the term context and introduced a new definition that has become widely accepted in computer science literature nowadays. They defined context as *"any information that can be used to characterize the situation of entities that are considered relevant to the interaction between a user and an application, including the user and the application themselves"*. Furthermore they also described requirements of a conceptional framework for context-aware applications. All the definitions of context in computer science have in common that there are features and relations that describe what context is. Zimmermann *et al.* [2007] elaborated *"five fundamental categories of context information"* and Schwarz [2006] pointed out different *"aspects of context"*.

Dey [2000] evaluated several context capturing frameworks available until 2000. Most of the frameworks were able to sense virtual features, for example, currently visit web page or content of currently viewed document, as well as physical context features, for example, location, screen resolution, camera pictures, or temperature. None of these evaluated frameworks used semantic technologies for representing the relations between the captured context features or for describing the user context. Since application and hardware change in the course of time and new applications become popular on the desktop, the ancient sensors may be outdated. Most frameworks had their own idea which sensors are required to observe information about the user context as well as how this contextual information had to be processed and stored. Dey [2000] described a process to decide which contextual information about the user should be observed and passed to the application. In his view the application is the steering unit in this decision process. A selection of context capturing frameworks since 2001 are presented and discussed in Section 2.3.

2.2 Context Modeling

Strang & Linnhoff-Popien [2004] wrote a survey paper about the different context modeling approaches and models that have been around till then. They distinguished between key-value models, markup scheme models, graphical models, object oriented models, logic-based models and ontology-based models. A rather critical discussion about the possibility to define or represent context can be found in [Dourish, 2004] where Dourish stated that *"context is slippery"*, because context is continuously renegotiated and defined in the course of the user's action. Both concluded that for the requirements they defined for ubiquitous computing systems, the ontology-based approach for modeling context was the most appropriate. A more recent context modeling survey from [Baldauf *et al.*, 2007] also favored the ontology-based approach and hereby went hand in hand with Strang & Linnhoff-Popien [2004] and Dourish [2004] conclusions about ontology-based context models. In Section 3.3 the approach of this dissertation research went along with their findings and followed an ontology-based approach for modeling user interactions with resources, applications and the operating system. One of the first ontology-based user context models has been proposed by Ötztürck & Amodt [1997] in 1997 in which they analyzed psychological experiments in a clinical setting. They investigated the difference between recall and recognition of several issues in combination with contextual information and derived the necessity of normalizing and combining knowledge from different domains [Strang & Linnhoff-Popien, 2004].

2.2.1 Personal Information Ontologies

The first vision of personal information management (PIM) systems was pointed out by Vannevar Bush who noted that the human mind operates by associations and that we should *"learn from it"* in building *Memex* [Bush, 1996]. This theory of how the mind works is also shared by Anderson

[1983] who described the architecture of cognition. Ontologies can be seen as a possible way of modeling these connections between entities. By applying the association modeling approach to the user's desktop environment several researchers created an ontology for the user's context. For example, ontologies were suggested by Lindstaedt *et al.* [2008b] for modeling the user's domain, her tasks and her learning goals as well as the corresponding interrelations. Maier & Sametinger [2007] proposed a top-level ontology for documents by focusing on the dimensions when, what, where, who, why and how in order to make the access and the retrieval of documents smarter through federating existing meta-data standards and ontologies.

Selected approaches that focus on the modeling of ontologies for personal information management on the user's computer desktop domain are briefly described in this section. These approaches have been chosen because they are the most related ones for this research in terms of modeling the user interaction context by means of an ontology (see Section 3.3). Other more general ontology-based context modeling approaches are discussed by Strang & Linnhoff-Popien [2004] and for ubiquitous systems by Baldauf *et al.* [2007].

2.2.1.1 Haystack

The *Haystack* project's goal is *"to develop a tool that allows users to easily manage their documents, e-mail messages, appointments, tasks, and other information"* [Huynh *et al.*, 2003]. Haystack utilizes an ontology-based approach for interrelating the user's information and addresses four specific expectations of the user: (i) maximum flexibility for the user how to organize information, (ii) support for user-defined ontologies, (iii) easy manipulation and visualization of information in ways appropriate to the task at hand as well as (iv) delegation of repetitive information processing tasks to agents. For providing homogeneous access Haystack uses the *resource description framework* (RDF)[1] and ontologies for organizing, manipulating and retrieving personal information. A more detailed specification of the ontology, its concepts and its properties was not found in the literature or on the project's web page. Hence it was difficult to discuss the applicability for task detection.

2.2.1.2 Learning In Progress (LIP) Ontology

The *learning in progress (LIP) ontology* [Schmidt, 2007] represents four types of context features: (i) *personal*, (ii) *social*, (iii) *organization* and (iv) *technical*. Personal ones include the previously acquired knowledge or competencies, goals, preferred interactivity level and semantic density, preferred communication channels and current time capacity/time pressure. Social ones refer to the information about the user's social network whereas organization ones represent the units, the roles and the processes the user is involved in. The technical features are operating system, used applications, network and audio information. All four types of features can be observed in Figure 2.1. The LIP ontology was developed in order to enrich learning solutions with context-awareness.

[1]Resource description framework (RDF): `http://www.w3.org/RDF/`

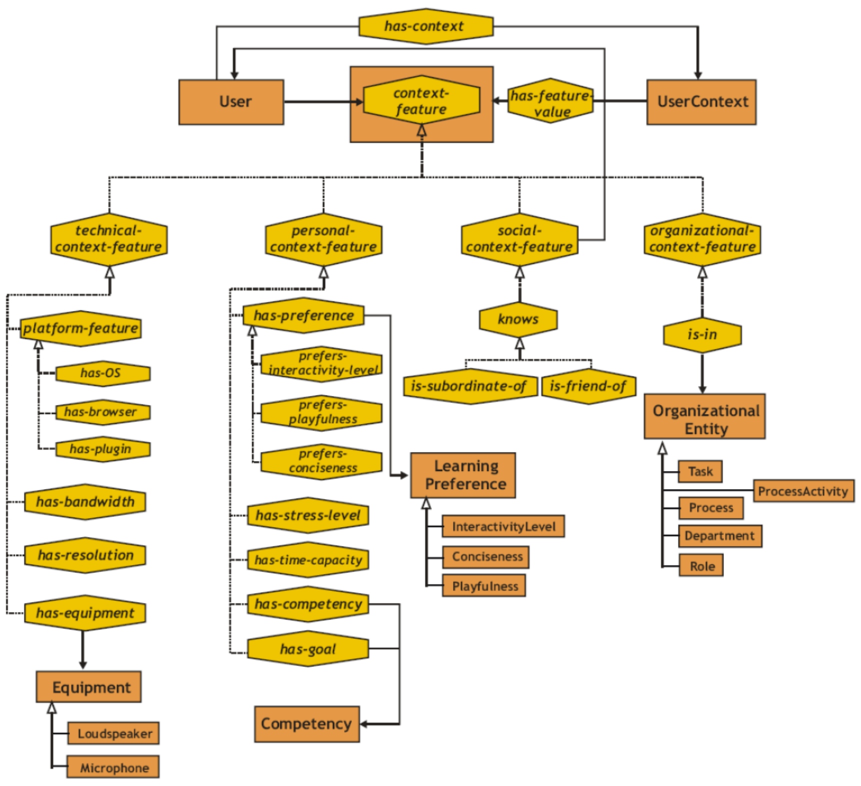


Figure 2.1: This figure shows the Learning in Progress (LIP) ontology with the following four types of context features: (i) personal, (ii) social, (iii) organization and (iv) technical. This figure was published in [Schmidt, 2007].

The LIP ontology does not have any concepts or properties for modeling desktop resources, applications or user interactions. From the perspective of the user interaction context the LIP can be seen as a high-level ontology and hence not appropriate to reach the goals of this dissertation research.

2.2.1.3 Native Operations Ontology (NOP)

The *Native Operations* (NOP) ontology[2] which is used in the Mymory project [Biedert *et al.*, 2008] models native operations (e.g., `AddBookmark` or `CopyFile`) on *generic information objects* (e.g., emails, bookmarks or files) which are recorded by system and application sensors. *"NOPs correspond to actions undertaken during work"* [Biedert *et al.*, 2008]. The `DataObject` concepts describe several desktop resources in a coarse granular way. The NOP ontology has 102 concepts, 40 datatype properties and 22 object type properties. The majority of the concepts are sub-concepts of the `NativeOperation` concept which shows a strong focus on modeling the types of operations. The `DataObject` concept has 12 sub-concepts that contains the application and desktop resources as concepts as well as an abstract concept called `PimoConcept`. The kinds of datatype and objecttype properties indicate that the focus of this ontology was not to model

[2]NOP Ontology: `http://usercontext.opendfki.de/wiki/NopOntology`

resources but instead the *“native operations a user does when using a device”* [Biedert *et al.*, 2008]. The picture of the NOP ontology in Figure 2.2 illuminates this as well. The user interaction context also includes all user interactions with applications and the operating system. Since NOP is not about observing applications in great detail or capturing all user interactions (mouse or keyboard interactions) the user interaction context can not be modeled with this ontology without extending it. It was decided not to extend this ontology because at the time the first version of the NOP was published on the Internet the user interaction context ontology (UICO) (see Section 3.3) as a realization of the semantic pyramid (see Section 3.2) was almost finished.

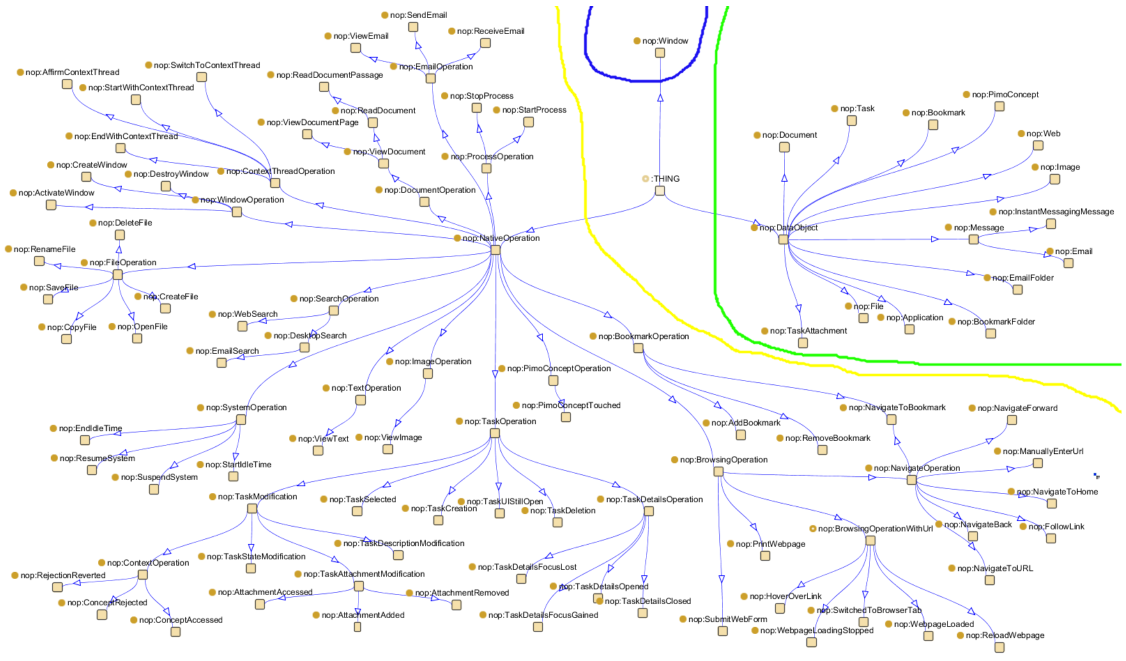


Figure 2.2: The Native Operations (NOP) ontology visualized in Protégé. In the left area this figure shows the `NativeOperation` concepts, in the right area the `DataObject`, and in the top area the `Window` concept.

2.2.1.4 Personal Information Model Ontology (PIMO)

The *Personal Information Model Ontology* (PIMO) [Sauermann *et al.*, 2007] visualized in Figure 2.3 was first developed in the research project GNOWSIS [Sauermann, 2003, 2005].
The PIMO was further refined in EPOS [Sauermann *et al.*, 2006a] and NEPOMUK[3] [Groza *et al.*, 2007]. The purposes of the PIMO is to allow the user to model her personal information space. PIMO provides a set of “high level” concepts such as people, places, topics, documents, and time [4]. These *personal information objects* [Sauermann *et al.*, 2007] can be extended manually by the user. During the work done in the NEPOMUK project, the PIMO became the central

[3]NEPOMUK’s project web page: `http://nepomuk.semanticdesktop.org`
[4]Personal Information Management Ontology (PIMO): `http://dev.nepomuk.semanticdesktop.org/wiki/PimoOntology`

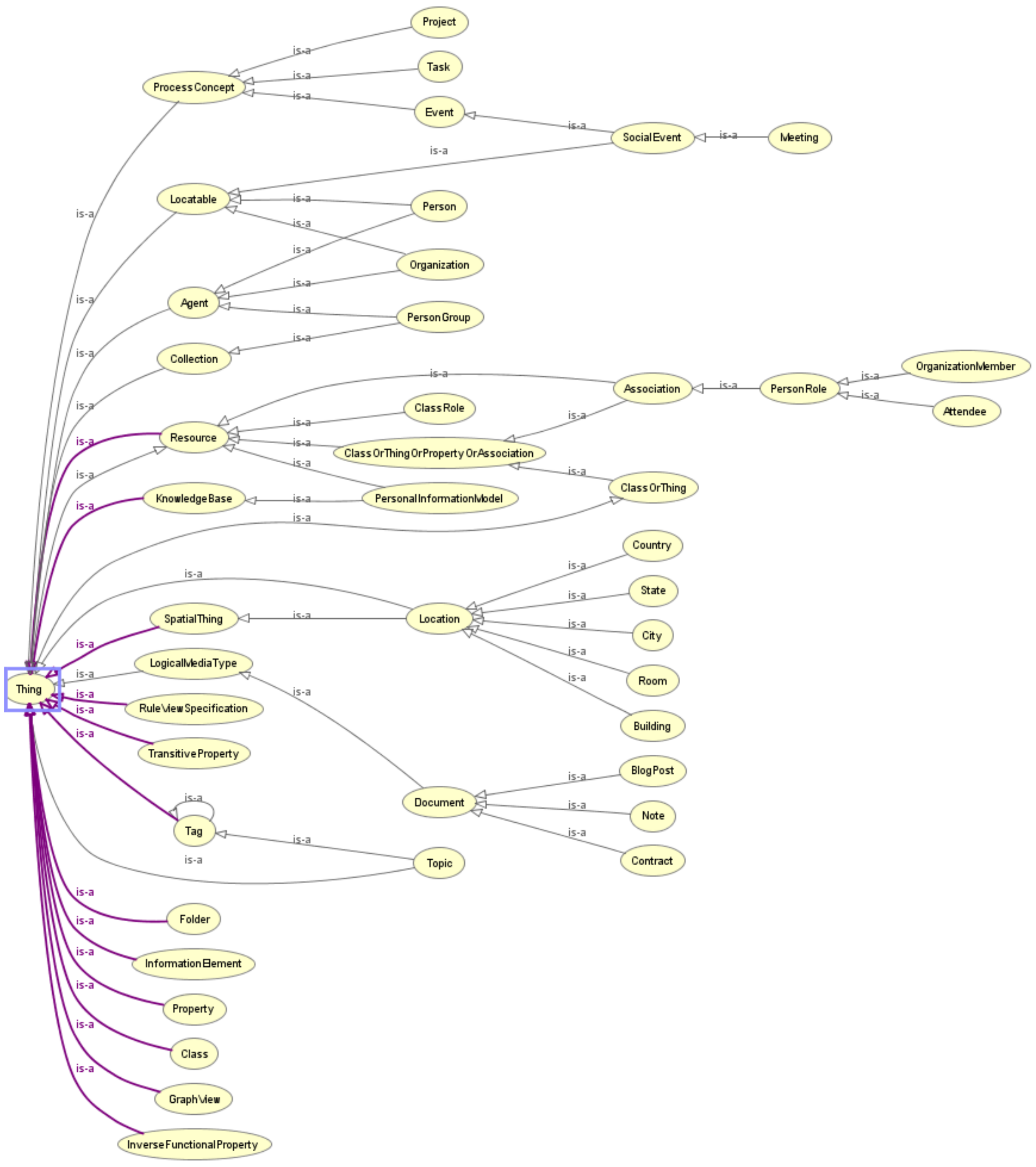


Figure 2.3: The Personal Information Management Ontology (PIMO) allows the user to model her personal information space by providing a set of "high level" concepts such as people, places, topics, documents, and time. Here the PIMO is visualized in Protégé.

part of the social semantic desktop for a user. The PIMO ontology can be seen as the realization of the so called *personal information model*:

> *"A PIMO is a Personal Information Model of one person. It is a formal representation of parts of the users Mental Model. Each concept in the Mental Model can be represented using a Thing or a subclass of this class in RDF. Native Resources found in the Personal Knowledge Workspace can be categorized, then they are occurrences of a Thing." [Sauermann* et al.*, 2007]*

For modeling user interactions, application details or fine-grained resource data and metadata, PIMO only consists of high-level concepts and is hence not appropriate for storing the fine-grained user interaction context.

2.2.2 Connection to this research...

The definition of the term *"context"* [Dey *et al.*, 2001] served as a starting point of this research of the user's context. Although context research has been performed intensively in the recent years, it still requires further work to understand how interactions influence and can contribute to the user context. Furthermore it is still unclear if there are relations/features/parts of the user context that are more significant for a user task than others. In other words, are there context features that have a higher discriminative power for distinguishing tasks than others. An example of this would be the following with the features `content` and `keyboard input`: *"Has the content of a viewed document a higher discriminative power for distinguishing tasks than the user's keyboard input?"*. The challenge of finding the most relevant features for a given task is a not yet completely solved issue. However, first well-discriminative features have been proposed by [Lokaiczyk *et al.*, 2007; Oliver *et al.*, 2006; Shen, 2009]. Well known features comprise the `window title`, the `file URL of a used resource` and the `window switching sequence`. For getting a deeper understanding, this dissertation research argues to study the discriminative power of various context features for different tasks, domains and users.

In order to allow the exploration of the feature space, a representation of the user context is required in order to show clearly what kind of information about the user context is available/stored. Ontologies are a well-formed representation which allow not only a good overview of the available contextual information but also to validate the data against the ontology specification. Further discussions of using an ontology for storing user context information can be found in [Baldauf *et al.*, 2007; Strang & Linnhoff-Popien, 2004] and in Section 3.6.

2.3 Context Observation

In 2004 a standard for modeling user attention called *Attention.Xml* [Sifry *et al.*, 2007] was proposed. This standard has been adapted and extended in the *contextual attention metadata framework* [Najjar *et al.*, 2006]. Regarding to Wolpers *et al.* [2007] this framework is able to capture information about the user's handling of digital content and to describe what a user likes, dislikes, reads, publishes, produces, watches and listens to. In EPOS [Schwarz, 2006] the captured usage data is even sent as a RDF [Herman *et al.*, 2008] graph to the context processing facility. A further way to capture context is elaborated in [Chirita *et al.*, 2006] where the authors identified desktop usage context based upon the *distance between documents*, also taking into account the documents' access time stamps. The idea behind distance is that if two files are often accessed in a small window of time the distance is smaller and hence higher related to each other. A similar approach is followed recently by Pedersen & McDonald [2008]. They related all documents displayed at the same time on the user's desktop to add a further dimension of relevancy between documents and leveraged this dimension for enhancing information retrieval.

In several recent research projects the collection of attention metadata from the user's desktop has become a key element. Once the information about the user's behavior is recorded, a wide area of possibilities for exploitation opens. Application domains are personal information management [Fernandez-Garcia *et al.*, 2006; Kleek *et al.*, 2007; Sauermann *et al.*, 2006b], task recognition and management [Kersten & Murphy, 2006; Kleek & Shrobe, 2007; Oliver *et al.*, 2006; Shen & Dietterich, 2007], technology enhanced and work-integrated learning [Lindstaedt *et al.*, 2008a, 2009b; Maier & Schmidt, 2007; Wolpers *et al.*, 2007] as well as context-aware information retrieval [Fuhr, 2005; INTELLEXT Inc., 2007; Ochoa & Duval, 2006]. A selection of projects relevant to this research effort are discussed bellow.

2.3.1 APOSDLE

APOSDLE [APOSDLE, 2006](Advanced Process-Oriented Self- Directed Learning Environment) is a work-integrated learning framework that enhances the productivity of knowledge workers by integrating learning, teaching, and working [Lindstaedt *et al.*, 2005]. The goal of APOSDLE is to enhance knowledge worker productivity by supporting informal learning activities: (i) during work task execution and tightly contextualized to the user's work context, (ii) within the work environment, and (iii) utilizing knowledge artifacts and people available within the organizational memory, consisting of textual documents, video documents and knowledgeable persons for learning [Lindstaedt *et al.*, 2008a]. In order to get to know the user better APOSDLE employs *"scruffy methods"* [Lindstaedt *et al.*, 2008a] for capturing and analyzing the user's actions.

The APOSDLE prototype features a daemon running in the background that observes various contextual features from the complete user's computer desktop [Lokaiczyk, 2008; Lokaiczyk *et al.*, 2007]. An overview of the automatic context sensing capabilities of the APOSDLE framework is given in Figure 2.4. It can be observed that a variety of sensors were implemented ranging from (i) capturing keyboard input and mouse interactions to (ii) context features specific to Microsoft Word, Microsoft Outlook and Microsoft Internet Explorer as well as (iii) operating specific sensors for tracking file system, window, process and printer information Based on the observed user context data the APOSDLE system automatically detects the current task of the user [Lokaiczyk & Goertz, 2009; Lokaiczyk *et al.*, 2007] and constructs a digital user profile [Lindstaedt *et al.*, 2009a] via inference mechanisms and heuristics. This user profile not only stores the user's usage history but also keeps the current context with respect to her personal work, learning, and collaboration related experiences of the user. This digital user profile information is then utilized to determine the user's knowledge levels as well as enabling adaptive support to the user's needs, her competences [Ley *et al.*, 2008] and her interests, such as recommendation for work and learning material [Lindstaedt *et al.*, 2008b].

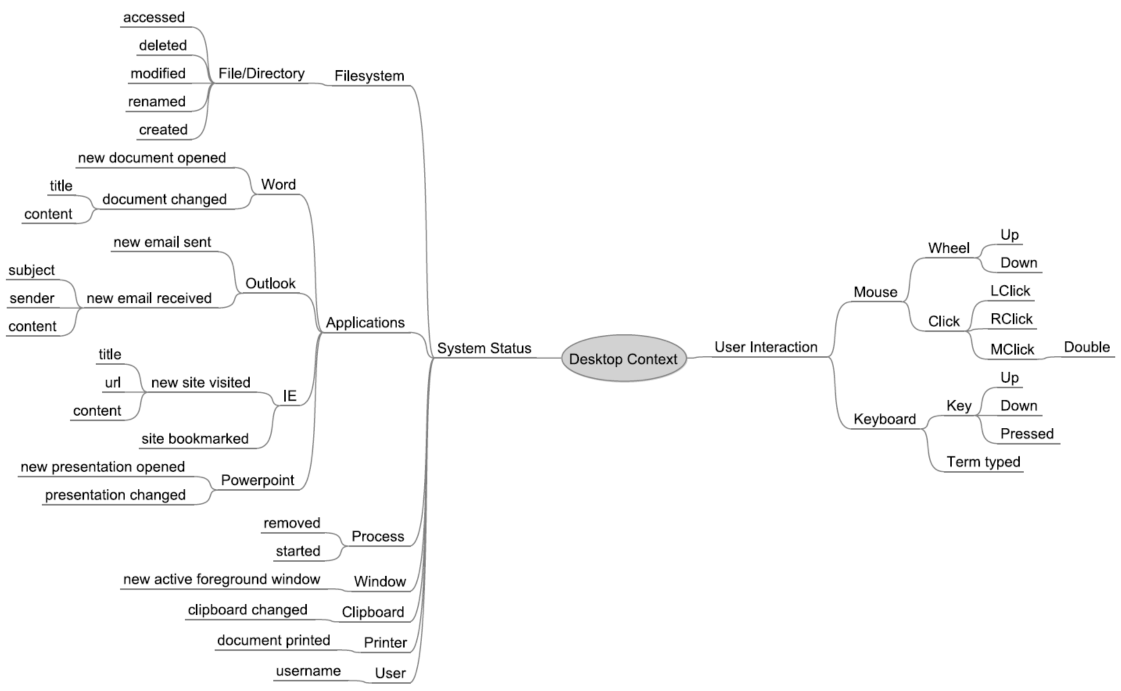


Figure 2.4: This figure taken from [Lokaiczyk et al.*, 2007] shows the observed context features in the APOSDLE system.*

2.3.2 Contextual Attention Metadata Framework

The observation of attention metadata about the user's activity is the focus of the *contextual attention metadata (CAM) framework* described in [Wolpers *et al.*, 2007]. The design goal of the CAM schema was to allow the tracking of user activities in all systems the user may interact with while working with documents. A specific XML format was introduced to observe user attention on the computer desktop referred to as *contextual attention metadata*. CAM is based on an open schema called *Attention.Xml* [Sifry *et al.*, 2007] which is a specification how to capture data about people using information in diverse applications. The Attention.Xml format was extended by CAM in order to allow the tracking of attention information about the application and context of usage, the type of task and actions the user is involved in, user information, working session, search activities and obtained results as well as user ratings and annotations for accessed documents [Najjar *et al.*, 2006].

Wolpers *et al.* [2007] motivated their approach based on the fact that information overload in learning and teaching scenarios is a hindering factor for learning. By taking into account the user's attention in the learning environment or the complete computer desktop, detailed user profiles can be build in order to provide contextual services. Statistical data about the usage of learning objects and the calculation of interest indicators for learning content were one of the simpler use cases they mentioned. More advanced ones were the identification and extraction of patterns of user behavior such as correlation of activities carried out by one user and related to the context and content of other users. User profiles can be utilized for a clustering of users/learners and the attention metadata to detect the user's goals and to identify user aims. The CAM framework

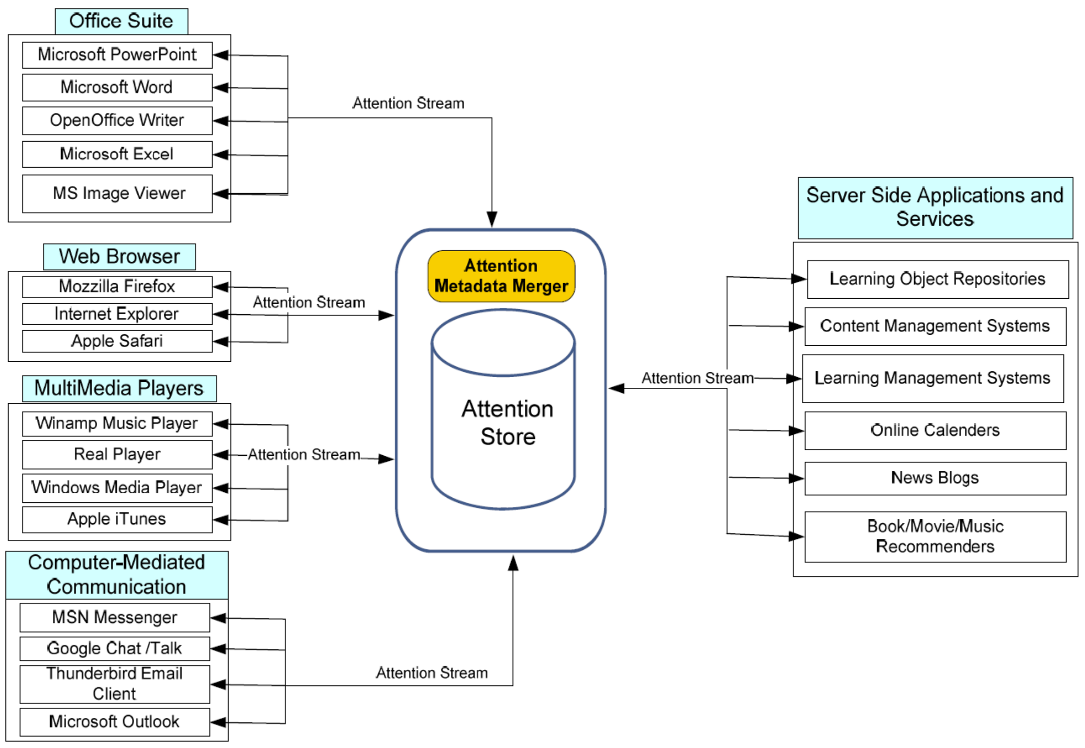


Figure 2.5: The contextual attention metadata (CAM) framework and its usage tracking capabilities. This figure was published in [Wolpers et al.*, 2007].*

can observe contextual data in several applications: WINAMP, web browsers, Microsoft Office and OpenOffice as well as in chat and email applications. A detailed overview of the framework and its sensing capabilities is given in Figure 2.5.

2.3.3 Mylar/Mylyn

Mylar [Kersten & Murphy, 2006] is an open source plug-in for the Eclipse IDE[5] that observes the user's attention on different Eclipse *views* and elements. In the background of this plug-in there is a context model that relates tasks categories, tasks and context information. Context information origins from changes in the source files and in the file system. This context information is utilized for task management, like task scheduling and planing, and user interface adaption, like e.g., only displaying files or elements of the source code the user has recently interacted during a specific task. The other parts of the source code as well as file or package listings are filtered. The context detection is only possible in the Eclipse workbench. Mylar has been renamed to *Mylyn* and has become a standard plugin in the recent Eclipse distributions.

[5]Eclipse IDE: `http://www.eclipse.org`

2.3.4 Plum

Plum [Kleek & Shrobe, 2007] stands for personal lifetime user modeling and is a research project at MIT CSAIL[6] that captures user interaction and application usage data on the MacOS X operating system for learning long-term models of user activity. It employs ontologies to organize the captured data in a versatile, reusable representation and allows continuous learning of models of activity. Plum in comparison to this dissertation research approach is similar in terms of storing the information about the user context in a semantic triple store. Plum is focused on the MacOS X operating system by utilizing AppleScript[7] for user observation.

[Kleek & Shrobe, 2007] reported the context observation possibilities of Plum on the MacOS X operating system to be the determination of window placement, the application focus, the actively running processes, the nearby WiFi access points, the keyboard/mouse idleness, the active network connections and the accessed documents within the user's home directory as well as the retrieval of the viewed content of the following applications: Adobe Reader, Apple Safari, Mozilla Firefox, Apple Mail, Preview, iTunes, iChat and Microsoft Word. According to the Plum web site they are extending their context sensing capabilities to the Microsoft Windows operating system and to Microsoft Office applications.

2.3.5 TaskTracer

The *TaskTracer* tool [Dragunov *et al.*, 2005] developed at the Oregon State University allowed the desktop user to define a set of projects and corresponding activities that characterize the user's desktop work. By continuously observing the user's interactions in various applications, digital artifacts, like files, folders, links are related to tasks and projects. Events from MS Office 2003, Microsoft Visual.Net, Internet Explorer and from the Microsoft Windows XP operating system are collected by the TaskTracer system [Shen, 2009]. They also did experiments with machine learning algorithms to automatically detect task switches. The task switch detection components are named *TaskPredictor 1* [Shen & Dietterich, 2007] and the newest one *TaskPredictor 2* [Shen *et al.*, 2009]. They reported good task switch detection performances with a combination of the `window title` and the `file url` features in *TaskPredictor 1* [Shen & Dietterich, 2007]. The differences between the TaskTracer approach and the one presented in this dissertation research are the user observation techniques used, the usage data exchange format, the type and number of the observed context features as well as the interpretation and representation the user's context. Detailed descriptions of *TaskPredictor 1* and *TaskPredictor 2* are given in Section 4.4.

2.3.6 The Semantic Logger

The *Semantic Logger* is a system for importing, housing and exploiting of personal information [Tuffield *et al.*, 2006]. Its aim is to aggregate as much information as possible about the user and its context into a central semantic web technology enabled knowledge space. Hereby they utilized a variety of sensors which can be observed in Figure 2.6. The raw sensor data is

[6]PLUM Web page: `http://plum.csail.mit.edu`
[7]AppleScript web page: `http://www.apple.com/applescript`

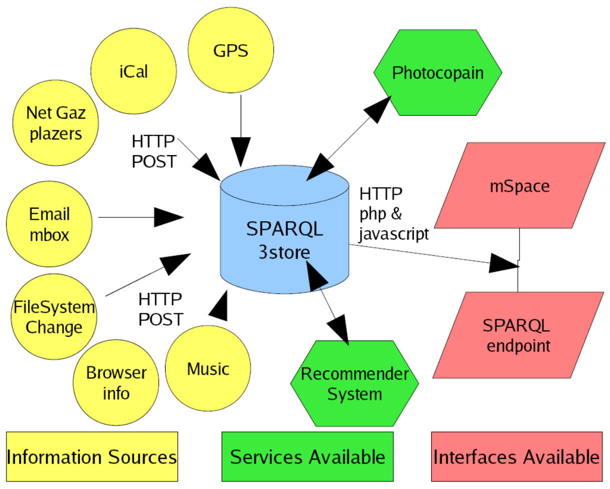


Figure 2.6: The sensing architecture of the Semantic Logger tool taken from [Tuffield et al.*, 2006].*

automatically mapped to RDF representations based on vocabularies published by the W3C. The authors argue that one of the main advantages next to the automatic context gathering is the fact that they collect all their information in a semantic store which can be queried with the semantic query language *SPARQL*. Two context-aware applications were presented based on the Semantic Logger tool, namely (i) a recommender system that utilizes contextual information in order to improve the accuracy of the recommendations and (ii) a photo-annotation tool that automatically extends the available metadata with context and community based knowledge [Tuffield *et al.*, 2006].

2.3.7 Connection to this research. . .

Automatic context observation mechanisms developed in the mentioned projects above and the corresponding lessons learned from these projects served as a valuable starting point for the design and the development of the context observation mechanisms for this research effort. Especially helpful were the different ways shown for capturing usage data from the computer desktop. During this dissertation research new context sensors had to be developed from scratch because only sensors for Mozilla Thunderbird[8], Mozilla Firefox[9] and Microsoft Outlook[10] were freely available. Examples of context observation mechanisms are plug-ins, macros, extensions, application and operating system hooks via the component object model (COM) interface of Microsoft Windows or sensor applications themselves.

[8]Mozilla Thunderbird 1.5 extension: `http://dragontalk.opendfki.de/wiki/Thunderbird_userobs`
[9]Mozilla Firefox 1 and 2 extension: `http://dragontalk.opendfki.de/wiki/Firefox_userobs`
[10]Outlook 2003 and 2007 extension: `http://sourceforge.net/projects/activity-logger/`

Table 2.1 gives a global summary of the presented context observation approaches in this section in respect to the following criteria: (i) *context sensing*, (ii) *context model*, and (iii) *context storage*.

Description of the criteria:

(i) **Context sensing** describes the capabilities of sensing the context in applications and from the operating system.

(ii) **Context model** distinguishes the representation of the context according to Strang & Linnhoff-Popien [2004]'s categorization: key-value models, markup scheme models, graphical models, object oriented models, logic-based models and ontology-based models.

(iii) **Context storage** [Truong & Dustdar, 2009] is about the following aspects: the storage location of the context (*storage model*), the database used for storing the context (*storage database*), the interface available to access the stored context (*access interface*), and the query language used for querying the context (*request specification*). The storage model distinguishes between central and distributed storage of the context. The storage databases can be relational databases (rel), semantic stores (sem), or XML based stores (xml). The access interfaces are web services (ws) and others. A SPARQL endpoint is also considered as a web service. The request specification is categorized in SQL, XQuery/XPath, SPARQL and others.

The goal of the context sensors of this research was to keep them as simple as possible and to shift the complexity of the inferring algorithms to a central unit. The XML format suggested by the *CAM* approach seemed to be too sophisticated for this purpose. The storage of personal information in a semantic store suggested by the Semantic Logger and Plum approach including the resulting advantages regarding querying and inferencing possibilities influenced the decision to use semantic technologies for user interaction context observation, representation and storage. Since good task detection performances were reported for the `window title` feature in *SWISH* and for a combination of the `window title` and the `file URL` features in *Task Predictor 1* context sensors for these two features were implemented as well.

Criteria	**Sub Criteria**	**APO**	**CAM**	**MYL**	**PLUM**	**TT**	**SL**
Context Sensing	*Application:*						
	Office Suite[12]	x	x		x	x	
	Multimedia[13]		x		x		x
	Instant Messaging[14]		x			x	
	Web Browser	x	x		x	x	x
	Email	x	x		x	x	x
	Development Environment			x		x	
	Others		x				x
	Operating System:						
	Mouse Input	x		x	x		
	Keyboard Input	x		x	x		
	Application Details[15]	x	x	x	x	x	
	Printer	x					
	File System	x	x		x	x	x
	Clipboard	x			x		
Context Model	Graphical						
	Key-Value					x	
	Logic-based						
	Markup Scheme		x	x			
	Object Oriented						
	Ontology	x			x		x
Context Storage	Storage Model[16]	C	C	C	C	C	C
	Storage Database[17]	rel	xml	xml	sem	?	sem
	Access Interface	ws	?	?		ws	ws
	Request Specification	SQL	XQuery	?	SPARQL	?	SPARQL

Table 2.1: This table shows a summary of the context observation approach presented in this section: APOSDLE (APO), Contextual Attention Metadata Framework (CAM), Mylar/Mylyn (MYL), Personal Lifetime User Modeling (PLUM), TaskTracer (TT) and Semantic Logger (SL). The question mark "?" indicates that no information about this criterion was available. The "x" and the " " signal that the criterion is met or not respectively.

[12] Office applications: text editors, spreadsheet applications, presentation applications, etc.
[13] Multimedia applications: Winamp, Windows Media Player, graphic editor, photo viewer etc.
[14] Instant messaging applications: Skype, ICQ, MSN Messenger, etc.
[15] Application details: window title, process name, application name, file/web page URL etc.
[16] Storage model: C. . . centralized storage, D. . . distributed storage.
[17] Storage database: relational database (rel), semantic store (sem), XML (xml)

2.4 Context Utilization and Exploitation

The utilization of semantic technologies for context modeling and context-aware applications has become quite popular recently. However, many of the projects undertaken between Dey's evaluation [Dey, 2000] in 2000 and now, define their own data capturing schema and implement the context capturing mechanisms for their purpose by their own again. In SWISH [Oliver *et al.*, 2006], a Microsoft research project, used only the `window title` to determine the tasks of a user. Further context features in the Microsoft Windows operating system environment were captured in the client-side monitoring framework introduced by Fenstermacher & Ginsburg [2002] and in the TaskTracer [Dragunov *et al.*, 2005] project. They observed file system and phone usage, web page navigation, text selection and various metadata from Microsoft Office products. Some of the captured context features were used for task switch detection and unsupervised task learning [Shen & Dietterich, 2007; Shen *et al.*, 2007, 2009].

Other approaches focusing on directly capturing the user context are Lumiere [Horvitz *et al.*, 1998], GNOWSIS [Sauermann, 2003], APOSDLE [Lindstaedt *et al.*, 2008a], EPOS [Dengel *et al.*, 2002] and its follow up project Mymory [Elst, 2006], Dyonipos [Tochtermann *et al.*, 2006], Plum [Kleek & Shrobe, 2007], Jourknow [Bernstein *et al.*, 2008], Watson [Budzik *et al.*, 2001; INTELLEXT Inc., 2007], Java Context Awareness Framework [Bardram, 2005] and Learning in Process [Schmidt, 2005a]. Mylyn [Kersten & Murphy, 2006], an Eclipse plug-in project, observes the attention of the programmer in Eclipse and adapts the user interface based on the observations. EPOS, Plum, Learning in Process, APOSDLE and Dyonipos utilize RDF to represent the user's context. Indirect approaches utilize log files of various applications as a representation of the performed work [Aalst *et al.*, 2005; Fenstermacher, 2005; Maruster *et al.*, 2002]. For analyzing log files of different granularity levels the open-source ProM Framework [ProM Framework, 2007] has become a standard workbench for process mining. It includes 150 task and process mining algorithms. The approaches mentioned above differ in the granularity of the captured usage data, whereas the direct approaches gather more fine-granular data about the users' activities than available in log files of applications and systems. The goals of the mentioned projects and the intentions to exploit contextual features overlap. Although the main areas of context exploitation identified in the mentioned projects are task and process detection and mining, context-based information retrieval, task as well as process modeling and optimization, work and learn support.

2.4.1 Connection to this research...

There are several projects going on in the area of context and context-awareness at the moment. The high number of projects shows that context is a hot topic in research. Interesting is that there are multiple areas in which context plays a crucial role in achieving the project's targets, such as in the work-integrated learning area, task and process management, information retrieval and personal information management. By knowing the approaches, goals and problems of the above mentioned projects this research built on their results and derived requirements and further directions. Especially this research benefited from the results achieved in the task mining area by [Fenstermacher, 2005] who used event logs from workflow management systems, [Oliver *et al.*, 2006] who utilized the window titles and the window switching sequence for task detection and [Shen *et al.*, 2007] who performed task switch detection based on *window title*, the *file pathname* and the *url of the web page*.

In comparison, this dissertation research studied novel context features/feature combinations engineered from the proposed ontology-based user interaction context model which are for example the interconnectivity of resources, user interaction patters, ontology structural features as well as new text based context features. In this research the discriminative power of these novel features/feature combinations as well as the one of features/feature combinations of existing approaches for distinguishing tasks are investigated (see Chapter 5 and 7).

2.5 Application Areas of Context

In the following sections an overview of projects and approaches involving user context information in the areas of (i) *task- and process mining and management*, (ii) *work-integrated learning*, (iii) *semantic desktop and personal information management*, and (iv) *information retrieval* are presented.

2.5.1 Task- and Process Mining and Management

The common process modeling approach, where processes are modeled manually based on the available process data or information, is called the top-down approach. Data and information about executed tasks and processes, involved persons and resources are usually obtained from interviews, existing workflow management systems (WFMS), observations during site visits, document inspection, or (if available) previous process descriptions. The various information sources and the retrieved data need to be structured and aligned manually by the process engineer [Rath *et al.*, 2006]. WFMSs have become quite popular for managing complex organizational processes, but fail in supporting knowledge-intensive and agile processes [Schwarz *et al.*, 2001]. The problem with this kind of process is that they cannot be modeled in advance. The context of a user can also not be modeled in advanced which means that it is not straight forward to match the observed user context with a pre-modeled one and hence difficult to detect the situation or the task in which a user currently is.

Further limitations of WFMSs are their minor ability to deal with dynamic changes [Aalst *et al.*, 2005] because of the implemented static process models. Weakly-structured workflows address this insufficiency by suggesting lazy and late modeling or interleaving process modeling with process execution [Elst *et al.*, 2003]. Detection of process changes is limited in standard WFMS, because refinement and deviations of standard workflows are usually not allowed and hence no workflow logs about the deviation exist.

The contrasting approach to process modeling is the bottom-up approach, which means that the information originates from process executors instead of process engineers [Riss *et al.*, 2005]. The bottom-up approach is also referred to as process mining [Aalst & Weijters, 2004; Fenstermacher, 2005; Wen *et al.*, 2008]. In this approach, the process model can be derived from workflow, task, and/or event logs. The instantiation of a workflow, a process or a task as well as the data collected about deviations are parts of the organizational context which is according to [Schwarz, 2006] an aspect of the user context. In order to transform the monitored data stored in the logs into tasks and processes advanced algorithms [Aalst *et al.*, 2004] are needed. The advantages of this bottom-up approach are the intensive data and information gathering possibilities as well as the continuous refinement and enhancement of the calculated processes as the number of cases, i.e., process executions, increases.

Event log mining [Fenstermacher, 2005] has the advantage of providing fine-grained data to the mining step in comparison to [Aalst *et al.*, 2004] where tasks from workflow logs are used as a basis. Event logs incorporate data about the executions of standard and ad-hoc processes and hence event log mining considers both types of processes when calculating the process model. Fine-grained event logs are also used for automatically detecting tasks of users.

In the area of task detection, Shen & Dietterich [2007] used a classifier to predict the current task based on features, like for example, `window title` and the `file pathname`, extracted from the window in focus. In [Oliver *et al.*, 2006] task assignment is based on relations of the windows on the user's desktop. Lokaiczyk *et al.* [2007] evaluated five algorithms for task detection based on observed context features and report achieved an accuracy of over 85% whereas the support vector machine (SVM) approach with the sequential minimal optimization algorithm performs best. In the area of task-centered information management Catarci *et al.* [2007] developed a top-down approach to task inference in which users define the main aspects of tasks using forms of declarative scripting. They developed a task specification language and proposed an architecture for supporting task inference.

2.5.1.1 Connection to this research...

From the process management area can be learned that there are some processes that cannot be modeled in advance, such as ad-hoc processes or weakly structured workflows. The user interaction context is quite similar to these ad-hoc processes or weakly structured workflows. Similar in such a way that both the user interaction context and processes and workflows are created and evolute while the user performs her actions, i.e., interacts with resources and applications. The recording of the user actions in a workflow management system for process and task mining can be compared to capturing the user interactions on the user's desktop while performing a task. The resulting event logs of the observed actions of both attempts can be utilized for mining or detecting tasks.

Event logs/streams are also exploited for task recognition [Granitzer *et al.*, 2008; Lokaiczyk *et al.*, 2007; Oliver *et al.*, 2006] and task switch detection [Shen, 2009; Shen & Dietterich, 2007]. The research results that have been achieved in these two areas about utilizing machine learning algorithms for task recognition were a starting point for this research. Further investigations have still to be carried out for understanding the influence of automatically observed context features for the accuracy of task detection. Since there are no standard task detection datasets available, experiments have to be designed and performed for collecting the task usage data from various domains, from different users and for different tasks. This dissertation effort targeted this issue and collected datasets from three large-scale laboratory experiments. These datasets were used to study influencing factors for task detection in order to enhance the accuracy of the task detection algorithms.

2.5.2 Work-Integrated Learning

Learning activities can also happen within work processes [Eraut, 2004]. *Work-integrated learning (WIL)* [Lindstaedt *et al.*, 2008a,b, 2009b; Smith, 2003] takes up on this and sees learning as a dimension of work. The goal of WIL is to foster learning at the workplace during work in order to enhance task performance. It focuses on the learning perspective of the individual as opposed to the training perspective of an organization [Lindstaedt *et al.*, 2008a]. Hereby it greatly differs from more traditional learning approaches like seminars, formal courses or e-learning approaches in such a way that WIL happens spontaneously and often unintentionally within work processes. For assisting the learner in this kind of learning situation her *user profile* [Fischer, 2001; Ulbrich *et al.*, 2006], her interests [Goecks & Shavlik, 2000] or her competencies [Ley *et al.*, 2008] as well as her *user context* [Lindstaedt *et al.*, 2008a; Schmidt, 2007; Wolpers *et al.*, 2007] are utilized to improve the quality of support mechanisms. Automatic task detection has also been explored for improving work-integrated learning by Lokaiczyk *et al.* [2007].

2.5.2.1 Connection to this research...

The user context plays an important role in work-integrated learning because learning happens spontaneously within work tasks. Hence it is important o understand the user's current situation which is described by her user context. The user context not only describes the user's current situation but also includes her current task. Through automatic user context detection and task detection mechanisms a situation and task specific learn support is possible. Examples include retrieval of learning objects [Duval & Hodgins, 2003] or suggestions of course material, links, documents or topical experts [Lindstaedt *et al.*, 2008a]. WIL will benefit from the outcomes of this thesis in terms of a more accurate computation of user profiles, user competencies and user interests as well as up-to-date task and user context information for enriching learn support.

2.5.3 Semantic Desktop & Personal Information Management

The hype of the *Semantic Web* has also reached the user's computer desktop. Projects like GNOWSIS [Fernandez-Garcia *et al.*, 2006; Sauermann *et al.*, 2006b], EPOS [Dengel *et al.*, 2002] and NEPOMUK [Groza *et al.*, 2007] have been started to combine the power of semantic web technologies and the user's computer desktop environment. One of the main goals of this project was to identify every item on the user's computer desktop with an unified resource identifier (URI) and represent it as a semantic object with metadata relations. Many new ways of doing search, supporting personal information management [Sauermann *et al.*, 2006a] and organize work [Riss & Grebner, 2006] were built upon the semantic desktop. In 2007 Braun *et al.* [2007] evaluated four semantic desktop systems for context awareness and discussed requirements and architectural implications for such kind of systems. They concluded that such systems have to take into account temporality, imperfection and integration issues as well as invest into scalability and flexibility of information integration.

Personal information management (PIM), first introduced in the 1980s by Lansdale [1988], deals with how people process and manage information. PIM is about finding, keeping, organizing, maintaining and evaluating information and making sense of the information [Jones, 2007]. Several PIM tools have been created to study and improve the human-computer interaction [Boardman & Sasse, 2004; Teevan *et al.*, 2008]. These PIM tools combine email management with time, task, and contact management [Jones, 2007], support the finding and re-finding [Dumais *et al.*, 2003] as well as note taking [Bernstein *et al.*, 2008]. In the combination of these approaches there is still potential to take into account the context of the users, which means that they would benefit in including information about the situation of the users when doing PIM activities and about how they do it. This is why this *"activity dimension"* requires further research. PIM sometimes fails because there is too much effort for the user to add the relations between PIM objects [Sauermann *et al.*, 2007]. A possibility to reduce this additionally introduced user workload would be automated or semi-automated relation creation mechanisms.

2.5.3.1 Connection to this research...

The connection between semantic desktop systems and this research is the usage of unified resource identifiers (URI) to represent every resource on the computer desktop the user interacted with. Common is also the utilization of semantic technologies for storing data and metadata about resources and their relations in an ontology. The advantage of an ontology-based user interaction context model is that it would allow a seamless and straight forward integration of contextual information into semantic desktop systems. By keeping in mind that the user context relations are automatically computed from observed user interactions, this would contribute a new dimension, an so called *action dimension* to a semantic desktop system and to PIM tools. The user interaction context model proposed in this dissertation research (see Section 3.3) is able to handle dynamic change and temporality, which were demanded by Braun *et al.* [2007].

2.5.4 Information Retrieval

Context-aware information retrieval [Fuhr, 2005] differs from classical information retrieval [Rijsbergen, 1979; Salton & McGill, 1986] in such a way that the user context is included in the information retrieval task. In the SIGIR workshop report [Allan *et al.*, 2003] contextual retrieval is defined as *"combining search technologies and knowledge about query and user context into a single framework in order to provide the most appropriate answer for a user's information needs"*.

In a recent ACM SIGIR Forum meeting an information retrieval research agenda was elaborated. It lists three important elements regarding context [Callan *et al.*, 2007]: (i) *understanding the user* who is asking questions (search), (ii) the *underlying information domain* which represents the relationships between documents as well as other rich entities within them, and (iii) the *larger task* the user tries to accomplish. They concluded that the better the context of the search (user, domain of interest, larger task) is understood, the better *the right information can be delivered to the right people at the right time.*

Context-aware information retrieval components become more and more popular to be included in tools developed in today's research projects. These tools observe the user on the desktop, on the web or in mobile environments. Several do not exploit the relationships among the contextual elements they observe [Budzik *et al.*, 2001; Dragunov *et al.*, 2005; INTELLEXT Inc., 2007; Oliver *et al.*, 2006] and hence not utilizing the complete strength of context. Some of them though keep the relations they discover but hardly use them in the information retrieval task [Belizki *et al.*, 2006; Kröll *et al.*, 2006; Lindstaedt *et al.*, 2008a; Schmidt, 2005a]. One big challenge in the area of context-aware information retrieval is the evaluation. There are no standardized datasets with usage data available for comparing such systems [Callan *et al.*, 2007]. The privacy protection issues when dealing with contextual data of users *"in the wild"* unveil further difficulty. A possible solution for making usage data anonymous was proposed by [Chernov *et al.*, 2008].

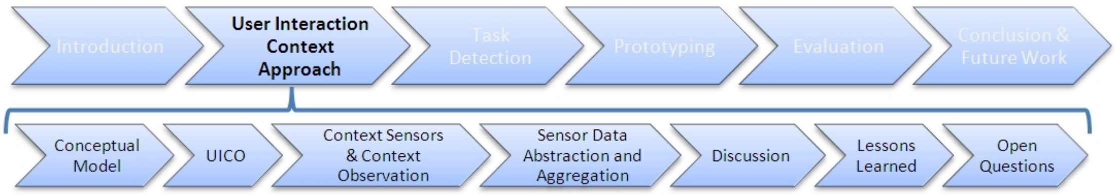
2.5.4.1 Connection to this research...

In context-aware information retrieval systems some features of the user context is used to trigger or enhance the retrieval process. The selection which features to use in a current situation to construct a query is kind of similar to selecting the most significant features for a current situation of a user when performing a task. The *Watson* system [INTELLEXT Inc., 2007] uses the content of the document that is currently in focus for query into online information sources, whereas in the Dyonipos system [Kröll *et al.*, 2006; Rath *et al.*, 2008] and in the application described in [Belizki *et al.*, 2006] various aspects of the user context, for example, the window title, the document name or the subject of an email is used to enhance proactive information delivery and full-text desktop search respectively.

2.6 Summary

This chapter gave an overview of the related literature in the area of user context detection. Hereby approaches to user context modeling and user context observation were presented and discussed. Context utilization and exploitation work as well as application areas of context including (i) *task- and process mining and management*, (ii) *work-integrated learning*, (iii) *semantic desktop and personal information management*, and (iv) *information retrieval* were highlighted.

3 User Interaction Context Approach



This chapter introduces the *user interaction context detection approach* proposed by this dissertation research. In Section 3.2 the term *user interaction context* as a subset of Dey's definition of user context [Dey *et al.*, 2001] is presented. A conceptual model for this user interaction context, referred to as the *Semantic Pyramid*, and the realization of this conceptual model as a user interaction context ontology (UICO) are introduced and explained in Section 3.3.

Furthermore this chapter elaborates on the three key steps of the *automatic ontology-based user interaction context detection pipeline* which are (i) the automatic observation of the user interaction context, (ii) the automatic abstraction and aggregation of the raw sensor data and (iii) the automatic population of the user interaction context ontology (UICO). Section 3.4 describes the capabilities of the sensors that were implemented for observing the user interactions with resources, applications and the operating system. It also gives an overview of the data and metadata that can be sensed. The abstraction and aggregation mechanisms for transforming the raw sensor data to the *action* and *resource* pair constituting user interactions with resources is elaborated in Section 3.5. Section 3.5 also explains the automatic instantiation of concepts of the ontology as well as the computation and creation of the relations between the concept instances.

The complete user interaction context detection pipeline starting from the sensors that capture the user interaction context via the automatic model population mechanisms including the construction of an event from low-level sensor data, resource discovery, resources building as well as event to event block aggregation to the populated user interaction context model is illuminated in Figure 3.1. A discussion about the advantages and disadvantages of following an ontology-based user context modeling approach from a practitioner's perspective in Section 3.6 rounds off this chapter.

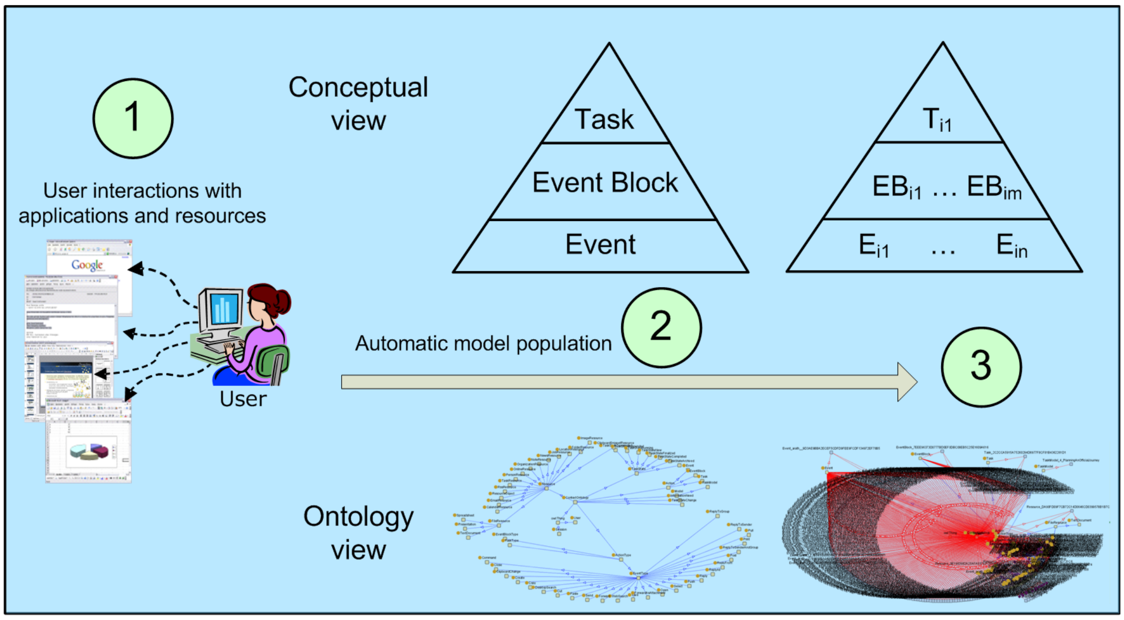


Figure 3.1: This figure visualizes the user interaction context detection pipeline starting from the context sensors (1) that capture the user interaction context to (3) the populated user interaction context model. The automatic model population mechanism (2) includes the construction of an event from low-level sensor data, resource discovery, resources building as well as event to event block aggregation.

3.1 Introduction

For a user it is not convenient to manually enter the data about her user context on a fine-granular level as defined in the user interaction context ontology (UICO). Hence semi-automatic and automatic mechanisms are required to ease the process of "populating" the ontology. Rule-based, information extraction and machine learning approaches are utilized to automatically populate the ontology and to automatically derive relations between the model's entities. The user interaction context is observed, instances of concepts are created and relations between the concept instances are augmented. The following sections elaborate on what kind of sensors are used to observe user interactions, how resources the user has utilized are automatically discovered and built as concept instances of the `Resource` concept, connections between resources are computed, and the aggregation of single user interactions (events) to user interactions on the same resource (event blocks) and tasks.

3.2 Conceptual Model - The Semantic Pyramid

The *user interaction context* is defined as

> " *all interactions of the user with resources, applications and the operating system on the computer desktop. Resources are digital artifacts on the computer desktop, e.g., documents, web pages, emails, persons, appointments and notes."*

This user interaction context is a subset of Dey *et al.* [2001]'s definition of context which is defined as

> *"any information that can be used to characterize the situation of entities that are considered relevant to the interaction between a user and an application, including the user and the application themselves".*

The user interaction context can be seen as a pyramid, the *semantic pyramid* [Rath *et al.*, 2006] from a user action point of view which connects the user's actions with resources and aggregate them. It also integrates the idea of delivering resources that are relevant to the user's actions based on information needs of the user. The semantic pyramid describes the continuous evolution of contextual information through different *semantic layers* and is illuminated in Figure 3.2.

Starting at the bottom with events that result from user interactions with the computer desktop by one user and ending with tasks. The semantic pyramid is from a users' perspective who utilize the computer desktop and its digital resources and applications to fulfill their tasks. The layers of the semantic pyramid represent the aggregation levels of the action dimension from the event level to the task level of a single user. The levels of the semantic pyramid can be compared to the hierarchical structure of an activity (operation, action, and task) like defined in activity theory [Leont'ev, 1978]. The operation level would correspond to the event block level and the action level to the task level. The activity level could be seen as a more coarse granular task (e.g., leading a project) since the boundaries between the activity and action level are not clearly distinguishable The event level is too fine-granular and hence has no correspondent in activity theory but can be dedicated to the operation level.

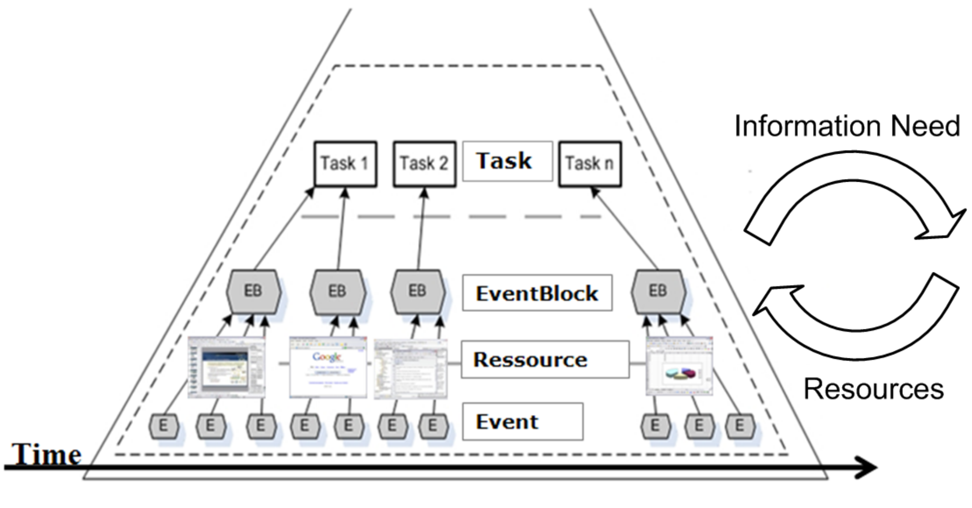


Figure 3.2: The semantic pyramid [Rath et al.*, 2008] comprises the event, the event block and the task layer. Information needs can origin from each layer and are fulfilled by resources.*

User interactions with the computer desktop represent **events**. Events form atomic units within an event block. Events can be for example user keyboard inputs, such as mouse movements, mouse clicks, starting a program, creating a folder, a web search, or opening a file. An event is an action on a single resource. At the moment only actions on a single resource are considered, but this concept could be extended to actions on multiple resources as well. An example for this would be if the user selected multiple files on the computer desktop or communicating with multiple users in an instant messaging application.

An **event block** is defined by a sum of events that act on the same resource. These events are totally ordered. An example of an event block is *"editing a document on page 2"*. Event blocks are formed using predefined static rules, which map a set of events originating from the user interactions with a single resource to an event block. Event blocks are combined into tasks by grouping together similar event blocks into semantic sets. Thus one resulting set represents one task.

A **task** is a well-defined step in a process, which can not be divided into sub tasks and in which only one person is involved. Collaborative tasks are not included at the task level. This means that collaborative tasks are tasks from various persons that run parallel in a process.

Information needs are needs that emerge from an active information request or from a change of the user interaction context. In the first situation the user directly requests information from an information source, for example, an external or local database, an application's data storage, or the web. The detection of information needs is based on the detection of a change in the user interaction context. The investigation of information needs is not in focus of this research but is mentioned for completing the picture for the reader.

Resources are digital artifacts on the user's computer desktop. All resources can be identified with a uniform resource identifier (URI). Examples of resources are documents, web pages, emails, persons, appointments, notes, and so on. A resource is not bound to a specific application, for example a text document can be modified in Microsoft Word, in the Windows Notepad or in the Vi editor. A further example is an email that can be read on the web or in diverse email applications, like Microsoft Outlook or Mozilla Thunderbird. For resources that do not have an URI-based representation on the user's computer desktop, e.g., clipboard content, an artificial URI is computed. Resources are also used to fulfill the user's information needs.

3.3 User Interaction Context Ontology (UICO)

The *user interaction context ontology* (UICO) can be seen as the realization of the semantic pyramid with the support of semantic technologies. The UICO is a fine-grained ontology, driven by the goal of representing automatically captured low-level user interaction information. The UICO follows a bottom-up approach. It is built on the basis of the semantic pyramid. New relations are incrementally added as new context sensor data or user interaction context analysis algorithms are added. The UICO reflects the sensed information about the user interaction context and relates the information automatically derived from it. User interaction context information here means the concepts and the relations between concepts of the semantic pyramid as well as the

resource data and meta data captured by the context sensors. A bottom-up approach for adding concepts and relations to the UICO has been chosen in order to be continuously synchronous with the capabilities of the context sensors (see Section 3.4) as well as the user interaction context analysis algorithms (see Section 3.5).
At the moment the UICO contains 107 concepts (classes) and 281 properties modeled. From these 281 properties there are 224 datatype properties and 57 objecttype properties. The ontology is in OWL-DL [OWL, 2007]. OWL was used because (i) it is a W3C standard for modeling ontologies, (ii) it is widely used in the semantic web community and (iii) tools for supporting the ontology modeling process are available. The Protégé ontology modeling tool [Protégé, 2009] was used for modeling the UICO. A visualization of the concept hierarchy (sub-class relation) in Protégé is given in Figure 3.3. A comparison with related ontologies is elaborated in Section 3.3.6.

Five different dimensions can be identified in the UICO: (i) *action dimension*, (ii) *resource dimension*, (iii) *information need dimension*, (iv) *application dimension* and (v) *user dimension*. These five dimension of the UICO are described in further detail in the next sections.

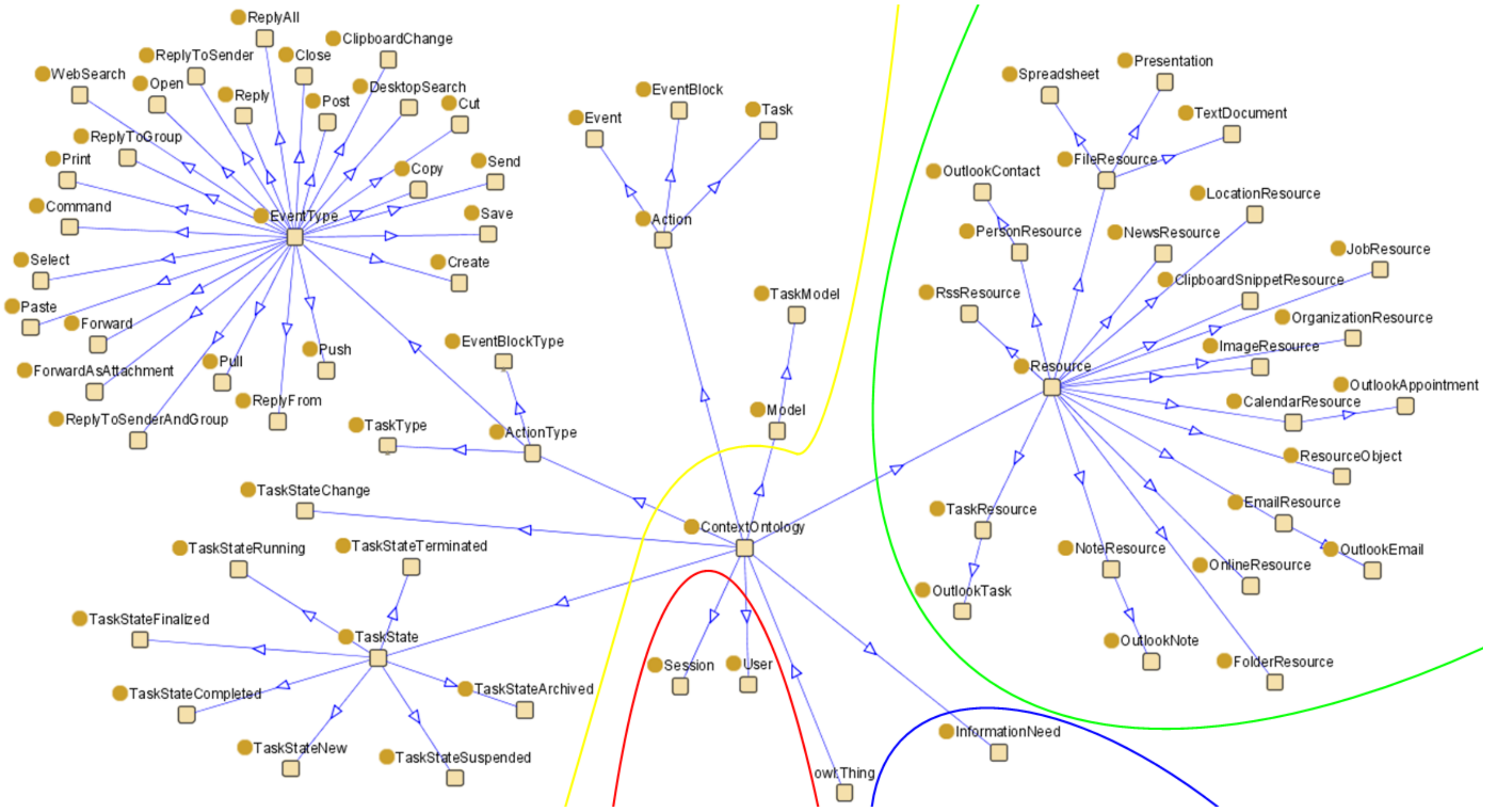


Figure 3.3: The concepts of the user interaction context ontology (UICO) visualized in the Protégé tool. In the left area this figure shows the action dimension, in the right area the resource dimension, in the bottom left area the user dimension and the information need dimension on the bottom right area. The application dimension has no concepts and hence not visible here.

3.3.1 Action Dimension

The action dimension consists of concepts representing user actions, task states and connection points to higher level concepts of an upper ontology. User actions are present based on the granularity, i.e., `Event` at the lowest level, then `EventBlock` and then `Task`. These action concepts corresponds to the levels of the semantic pyramid. The `ActionType` concepts specify which types of actions are distinguished on each granularity level. Currently only types of actions on the event level (`EventType` concept) are distinguished but the elaboration of `EventBlockTypes` and `TaskTypes` are in progress. There are 25 different `EventTypes`, each one representing a single type of user interaction. An example is the following: if the user clicks on the search button of a search engine's web page in a web browser, this user interaction will generate an Event of type `WebSearch`. The various types of events are described in more detail in Table 3.3.1.

EventType	**Description**
ClipboardChange	Indicates that the clipboard application has received new content, e.g., user pressed the key combination `Control+C`
Close	The user has closed an application, e.g., the user clicks on the top right corner of the application window such that the application closes.
Command	This is a special `EventType` that is used to remote control the user interaction context observation process, e.g., "create a new task", "stop context observation".
Copy	Indicates that the user has copied some content from a resource.
Create	Indicates that the user has created a resource, e.g., create a new email.
Cut	Indicates that the user has cut content from a resource, e.g., cut a piece of text from a text document.
DesktopSearch	Indicates that the user has performed a desktop search, e.g., Google Desktop Search, MSN Desktop Search.
Format	A style modification of text is indicated by this concept and its 14 sub-concepts, e.g., a user formats a text to a bold text.
ForwardAs Attachment	The user forwards a resource as an attachment, e.g., forwarding an email as an attachment.
Open	The user opens a resource, e.g., opening an appointment or an address book entry.
Paste	The user pastes the content of the clipboard to a resource.

Continued on next page

Table 3.1: This table lists the `EventType` concepts of the user interaction context ontology and gives a short descriptions when an event is related to an instance of a `EventType` concepts.

EventType	**Description**
Post	Indicates that the user posts a resource, e.g.,posting a message to a newsgroup.
Print	Indicates that the user prints a resource, e.g., printing of a text document, source code of a program, a picture, or a presentation slide.
Pull	The `Pull EventType` indicates that the user consumes the content of the resource that is present in an application. This `EventType` is assigned to the `Event` if navigational and command inputs are executed by the user.
Push	The `Push EventType` indicates that the user produces content in a resource that is present in an application. This `EventType` is assigned to the Event if non navigational and non command keys are executed by the user.
Reply, ReplyToSender, ReplyToSenderAnd-Group, Reply-ToGroup, ReplyFrom	The user replies to one or more `PersonResource` concept instances, e.g., reply to the sender of an email, task assignment or appointment invitation.
ReplyAll	The user replies to all `PersonResource` concept instances, e.g., reply to the sender of an email, task assignment or appointment invitation.
Save	The user saves a resource, e.g., saving the contact details of a person.
Select	The users selects a resource for viewing in the current application or selects some text of a resource.
Send	Indicates that the user sends a resource, e.g., email, appointment invitation or contact.
WebSearch	The users executes a search on a search engines web page, e.g., Google, MSN Live or Yahoo!.

Table 3.1: This table lists the `EventType` concepts of the user interaction context ontology and gives a short descriptions when an `Event` instance is related to an instance of a `EventType` concept.

The `TaskState` concept and its sub concepts (visualized in the upper right corner of Figure 3.3) are used to model the ways the user does task management and task executions. The types of task states are derived from the *NEPOMUK Task Management Model* [Grebner *et al.*, 2007] and integrated into the UICO. Figure 3.4 shows the task states and its transitions. With the help of task states UICO can model the user's task handling, i.e., creating, executing, interrupting, finishing, aborting, approving and archiving a task. The behavioral patterns of the user's task handling and task state changes are tracked via the `TaskStateChange` concept.

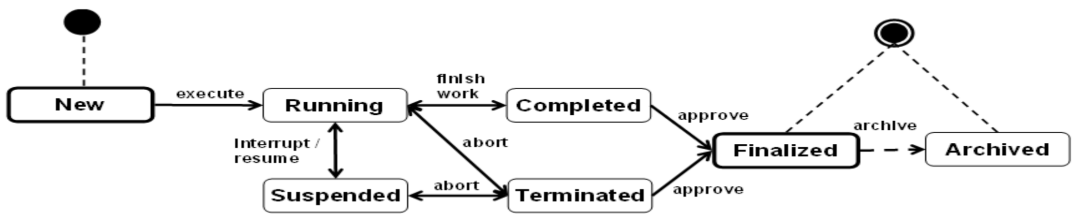


Figure 3.4: This figure displays the NEPOMUK's Task Management Model from [Grebner et al.*, 2007].*

The `Model` concept has been introduced to have connection points to high level concepts of other ontologies, e.g., semantic desktop ontologies. Currently only one connection point in form of the `TaskModel`, a sub class of the `Model` concept, is present. The `TaskModel` concept is similar to those defined in the area of workflow management systems or task process analysis.

At the moment, the `TaskModel` concept can be seen as a way of categorizing a task. An example of instances of the `TaskModel` and the `Task` concept is “Planning a journey” and “Planning the journey to CIAO 2009 workshop” respectively.

3.3.2 Resource Dimension

The resource dimension, visualized in the upper left corner of Figure 3.3 contains concepts for representing resources on the computer desktop. Specifically the focus is on modeling resources used by office workers. These were identified by informal interviews. The UICO can easily be extended by further types of resources by adding new concepts and relations. In the UICO there are 28 different resource concepts and sub concepts at the moment. A resource is constructed based on the data and meta data captured by the context sensors. The detailed description of the resource discovery and construction processes is given in Section 3.5.2.

Relations are defined between concepts of the resource dimension and the action dimension for modeling on which resources what kind of user actions have been performed. For example, if the user enters a text in a Microsoft Word document, all keyboard entries are instances of the `Event` concept, connected via the objecttype property `isActionOn` to the same instance of a `TextDocument` (and a `FileResource`) representing that document.

3.3.3 Information Need Dimension

The information need dimension represents the context-aware proactive information delivery aspect of the UICO. An information need is detected by a set of fixed rules based on the available user context information. The `InformationNeed` concepts has properties to define the accuracy of the detection and the importance to fulfill the information need in a certain time-frame. For details about information need detection it is referred to [Rath *et al.*, 2007].

An information need is associated with the user's action(s) that trigger(s) a rule. Hence a connection between the information need dimension and the action dimension exists. The resource dimension is also connected to the information need dimension in such a manner that each resource that has been found for fulfilling the user's information need is related to this one via the objecttype property `suggestsResource`.

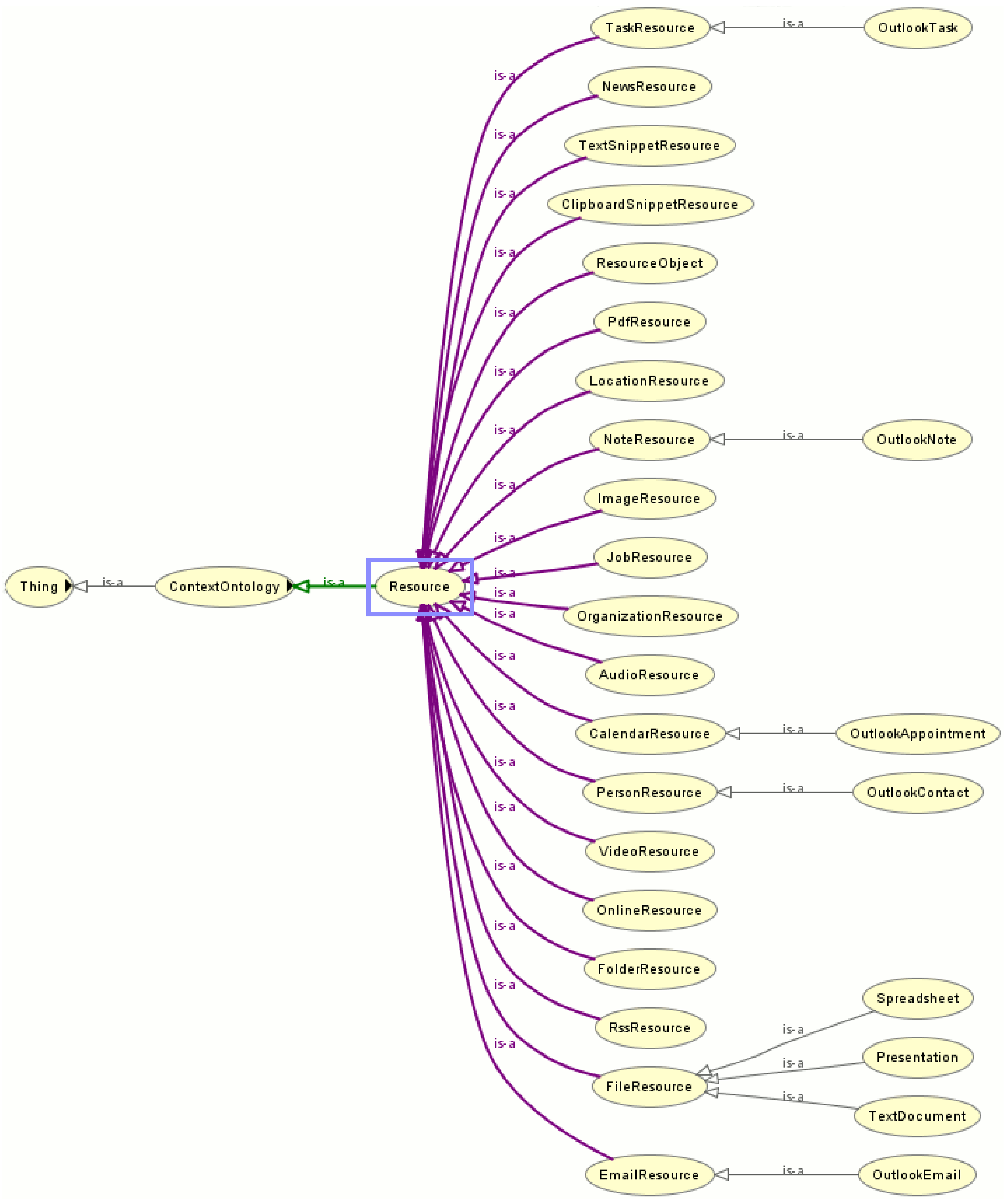


Figure 3.5: This figure shows the resource dimension of the user interaction context ontology (UICO) visualized in the Protégé ontology modeling tool [Protégé, 2009].

3.3.4 User Dimension

The user dimension contains two concepts, the `User` and the `Session` concept. The `User` concept includes basic user information such as user name, password, first name and second name. The user dimension is related to the action dimension in such a way that each `Action` is associated with a `User` via the objecttype relation `hasUser`. Indirectly the user dimension is also related to the resource dimension and the information need dimension via the action dimension. The `Session` concept is used for tracking the time of user logins and the duration of a user session in the application.

3.3.5 Application Dimension

The application dimension is a *"hidden"* dimension because it is not modeled as concepts in the UICO. This dimension is present in such a way that each user interaction happens within the focus of a certain application, e.g., the user's desktop, Microsoft Word or the Microsoft Windows Explorer. The `Event` concept holds the information about the user interaction with the application by the datatype properties `hasApplicationName` and `hasProcessId`. Standard applications that run on the Microsoft Windows desktop normally consist of graphical user interface (GUI) elements. Also console applications have GUI elements such as the window itself, scroll bar(s) and buttons for minimizing, maximizing and closing the application. Most of the GUI elements have an associated *accessibility object* [Microsoft, 2009] which can be accessed by context sensors. Datatype properties of the `Event` concept hold the data about the interactions with GUI elements. Later in Section 7.7 it is shown that these accessibility objects play an important role in automatic task detection.

A resource is normally accessed and manipulated by the user within an application hence there is a relation between the resource dimension and the application dimension. This relation is indirectly captured by the relation between the resource dimension and the action dimension, i.e., by the datatype property `hasApplicationName` of the `Event` concept.

For a user it is not convenient to manually enter the data about her user interaction context on such a fine-granular level. Hence semi-automatic and automatic mechanisms are required to ease the population process.

3.3.6 Comparison with existing Personal Information Ontologies

The UICO is similar to the *Personal Information Model Ontology* (PIMO) [Sauermann *et al.*, 2007] in terms of representing desktop resources. However, for the purposes of automatic user interaction context capturing, a limitation of the PIMO is the coarse granularity of concepts and relations. The UICO is a fine-grained ontology, driven by the goal of representing automatically captured low-level user interaction information with applications and resources on the computer desktop whereas the intention of PIMO is to enable the user to manually extend the ontology with new concepts and relations to define her environment for personal information management. Although it would be possible to allow the user to manually add new concepts (e.g., tags) and relations (e.g., `isProjectMemberOf`, `hasTag` etc.) to the UICO this is not the focus.

Native Operations (NOP) [Biedert *et al.*, 2008] define operations on information objects, so called *data objects*. These are similar to the UICO's `ActionType` concepts and more specifically to the `EventType` concepts. The `DataObject` concepts of the NOP ontology contains representations of several desktop resources but only in a more coarse granular way than the UICO's `Resource` concepts.

In [Xiao & Cruz, 2005] a layered and semantic ontology-based framework is described which follows the principles of *semantic data organization*, *flexible data manipulation* and *rich visualization*. The framework consists of an *application*, *domain* and *resource layer* as well as a *personal information space*. The resource dimension of the UICO can be seen as a combination of the domain and resource layer because resource instances are mapped to concepts of the domain layer. The intention of the approach is to propose a framework for PIM whereas the UICO focuses on representing the user interaction context. The main differences to the UICO are (i) the lack of an ontology for resources and (ii) the missing concepts and relations for representing user actions.

3.4 Context Sensors and Context Observation

Context sensors, also referred to as context observers or just sensors, observe the user's interactions with resources, applications and the operating system, which is a similar approach as followed by contextual attention metadata [Najjar *et al.*, 2006; Wolpers *et al.*, 2007] and other context observation approaches [Budzik *et al.*, 2001; Chernov *et al.*, 2008; Dragunov *et al.*, 2005; Kleek & Shrobe, 2007; Lokaiczyk *et al.*, 2007]. This section deals with the mechanisms used to capture the user's behavior in achieving her goals. For this low-level operating system and application events initiated by the user while interacting with her desktop, are recorded by context sensors. Context sensors observe the user interaction context. The information about the occurred events is sent as a XML stream to the context capturing framework for processing and analysis. In this processing and analysis step, the information from different sensors are collected and aggregated. The result of these steps is a semantic representation of the metadata and data about the events received from various sensor clients.

The targeted domain of this dissertation research is the Microsoft Windows environment. Especially, the focus is on supporting tools office workers are likely to use in their daily work. Through informal interviews following applications were identified to be worthwhile for developing context sensors for: the Microsoft Office suites, Microsoft Internet Explorer, Mozilla Thunderbird, Mozilla Firefox, Novell GroupWise, and Microsoft Outlook.

3.4.1 Context Observers

Context observers, also referred to as context sensors or simply sensors, observe the users' interaction behavior on their computer desktop. Context sensors are programs, macros or plugins that exploit application program interfaces (APIs) of applications and the operating system. Sensors on the computer desktop are distinguished based on the origin of the sensor data they deliver. System and application sensors were developed [Rath *et al.*, 2007]. It could be also reasonable to integrate an environmental sensor [Zimmermann *et al.*, 2005], like for example, a GPS receiver, for mobile applications, or biometric sensors for getting information about physical conditions of the user, for example, the user's stress level.

Sensor	**Observed Metadata and Data**
File System Sensor	copying from/to, deleting, renaming from/to, moving from/to, modification of files and folders (file/folder URL)
Clipboard Sensor	clipboard changes, i.e., text copied to clipboard
Network Stream Sensor	header and payload content from network layer packets (http, ftp, nttp, smtp, messenger, ICQ, Skype,. . .)
Generic Windows XP System Sensor	mouse movement, mouse clicks, keyboard input, window title, date and time of occurrence, window id/handle, process id, application name, accessibility objects
Accessibility Object Sensor	name, value, description, help text, help text description and tooltip of the accessibility object.

Table 3.2: This table lists the context sensors for the Microsoft Windows operating system.

System sensors: System sensors capture data related to system events. System events are input device data streams, like for example, keystrokes, mouse movements and mouse clicks. On the operating system level information about clipboard events, file system changes and network stream monitoring [1] is also included. A detailed overview of the developed system sensors is given in Table 3.2.

Application sensors: Application sensors collect information about the user's behavior when interacting with specific applications. Typically application sensors are plugins, macros or small programs that utilize application specific libraries for the user interaction context observation. Application sensors for Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Microsoft Explorer, Microsoft Internet Explorer 6, 7 and 8, Mozilla Thunderbird and Mozilla Firefox [2], Microsoft Outlook 2003 and 2007 were developed. These sensors were utilized for the experiments described in Chapter 7.

In Table 3.3 a comprehensive list of the supported applications including the captured data and metadata are listed.

3.5 Sensor Data Abstraction and Aggregation

This section describes the steps from low-level sensor events containing semi-structured data to structured data in from of a populated ontology. The algorithms and techniques used for discovering user interaction with resources and relationships between resources in the event data stream as well as aggregating events to blocks of events (event blocks), are described and elaborated in this section.

[1]The Network Stream Sensor was developed by Pichler [2007].

[2]The Mozilla Thunderbird and the Novell GroupWise sensor were developed by Rechberger [2007].

Application	**Observed Metadata and Data**
Microsoft Word	document title, document URL, folder, user name, language, text encoding, content of visible area, file name
Microsoft PowerPoint	document title, document URL, document template name, current slide number, file name, language, content
Microsoft Excel	spreadsheet title, worksheet name, folder, spreadsheet URL, user name, authors, language, content of the currently viewed cell, file name, file URI
Microsoft Internet Explorer	currently viewed URL, URLs of embedded frames, content as HTML and content as plain text
Microsoft Explorer	currently viewed folder/drive name, URL of folder/drive path
Mozilla Firefox 2.x and 3.x	currently viewed URL, URLs of embedded frames, content as HTML
Mozilla Thunderbird	(HTML/plain text) content of currently viewed or sent email, subject, unique path (URI of email/news message) on server, user's mail action (compose, read, send, forward, reply), received/sent time, email addresses and full names of the email entries
Microsoft Outlook 2003/2007	(create, delete, modify, open and distribute) tasks, notes, calendar entries, contacts and data about email handling
Novell GroupWise email client	(create, delete, modify, and distribute) tasks, notes, calendar entries, and todos, and data about email handling like in the Mozilla Thunderbird application

Table 3.3: This table shows a listing of the developed application specific sensors.

3.5.1 Event Creation

Events are observed by application and operating system sensors and submitted to a central processing unit that transforms the XML-based data to an object that can be further manipulated. This process is similar to the contextual attention metadata (CAM) approach [Wolpers *et al.*, 2007] for supporting learners or building user profiles. Based on the event object an instance of the `Event` concept is created that has the same identifier (URI) as the event object. Resource discovery and resource building algorithms (see Section 3.5.2) as well as algorithms for detecting the user's information need (see Section 3.2) are applied. The results of the applied algorithms lead to new instances of the `Resource` concept, the `InformationNeed` concept and the `EventBlock` concept as well as relations between the concept instances. What kind of instances and relations are created and added are elaborated in the following sections.

3.5.2 Resource Discovery and Resource Building

Resources can be links, documents, persons, emails, files, folders, web pages, organizations, locations, files, folders, presentations, text documents and spreadsheets. *Resource discovery* is about the identification of resources and the extraction of meta data about the referenced resources in the user interaction context data stream, referred to as the *event stream.* Furthermore resource discovery deals with finding resources the user has interacted with as well as identifying resources that are included or referenced in used resources. A resource is included in another resource if the content of a resource is part of the content of another resource, e.g., copy of a part of a text from the content of an email to a text document. A resource is referenced by another resource if the location of the resource appears in the content of another resource, e.g., a link to a web page appears in the content of an email.

Three techniques are applied to discover the resources. These are (i) the *regular expression*, (ii) *the information extraction* and (iii) the *direct resource identification* approach.

1. **Regular Expressions:** The *regular expression* approach identifies resources in the event stream based on certain character sequences predefined as regular expressions. This approach is utilized to identify files, folders, web links and email addresses for example.

2. **Information Extraction:** The *information extraction* approach extracts person, location and organization entities in text-based elements in the event stream. These entities have to be predefined in a so called look-up list. A look-up list entry is a pair of strings, in which the first element defines the character sequence to look for. The second element is the uniform resource identifier (URI) which uniquely identifies the found character sequence as a specific resource. An example of such an entry for a person is `(Rath Andreas; mailto:arath@know-center.at)`, for an organization `(Know-Center; http://www.know-center.at)` and for a location `(Graz; http://www.graz.at)`.

3. **Direct Resource Identification:** The *direct resource identification* approach uses the data about a resource sent by the context sensor to directly identify and build the resource. With this approach it is possible to directly map certain fields of the event stream data to a resource. An example for this is the sensor data about an email that the user has opened. It this case the sensor sends the information that a specific email identified by the `server message id` has been opened for reading. Additionally, metadata about the email is attached by the sensor an added to the referenced resource. Another example is the `ClipboardSnippetResource` which is built based on the content of the clipboard application sensed by the clipboard observer. In this case an artificial URI is calculated based on the content of the clipboard application in order to uniquely identify it.

The resources identified by these resource discovery mechanisms are related to instances of the `Event` concept by the `isActionOn` objecttype property. There are two special types of resources: (i) *fake resources* and (ii) *artificial resources.* Fake resource are introduced to support the understanding of the relation between sensor data and resource discovery whereas artificial resources allow a more fine-granular tracking of user interactions on parts of resources.

Fake Resources: Fake resources are resources that are not identifiable in the application nor in the file system. Typical examples are the following: suppose a new Microsoft Word 2007 document or a new Microsoft PowerPoint 2007 presentations has been been created without being saved afterwards. This document has the window title “Document1 - Microsoft Word” or “Presentation1 - Microsoft PowerPoint” respectively. The sensed path of the word document is

"Document1". For the presentation the sensor does not sense a path at all but senses the name of the presentation as being "Presentation1". These peculiarities appear for multiple sensors and use cases. These special cases have to be taken into account when grouping events to event blocks as well as in the resource discovery and resource building process.

Artificial Resources: Artificial resources are resources that can not be identified by an unique URI pointing to a file system or web location. Examples for artificial resources are parts of resources, e.g., a text paragraph of a document or sub page or frame of a web page. Identifying this type of resources is interesting for tracking user interactions on a fine-granular level, i.e., user interactions with parts of resources. Consider that the user selects some text in a resource, then the selected text is transformed into an artificial resource with an unique identifiable automatically generated URI. This allows to identify which parts of a text appears in different resources. Artificial resources allow to study which parts of a resource are edited, read, formatted or extended by a user. Merging multiple user interaction contexts enables then to identify collaboration topics and the user's collaboration/editing role (e.g., reader, writer, designer, etc.).

Resources of the type `ClipboardSnippetResource` are also artificial resources. If a user copies some text from one resource to another, one an instance of a `ClipboardSnippetResource` concept is built and related to the resource where the text has been copied from as well as the target resource via the `isIncludedResourceFrom` objecttype property. Instances of the `Copy` and `Paste` concept which are a sub-concept `EventType` and hence a sub-concept of `ActionType` are related to the instance of the `ClipboardSnippetResource`. The difficulty here is to identify the paste location. One approach is *"monitoring the clipboard captures Cut and Copy episodes, and Paste episodes are determined through analysis of changes in window contents"* [Pedersen & McDonald, 2008]. A different much simpler approach is introduced and followed in this research which is user interaction driven. It leverages the fact that the user has to interact with input devices or specific application and desktop elements to do a "copy", "cut" or "paste". Accessibility objects are used to identify if a user clicks on a "copy", "cut" or "insert" button or respective menu elements. Since the name of the desktop elements derived from accessibility objects are language depended only algorithms for the targeted languages, which are German and English, were implemented. The monitoring of key stroke events also gives hints if a user copies and pastes when looking for key stroke combinations like "Control+C" and "Control+P" on an English keyboard or "Strg+C" and "Strg+V" on an German one. This approach is easier to realize than window content analysis but has a few drawbacks. One is that it does not take into account manually configured keyboard shortcuts by the user. Another one is that it fails to recognize copy and paste events if the application does not follow the standard keyboard shortcuts for copy, cut, paste or does not name the menu elements in an intuitive way.

Until this point no aggregation in terms of grouping actions of the user that belong together was done.

3.5.3 Event to Event Block Mapping Rules

The context sensors observe low-level context data which result in events. For aggregating these events *logically* to blocks of events, so called event blocks, static rules are used. Logically in this sense means to group the events that capture the interactions of the user with a single resource together to an event block. Resources can be of various types (see Section 3.2) and used in various applications. Therefore for different types of applications different rules are applied in the grouping process. An application can also handle multiple resource types. As an example for such an application is Microsoft Outlook or Novell GroupWise in which emails, tasks, notes, appointments and contact details are handled. The complexity and accuracy of the static rules depend on the mechanism of the application to identify a single resource and on the possibility to capture this resource id with a sensor. If the application is not possible to deliver an unique identifier for a resource or it is not possible for a sensor to capture it then heuristics are used for identifying a resource.

Sensors influence the static rules: The static rules are further dependent on the received sensor data which is encapsulated in the attributes of the events. This means that if a sensor is not capable of delivering information about a specific attribute it can not be used in the event to event block mapping process. From the experience with the sensors gained in the course of the prototype testing and evaluation phases, some sensors seem to be more reliable than others. Especially the in deep operating system calls are sometime failing in delivering information about the pressed key strokes as well as the window and application currently in focus. For these reasons several backup scenarios are included in the mapping algorithms to get an accurate grouping.

Rules are dependent on the application domain: The domain in which the sensors are used for context observation and what aspects of the user's context to be observed direct the number of event to event block mapping rules required. Since the targeted environment of this research is the computer desktop of office workers, the rules are developed for common office applications. Sensors can be easily extended to other domains and applications if applicable.

Difficulty of Just-In Time Sensor Fusion: Sensor fusion here means the combination of data from various sensors to achieve a better information about the user interaction context. The challenge here origins from the fact that multiple sensors observe complementary information about the same user interaction but send this information at different points in time. The difficulty is to align the correct events and combine the data and hereby enrich the information about the user interaction context. It is not clear if there is another sensor that senses the same user interaction or how long to wait for another "similar" event from another sensor. For grouping events to event blocks a 3-events window "online processing", i.e., "just-in time processing" approach is followed. This means that an event will be assigned to an event block immediately. A dynamic refining of this assignment takes place for the last three events.
As an example consider a user writing an email. In the case of the utilized sensors for the experiments (see Section 3.4) the generic context observer recognizes the user inputs but misses the details about the email resource. These details are sensed by the Microsoft Outlook sensor

at two points in time when (i) the user clicks on *"Create new mail"* button and (ii) when the user sends the email. The effect is that both sensors observe the creation of the email and the sending of the email. These two events and all the events about entering the email text have to be grouped to the same event block. Furthermore the order in which the events are received is not fixed. Since the event block grouping rules do not have the details about a resource being available throughout all the events. The approach followed is to group "intermediate" events based on more general terms, i.e., the `window title` or `application name`.

3.5.3.1 Support Methods

The static rules listed in the following sections make use of so called *support methods* which help describing the functionality of the algorithms.

Support methods:

- The *TokenSimilarity(String s1, String s2)* calculates the token based similarity between the two supplied string values with the cosine similarity measure [Manning & Schutze, 1999]. The implemented measure of similarity is $similarity = \cos(\theta) = \frac{A*B}{\|A\|*\|B\|}$, whereas A and B are token vectors of n dimensions. Each dimension represents a token of the strings. The cosine of the angle between these two token vectors constitute the similarity measure.

- The *WeightedTokenSimilarity(String s1, String s2)* calculates the weighted token similarity between the two supplied string values based on the cosine similarity measure [Manning & Schutze, 1999]. The similarity measure is the same as for the *TokenSimilarity* except that the token vectors A and B are extended with new entries. The new entries comprise of equal sub-strings of the strings $s1$ and $s2$ which will get a weight of 2. Suppose $s1$="My name is Adam." and $s2$="Your name is wonderful." then the token vectors $A = \{$`my`,`name`,`is`,`Adam`$\}$, $B = \{$`Your`,`name`,`is`,`wonderful`$\}$ are extended both with a "`name is`" token with a weight of 2 because of the same sub-string.

- The *CharacterSimilarity(String s1, String s2)* calculates the character based similarity between the two supplied string values based on the cosine similarity measure [Manning & Schutze, 1999]. The used measure of similarity is $similarity = \cos(\theta) = \frac{A*B}{\|A\|*\|B\|}$, whereas A and B are character vectors of n dimensions. Each dimension represents a character appearing in the strings. The cosine of the angle between these two character vectors constitute the similarity measure.

- The *DefaultApplicationMappingRules(Event event, EventBlock eventBlock)* method stands for the default application rules that are described bellow in Section 3.5.3.2. These rules are used as a backup scenario if the application specific attributes for mapping the event to an event block are not sufficiently available in the event stream.

3.5.3.2 Default Application Rules

The *default application rules* are rules that are applied if no application specific rule is defined for the currently processed event. The goal of these rules are to heuristically group events to event blocks based on event attributes which can be observed application independently. Example for such kind of attributes are the title of the window of the application, the windows process number/id and the id of the window handle. The window handle id[3] is a unique identification of the window that is constructed by the Microsoft Windows operating system's window manager. The window title and the process id showed the best choice for a generic event to event block grouping in which no application specific attributes are present. In SWISH [Oliver *et al.*, 2006] a Microsoft research team showed that they can utilize the window title to identify selected tasks by about 76% accuracy. This finding goes hand in hand with Shen & Dietterich [2007] in which the window title next to the file path of the currently edited document played an important role. A description of the algorithm is presented bellow.

Algorithm 1 Default application rules for the aggregation of events to event blocks.

```
Let  $e$ denotes the current event and $e_{<ea>}$ its attribute $< ea >$
Let  $eb$ denotes the previous event block and $eb_{<eba>}$ its attribute $< eba >$
Let  $eb^{e^{last}}$ denotes the last event of the previous event block
     and $e^{last}_{<ela>}$ the last event's attribute $< ela >$
1: if $\exists eb$ then
2:   if $\nexists e_{hasWindowTitle} \vee \nexists\ eb_{hasWindowTitle}$ then
3:     if $\exists e_{hasProcessId} \wedge \exists eb^{e^{last}_{hasProcessId}} \wedge (e_{hasProcessId} == eb^{e^{last}_{hasProcessId}})$ then
4:       return true
5:     else if $\exists e_{hasApplicationName} \wedge \exists eb^{e^{last}_{hasApplicationName}}$
       $\wedge (e_{hasApplicationName} == eb^{e^{last}_{hasApplicationName}})$ then
6:       return true
7:     else
8:       return false
9:     end if
10:  else
11:    $Sim_c \leftarrow CharacterSimilarity(e_{hasWindowTitle}, eb_{hasWindowTitle})$;
12:    $Sim_{wt} \leftarrow WeightedTokenSimilarity(e_{hasWindowTitle}, eb_{hasWindowTitle})$;
13:    if $Sim_c > 0.985$ then
14:      return $Sim_{wt} > 0.7$
15:    else
16:      $Sim_t \leftarrow TokenSimilarity(e_{hasWindowTitle}, eb_{hasWindowTitle})$;
17:      return $Sim_t > 0.8 \vee Sim_{wt} > 0.85$
18:    end if
19:  end if
20: else
21:   return false
22: end if
```

[3]NativeWindow Class, `http://msdn.microsoft.com/en-us/library/system.windows.forms.nativewindow.aspx`.

Attribute Description

- The `hasWindowTitle` is the title of the window that was instantiated by the application and drawn by the operating system's window manager. The operating system is responsible for drawing these windows and it is possible to access it with operating system libraries.
- The `hasProcessId` is the unique identification of an instance of an application during runtime. On a Microsoft Windows operating system this is a positive number.
- The `hasApplicationName` is the name of the application the user has interacted with.

Algorithm Description

The algorithm checks in the first place if the preferred generic event to event block grouping feature, the window title, is present or not (line 3). If this feature is not present then the process id of the window that is currently in focus will be used as the discriminating feature. If this feature is also not available, then the name of the application of the currently window in focus will be used for deciding if the event belongs to the last event block or should be used as the basis for building a new event block. But if the window title is available then it will be compared to the window title of the last event of the event block. The comparison takes place on two levels, on the character and on a token level. The cosine similarity measures between the characters and the tokens used in the window title are computed to decide if these two are similar. The thresholds used in the algorithm come from exploring and analyzing thousands of events about user interactions with various applications and resources. The threshold values have continuously been refined. Since the developed prototypes allowed the simulation of events, the threshold values and the algorithm could be further fine-tuned.

3.5.3.3 Clipboard Rules

The rules for the data from the clipboard sensor is simple because the action of copying a text snippet into the clipboard happens in the same application as the previous event. This previous event is an event about the interaction with the same application. Suppose the previous event would be an event in application $A1$ and the next seen event in another application $A2$, then this one would be the event of switching from $A1$ to $A2$, i.e., set the focus to the window of $A2$ and hence cannot be the clipboard sensor event. The only exception to this would be a sensor that sends data without any previous interaction from the user, for example, a sensor that notifies about a new incoming email. Since there are no sensors of this type deployed in this system, this use case is obsolete at the moment.

Algorithm 2 Rules for the aggregation of events from the clipboard sensor to event blocks

```
Let  $eb$ denotes the previous event block and $eb_{<eba>}$ its attribute $< eba >$
1: if $\exists eb$ then
2:     return true
3: else
4:     return false
5: end if
```

3.5.3.4 FileSystemWatcher Rules

The *FileSystemWatcher* is a sensor that monitors file system changes. Since many operating system and application services and applications use temporary files such kind of sensor is constantly creating a lot of events in a short time period. To restrict the events to file system changes that are useful for the aim of detecting the user interaction context, the sensor only monitors files and folders that have been at least accessed once by the user.

There is no special algorithm for grouping events from the *FileSystemWatcher* sensor to event blocks. Every event from this sensor is added to the last event block as a reaction of the file system to the behavior of the user.

3.5.3.5 Mozilla Thunderbird and Novell GroupWise Rules

These rules aggregate events from the Mozilla Thunderbird and Novell GroupWise sensors to event blocks.

Algorithm 3 Rules for aggregating events from the Mozilla Thunderbird and Novell GroupWise sensors to event blocks

```
Let  $e$ denotes the current event and $e_{<ea>}$ its attribute $< ea >$
Let  $eb$ denotes the previous event block and $eb_{<eba>}$ its attribute $< eba >$
Let  $eb^{e^{last}}$ denotes the last event of the previous event block
     and $e^{last}_{<ela>}$ the last event's attribute $< ela >$
 1: if $\exists eb$ then
 2:   if $\exists e_{hasUsedResource} \wedge \exists eb_{hasUsedResource}$ then
 3:     return $e_{hasUsedResource} == eb_{hasUsedResource}$
 4:   end if
 5:   if $\exists e_{hasSensor} \wedge$ ( ($e_{hasSensor} ==$ DYOBS_SENSOR)
      $\vee$ ($e_{hasSensor} ==$ DYGOBS_SENSOR )) then
 6:     return $\exists e_{hasUri} \wedge \exists eb^{e^{last}_{hasUri}} \wedge (e_{hasUri} == eb^{e^{last}_{hasUri}})$
 7:   else
 8:     return $DefaultApplicationMappingRules(e, eb)$
 9:   end if
10: else
11:   return false
12: end if
```

Constants Descriptions

- The `DYOBS_SENSOR` constant stands for the name of the Mozilla Thunderbird sensor (see Section 3.4.1).
- The `DYGOBS_SENSOR` constant stands for the name of the Novell GroupWise sensor (see Section 3.4.1).

Attribute Description

- The `hasUsedResource` attribute of an event stands for the resource that was associated with the event during the resource analysis process (see Section 3.5.2).

- The `hasSensor` attribute of an event holds the name of the sensor which observed the event.
- The `hasUri` attribute of an event holds the unique id of a resource in the Mozilla Thunderbird and in the Novell GroupWise application.

Algorithm Description
At the beginning the algorithm checks if there has already been a resource associated with the event and with the event block during the resource analysis process and if they are equal. If yes then the event belongs to the event block.

If the `hasUsedResource` attribute is not sufficiently available then the `hasUri` attribute is used for the event to event block mapping decision but only if the event is from a special application sensor, the `DYOBS_SENSOR`, (Mozilla Thunderbird application sensor) or the `DYGOBS_SENSOR` (Novell GroupWise application sensor). Since both mentioned sensors are only sending events on selected types of user interactions, key stroke and mouse click events from the *Generic Windows Context Observer* have to be added to the corresponding event block as well. This is assured by applying the *default application mapping rules* (see Section 3.5.3.2) on those events. An example for such a use case would be the writing of an email. For this the user has to create a new mail. This is sensed by the special application sensor, while the actual writing of the email is captured by the *Generic Windows Context Observer.* The sending of the email results in an event from a special application sensor. All the resulting events from all sensors about that particular email belong to the same event block.

3.5.3.6 Microsoft Windows Explorer Rules

The Microsoft Windows Explorer sensor senses the user interactions in the Microsoft Windows Explorer, more specifically the navigation behavior of the user. The event to event block algorithm groups together all the events that happen in a specific folder view.

Algorithm 4 Rules for the aggregation of events from the Microsoft Windows Explorer sensor to event blocks

```
Let  $e$ denotes the current event and $e_{<ea>}$ its attribute $< ea >$
Let  $eb$ denotes the previous event block and $eb_{<eba>}$ its attribute $< eba >$
Let  $eb^{e^{last}}$ denotes the last event of the previous event block
     and $e^{last}_{<ela>}$ the last event's attribute $< ela >$
1: if $\exists eb$ then
2:    if $\exists e_{hasUsedResource} \wedge \exists eb^{e^{last}_{hasUsedResource}}$ then
3:       return $e_{hasUsedResource} == eb^{e^{last}_{hasUsedResource}}$
4:    end if
5:    return $DefaultApplicationMappingRules(e, eb)$
6: else
7:    return false
8: end if
```

Attribute Description

- The `hasUsedResource` attribute of an event stands for the resource that was associated with the event during the resource analysis process 3.5.2.

Algorithm Description
At the beginning the algorithm checks if there has already been a resource associated with the event and with the event block during the resource analysis process and if they are equal. If yes then the event belongs to the event block. If this comparison can not be carried out, the default application mapping rules (see Section 3.5.3.2) are applied for the event to event block grouping.

3.5.3.7 Microsoft Outlook Rules

The rules for assigning an event to an event block is straight forward for all the various resources which can be interacted with in Microsoft Outlook because they all have an unique *entry id.* This entry id is observed by the Microsoft Outlook sensor. Resources that are observed by the sensor are emails, tasks, notes, contacts and calendar entries.

Algorithm 5 Rules for the aggregation of events from the Microsoft Outlook sensor to event blocks

```
Let  $e$ denotes the current event and $e_{<ea>}$ its attribute $< ea >$
Let  $eb$ denotes the previous event block and $eb_{<eba>}$ its attribute $< eba >$
Let  $eb^{e^{last}}$ denotes the last event of the previous event block
     and $e^{last}_{<ela>}$ the last event's attribute $< ela >$
 1: if $\exists eb$ then
 2:   if $\exists e_{hasUsedResource} \wedge \exists eb^{e^{last}_{hasUsedResource}}$ then
 3:     return $e_{hasUsedResource} == eb^{e^{last}_{hasUsedResource}}$
 4:   end if
 5:   if $\exists e_{hasEntryId}$ then
 6:     for all $event$ such that $event \in eb$ from last to first do
 7:       if $\exists event_{hasEntryId}$ then
 8:         return $e_{hasEntryId} == event_{hasEntryId}$
 9:       end if
10:     end for
11:   end if
12:   return $DefaultApplicationMappingRules(e, eb)$
13: else
14:   return false
15: end if
```

Attribute Description

- The `hasUsedResource` attribute of an event stands for the resource that was associated with the event during the resource analysis process (see Section 3.5.2).
- The `hasEntryId` attribute of an event holds the unique id of a resource in Microsoft Outlook. Resources can be emails, notes, tasks, contacts and calendar entries.

Algorithm Description

At the beginning the algorithm checks if there has already been a resource associated with the event and with the event block during the resource analysis process and if they are equal. If yes, then the event belongs to the event block.

If this comparison is not possible then the algorithm takes the `hasEntryId` attribute of the event and compares it with the `hasEntryId` attribute of each event of the last event block. If an entry id exists in one of the events of the event block and matches, then the event belongs to the last event block. If it does not match, then it does not belong to the last event block. If no `hasEntryId` attribute is present in any event of the last event block then the default application mapping rules (see Section 3.5.3.2) are applied.

3.5.3.8 Microsoft Word, Excel, PowerPoint Rules

For observed events in the Microsoft Word, Microsoft Excel and Microsoft PowerPoint application this section describes the static event to event block mapping rules. The reason why there is only one set of rules for all three applications is that the attributes allowing a good identification of a specific resource are the same for all three applications.

Algorithm 6 Rules for grouping events from Microsoft Winword, Excel, PowerPoint to event blocks

```
Let $e$ denotes the current event and $e_{<ea>}$ its attribute $< ea >$
Let $eb$ denotes the previous event block and $eb_{<eba>}$ its attribute $< eba >$
Let $eb^{e^{last}}$ denotes the last event of the previous event block
    and $e^{last}_{<ela>}$ the last event's attribute $< ela >$
1: if $\exists eb$ then
2:   if $\exists e_{hasUsedResource} \wedge \exists eb^{e^{last}_{hasUsedResource}}$ then
3:     return $e_{hasUsedResource} == eb^{e^{last}_{hasUsedResource}}$
4:   end if
5:   if $\exists e_{hasPath} \wedge \exists eb_{hasPath}$
     $\wedge$ $CharacterSimilarity(e_{hasPath}, eb_{hasPath}) > 0.95$ then
6:     return true
7:   else if $\exists e_{hasName} \wedge \exists eb_{hasName}$ then
8:     if $WeightedTokenSimilarity(e_{hasName}, eb_{hasName}) > 0.8$
       $\vee$ $(CharacterSimilarity(e_{hasName}, eb_{hasName}) > 0.8\ )$ then
9:       return true
10:    else
11:      return $DefaultApplicationMappingRules(e, eb)$
12:    end if
13:  end if
14: else
15:   return false
16: end if
```

Attribute Description

- The `hasUsedResource` attribute of an event stands for the resource that was associated with the event during the resource analysis process (see Section 3.5.2).
- The `hasPath` attribute of an event holds the detected full path of the resource used in the application.
- The `hasName` attribute stands for the name, i.e. title, of the resource. For Microsoft Word this is the name of the document, for Microsoft Excel the name of the spreadsheet and for Microsoft PowerPoint the name of the presentation.

Algorithm Description

At the beginning the algorithm checks if there has already been a resource associated with the event and with the event block during the resource analysis process. If both associated resources are the same then the event belongs to the event block. This part of the algorithm is the same for all the other application specific rules.

If the comparison based on the used resources is not possible then the `hasPath` attribute is used to find out if the user continuous interacting with the same resource. If this attribute is not available than the `hasName` attribute of the event will be compared with the `hasName` attribute of the last event of the event block if present. The reason why a text comparison with thresholds is needed here is motivated by the following use case: suppose that a user opens a resource in or from the web browser, edits it and saves it to the local file system. While interacting with the resource, it lives only in memory and is not persisted in the local file system and therefore has no valid path. After saving the resource to the local file system the resource has a valid path. To not loose the events about the user interactions on this resource the name of the document is used for tracking. When the user saves the resource to the local file system it is likely that a similar file name is chosen. If none of the application specific attributes are sufficiently present than the default application mapping rules (see Section 3.5.3.2) are applied.

3.5.3.9 Microsoft Internet Explorer and Mozilla Firefox Rules

The rules for the grouping of events captured by the Microsoft Internet Explorer and the Mozilla Firefox sensor to event blocks are the same. The intention is to group all events that capture the interactions with a single web resources to an event block.

Algorithm 7 Rules for the aggregation of events from the Microsoft Internet Explorer and Mozilla Firefox to event blocks

```
Let $e$ denotes the current event and $e_{<ea>}$ its attribute $< ea >$
Let $eb$ denotes the previous event block and $eb_{<eba>}$ its attribute $< eba >$
Let $eb^{e^{last}}$ denotes the last event of the previous event block
    and $e^{last}_{<ela>}$ the last event's attribute $< ela >$
 1: if $\exists eb$ then
 2:   if $\exists e_{hasUsedResource} \wedge \exists eb_{hasUsedResource} \wedge$
      $(e_{hasUsedResource} == eb_{hasUsedResource})$ then
 3:     return true
 4:   end if
 5:   if $\exists e_{hasApplicationName} \wedge \exists eb^{e^{last}_{hasApplicationName}} \wedge$
      $(e_{hasApplicationName} \neq eb^{e^{last}_{hasApplicationName}})$ then
 6:     return false
 7:   end if
 8:   if $\exists e_{hasWebApplicationName} \wedge \exists eb_{hasWebApplicationName} \wedge$
      $(e_{hasWebApplicationName} == eb_{hasWebApplicationName})$ then
 9:     if $\exists e_{hasWebApplicationPageName} \wedge \exists eb_{hasWebApplicationPageName}$ then
10:       return $e_{hasWebApplicationPageName} == eb_{hasWebApplicationPageName}$
11:     else
12:       return true
13:     end if
14:   else
15:     return false
16:   end if
17:   if $\exists e_{hasDomain} \wedge \exists eb_{hasDomain} \wedge (e_{hasDomain} == eb_{hasDomain})$ then
18:     return true
19:   end if
20:   return $DefaultApplicationMappingRules(e, eb)$
21: else
22:   return false
23: end if
```

Attribute Description

- The `hasUsedResource` attribute of an event stands for the resource that was associated with the event during the resource analysis process (see Section 3.5.2).
- The `hasApplicationName` holds the name of the application from which the sensor captured the event. For Microsoft Internet Explorer the application name is `iexplore` or `explorer`.
- The `hasWebApplicationName` is the name of the web application that is available on the

web page. An example is an embedded wiki application.

- The `hasWebApplicationPageName` attribute stands for the name of the page of a web application. An example is a single wiki page of a wiki application.
- The `hasDomain` attribute is the domain of the web page which the user has interacted with.

Algorithm Description
At the beginning the algorithm checks if there has already been a resource associated with the event and with the event block during the resource analysis process and if they are equal. If yes, then the event belongs to the event block.

Since the algorithm is used for events captured in the Microsoft Internet Explorer and the Mozilla Firefox application, the algorithm has to check if the `hasApplication` attribute of the event is the same as the one of the last event of the event block. If they are not the same then the event does not belong to the event block.

For grouping the interactions of a user with a particular web application or even the interactions with a special page of a web application the next two rules check based on the `hasWebApplicationName` attribute and the `hasWebApplicationPageName` attribute if the user is still interaction with the same part of a web page.

On some web pages the URI of the currently viewed web page does not change when the user navigates on inner site pages, such that only the `hasDomain` attribute is present in the sensor data. For this reason the next rule is used to at least achieve a not that specific resource based grouping of events to event blocks. If there are no browser specific attributes present then the default application mapping rules (see Section 3.5.3.2) are applied.

3.5.4 Tasks

The aggregation of user actions into tasks is different form the previous rule-based approach since it would otherwise require to manually design rules for each task. This might be a reasonable approach for well-structured tasks, like administrative or routine tasks, but is obviously not appropriate for knowledge-intensive tasks that involve a certain freedom and creativity in the execution [Tochtermann *et al.*, 2006], e.g., "Planning a journey" or "Writing a research paper". To be able to also handle such kind of unstructured tasks the idea is to automatically extract tasks from the information available in the user interaction context model by means of machine learning techniques. Once automatically detected, these tasks will enrich the existing populated ontology model by contributing information about the user's tasks.

Chapter 4 describes and discusses several task detection task detection approaches. The ontology-based task detection approach proposed by this dissertation research is introduced in Chapter 5 and evaluated in Chapter 7.

3.6 Discussion about Ontology-based User Interaction Context Observation

Various representation formats were proposed by the research community for modeling the user's context. So why use an ontology to model the user context? Surveys from Strang & Linnhoff-

Popien [2004] and Baldauf *et al.* [2007] pointed out the strong points of using an ontology-based approach and advocate using ontologies for modeling the user context. Some strong points mentioned are strengths in the field of normalization and formality. This thesis research goes along with their conclusions and an ontology-based user context modeling approach. An ontology-based user interaction context model (UICO) was designed and implemented including context sensors and automatic ontology population mechanisms. This section reflects retro-perspectively on the advantages and disadvantages on the ontology-based approach.

3.6.1 Advantages

Using an ontology-based user context model brings several advantages next to the ones mentioned in recent surveys, such as [Rath *et al.*, 2009d]: (i) It allows to easily integrate new context data sensed by context observers to map the sensor data into a user context model. (ii) It can be easily extended with concepts and properties about new resources and user actions. (iii) The relationships between resources on various granularity levels can be represented. (iv) The evolution of datatype properties (i.e., data and metadata) into objecttype properties (i.e., relations between instances of ontology concepts) can be easily accomplished. (v) Being a formal model, it also allows other applications and services to build upon it and to access the encapsulated context information in a uniform way.

Easy Integration of new Sensors: Sensing context is done by so called sensors that observe the user desktop and the user's interaction with it (see Section 3.4). The kinds of sensors that are necessary depend on the context information that should be sensed which further depends on the application requirements. As an example, the types of sensors as well as the kind of sensor information for observing context information in an email application is different to sensing the location context on a mobile device. Here the strength of an ontology is to make the integration of new sensor data into the data model easy without invalidating previously recorded context data or rewriting database schemata. Ontology reasoners, like for example *Pellet* [Sirin *et al.*, 2007], can be utilized to validate the ontology model and check its consistency. The integration process only consists of adding new concepts and relations to the ontology and hence is very flexible.

Simple Mapping of Sensor Data to Ontology Concepts and Relations: The strengths of an ontology are in the fields of normalization and formalization as well as in the area of combining knowledge from different domains [Ötztürck & Amodt, 1997]. Because of the well-defined concepts and relations of an ontology the mapping of context data to them is straight forward.

Easy Extendability: Next to the unprocessed sensor data that may be included in an ontology-based user context model, inferred and derived information based on this data is stored as well. This new information about the user interaction context can easily be included by adding new relations between existing concepts or by adding new concepts and relations to the ontology. It also supports the "evolution" of datatype properties to objecttype properties if new inferring techniques or algorithms are added. An example for this is an instance of a `Resource` concept with the datatype property `hasContent` that holds a string literal representing the content of a text. Information extraction can detect cities mentioned in the text and hence connects the name of the mentioned cities as literals and the `Resource` instance with the `referencesCity` datatype property. There are no relations between cities with the same name until now. The required relations can be built by adding a new concept `LocationResource`, transforming the literals into instances of this concept and relating the instance of the `Resource` concept with the created `LocationResource` instance via the objecttype relation `referencesResource` utilizing `SPARQL CONSTRUCT` queries [Prud'Hommeaux & Seaborne, 2008].

Easy Access and Integration: The user context data has to follow the ontology specification. This can be verified by using reasoners that validate the data according to an ontology. This reliability allows other applications and services to easily access, query and process the user interaction context data. The easy access is provided by the utilization of the SPARQL query language.

From a **development perspective** good tools are already available for modeling ontologies. An example of a tool is Protégé [Protégé, 2009] which comes with different visualizations for ontologies and a variety of plug-ins. The OpenAnzo framework [OpenAnzo.org, 2008] used in this research also features the automatic generation of Java classes based on an ontology which eases/semi-automates the synchronization of the ontology and the Java object model.

From an **application perspective** an ontology-based user context model for this research enhances the performance of automatic task detection in comparison to previous mostly text-based approaches. This is achieved by extracting novel features from the ontology which increase the task detection performance on the studied datasets (see Chapter 7).

3.6.2 Disadvantages

The disadvantages experienced during the utilization of an ontology-based user context modeling approach for context observation on the computer desktop are (i) an increasing of computation time during the ontology population process, (ii) the increased query processing time on large populated ontology models, (iii) the large amount of data for storing all the sensor data and computed relations as well as (iv) the CPU and memory requirements.

Ontology Population Time: A population of an ontology is done by adding triples to the ontology model. The time required to add a triple increases by the size of the model. To avoid that two of the same triples are stored in the model, it has to be checked if the triple to be added already existed or not. In case of the triple store OpenAnzo [OpenAnzo.org, 2008] which is used in this thesis research, this check is performed with SPARQL queries. The query execution time increases by the size of the model and hence ontology population time increases. A solution implemented in this thesis research prototype for keeping the number of triples in the model low as well as assuring reasonable query execution time is to use named graphs [Sintek *et al.*, 2007] to split the big model into multiple smaller models.

Query Processing Time: Large populated ontologies consist of many triples/statements. By utilizing the SPARQL query language the query time increases as the number of triples increases. Complex SPARQL queries on large datasets may also take several minutes or even hours to terminate. In the automatic UICO population (see Chapter 3) it was important to keep the execution time of SPARQL queries as litte as possible to allow online user interaction context detection. This was successfully managed by reformulating and hence optimizing queries as well as splitting the populated user interaction context model into separate named graphs.

Large Amount of Data: Information about the user context is represented in terms of instances of concepts, literals, objecttype and datatype properties. These are stored as triples/statements. Depending on the granularity level of the context observation the number of triples of a populated context model for one day of work can easily result in 300000 to 500000 triples. A task approximately contains between 4000 and 50000 triples and requires in form of a RDF/XML file approximately 3 to 20 megabytes. The numbers of resulting triples and megabytes per task have approximated based on the tasks from the laboratory experiments described in Chapter 5. The amount of data can be reduced when switching off sensors and user interaction context analysis algorithms that are not needed for a specific type of context-aware application. The difficulty is to distinguish which features are required and which are not needed for a certain type of context-aware application. An example is task detection in which it is better to concentrate on sensing features with a high task discriminating power than sensing everything (see Chapter 7).

CPU and Memory Requirements: In the automatic ontology population step a significant amount of CPU performance and memory are required in order to analyze and to abstract the raw sensor information as well as constructing and executing SPARQL queries for inserting and verifying the information stored in the UICO. The low-level usage data is captured by context sensors. The CPU and memory usage of these sensors depend on the intervals of the user interactions. The shorter the intervals are the more data about the user interactions with resources, applications and the operating system has to be captured which increases CPU and memory load. The CPU and memory usage can influence the productivity of a computer desktop user. It decreases if the applications on the computer desktop stall and the operating system blocks any of the user's inputs. This normally happens if applications use a lot of memory and CPU performance. In case of the user interaction context detection approach, a standard desktop computer with 2 gigabytes of memory and a double core CPU with 2 GHz and all context sensors installed (see Section 3.4.1) can handle 8 events per second over a period of 8 hours (a work day) in the current implementation of the *KnowSe* prototype (see Section 6.3). In the context observation phases of the experiments described in Chapter 7 standard single as well as dual core desktop computers and notebooks with approximately 1-2 GHz CPUs and 1-2 gigabytes of memory were also successfully able to run the *KnowSe* prototype for recording task executions.

3.7 Lessons Learned

Continuous Resource Identification Data: Successful grouping of interactions on a single resource becomes more difficult if the sensors do not detect the data and metadata about the resource all the time, i.e., on each interaction with a resource. Heuristics and flexible rules have to be used for these “intermediate and incomplete” events.

Reliability of Sensors: Sometimes the sensors fail in observing events from the operating system or applications because of failing callback method calls of API methods or DLL (dynamic link library) calls to the in-depth operating system hooks. This leads to incomplete and not reliable sensor data which has also been identified by Pedersen & McDonald [2008] as an issue in automatic context detection. These issues have to be taken into account not only in the event to event block grouping algorithms but also in the resource discovery and resource building process.

3.8 Open Questions

Automatic user interaction context observation from the software development point of view seems to be almost “unlimited”. This means that the operating systems and the applications provide a variety of mechanisms to expose their usage. The questions which are still open are:

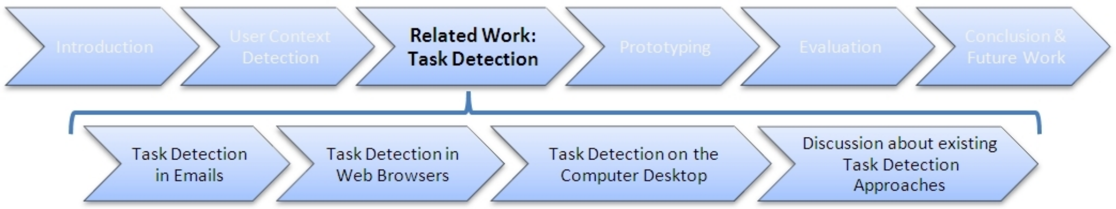
- Which sensors have to be developed?
- What kind of context features have to be observed/captured?
- On which granularity level should the context features be sensed?

For studying these questions in respect to the research goal of enhancing automatic task detection via an ontology-based user interaction context model multiple experiments in diverse domains were designed and executed. The following chapters will unveil more insights about which context features are good discriminators for user tasks and hence worth observing for detecting the user’s tasks automatically.

3.9 Summary

This chapter introduced the term *user interaction context* and the conceptual model referred to as the *Semantic Pyramid* for representing this *user interaction context.* The *user interaction context ontology* (UICO) as a realization of the *Semantic Pyramid* as an ontology-based user context model was presented and explained. Furthermore this chapter elaborated on the user interaction context observation mechanisms implemented and their capabilities to sense the user interaction context. Furthermore the algorithms for discovering resources and their relations in the event sensor streams were explained as well as the algorithms for aggregating low-level events, i.e., user interactions, to event blocks were shown. Hand in hand with the described algorithms the UICO population mechanisms were revealed. The advantages and disadvantages of an ontology-based user interaction context observation approach were discussed. At the end of this chapter lessons learned and open questions regarding automatic low-level user interaction context observation were mentioned.

4 Related Work: Task Detection


Introduction
User Context Detection
Related Work: Task Detection
Prototyping
Evaluation
Conclusion & Future Work
Task Detection in Emails
Task Detection in Web Browsers
Task Detection on the Computer Desktop
Discussion about existing Task Detection Approaches

This chapter gives an overview of the related work in the area of task detection on the computer desktop. A selection of relevant approaches to this research regarding automatic task detection in emails (see Section 4.2), in web browsers (see Section 4.3) and on the complete computer desktop (see Section 4.4) are highlighted in greater detail. A discussion of the relevant literature in respect to the research of this dissertation rounds off this chapter.

4.1 Introduction

Massive amounts of digital information are available to us today and are still constantly increasing. No matter if we talk about the information on the world wide web or the numerous documents, presentations, emails, and multimedia files we store on our computer desktops. Intelligent search technologies were developed to tackle the challenge of finding and re-finding information [Teevan, 2008] we need for achieving our goals but what is still missing is to understand the user context in which information is used and produced. Especially the relation between this user context and the user's task is of great interest [Callan *et al.*, 2007]. Task detection is an important sub challenge within user context analysis [Coutaz *et al.*, 2005; Dey *et al.*, 2001]: if the user's current task is detected automatically the user can be better supported with relevant information such as learning and work resources as well as task guidance. The relations between the used resources and the user's task also enriches personal information management [Dumais *et al.*, 2003; Teevan *et al.*, 2008].

Task detection belongs to the field of *activity recognition* [Andrews *et al.*, 2004; Horvitz *et al.*, 1998, 1999; Philipose *et al.*, 2004]. In activity recognition the system observes a sequence of events, tries to determine and understand the goals of the user, and responds to it. Task detection on the user's computer desktop can be further categorized into two categories: (i) *task switch detection* and (ii) *task classification*. Task switch detection which is a special case of the general problem of change-point detection involves *"monitoring the behavior of the* *user and predicting in real time when the user moves from one task to another"* [Shen, 2009]. Task classification deals with the recognition of a class/type of a task. An example for task classification is to recognize to which class the task "Planning an official journey to the CHI 2009 conference" belongs to. "Planning" would be an example task class for this task according to the task type classification of knowledge-intensive tasks proposed by *CommonKADS* [Schreiber *et al.*, 1999].

The focus in this section is put on approaches for detecting the user's task on the computer desktop since activity recognition can also be found in the ubiquitous computing field [Choudhury *et al.*, 2008; Favela *et al.*, 2007; Huynh & Schiele, 2005; Philipose *et al.*, 2004]. In this "off desktop" setting environmental sensors are preferably used as input for the activity recognition as opposed to system and application sensors used for task detection in computer desktop environments. For presenting a selection of the state of art approaches in the area of task detection on the computer desktop the following categorization is used: (i) task detection *in emails*, (ii) *on the web* and (iii) *on the whole computer desktop.*

4.2 Task Detection in Emails

Email has become a central tool in today's digital work environment. Since the amount of emails in the inboxes is continuously increasing several researchers have identified the issue of information overload [Whittaker & Sidner, 1996] in the user's inbox. The research challenges for coping with the big amount of email data focus on (i) reducing the amount of *"unwanted"* emails (spam and junk) [Balamurugan & Rajaram, 2008; Sahami *et al.*, 1998; Segal *et al.*, 2004], (ii) automatically categorizing emails to topics [Cselle *et al.*, 2007; Mock, 2001], (iii) email folders [Kiritchenko & Matwin, 2001] and (iv) activities [Dredze *et al.*, 2006; Kushmerick & Lau, 2005]. Text classification [Balamurugan & Rajaram, 2008; Boone, 1998] is one of the most popular approaches for classifying emails. In these approaches classifiers are trained based on the text content of an email. The majority of text-based email classification systems would use classification algorithms such as Naïve Bayes, rule learners, and support vector machines reported Dredze *et al.* [2006].

Regarding activity detection in emails, unsupervised learning approaches [Cselle *et al.*, 2007; Kushmerick & Lau, 2005] and supervised ones [Dredze *et al.*, 2006] were employed. Dredze *et al.* [2006] have developed the *SimSubset*, *SimOverlap* and *SimContent* algorithm in order to classify emails into activities. The first two algorithms compared the people involved in an activity against the recipients of each incoming message whereas the *SimContent* algorithm used a variant of latent semantic indexing called *iterative residual rescaling* [Ando & Lee, 2001] to classify emails into activities using the similarity based on message contents. They reported 93% accuracy of detecting activities by using the message content and message metadata (subject, sender, receiver) and the network of people involved in an activity. Kushmerick & Lau [2005] described an activity management system that clustered the user's email into user activities. This system considers the challenge of automatically learning an activity's structure which was modeled as a finite state automaton in order to derive workflow models. By utilizing text classification and clustering to attach labels, they reported accuracy values between 86% and 94% correctness in identifying the next state of an activity, the end of an activity and the *"message overlap"*[1].

4.3 Task Detection in Web Browsers

Kellar & Watters [2006] noted that the knowledge of a user's current task type would allow information filtering systems to apply useful measures for user interest. For that they studied the predictability of the task types *Fact Finding*, *Information Gathering* and *Transactions* in web browsers. Especially they focused on discovering the features significant for a specific task type. From a study with 21 participants consisting of university computer science students and 1192 tasks , they found strong differences in how the participants interacted with their web browsers during different tasks. By utilizing decision trees on the recorded task executions data they could successfully identify the task types of 53.8% of the task instances via stratified 10-fold cross-validation. In the first results of their experiments they discovered a high degree of variability across the participants implicit measures such as *dwell time*[2], *number of pages viewed*, and *number of windows loaded.* Driven by these results they trained the learning algorithm on the data from participants who performed more than 30 tasks separately. Hereby they achieved between 43.6% and 94.3% accuracy. An overview of all implicit measures used is given in Figure 4.1.

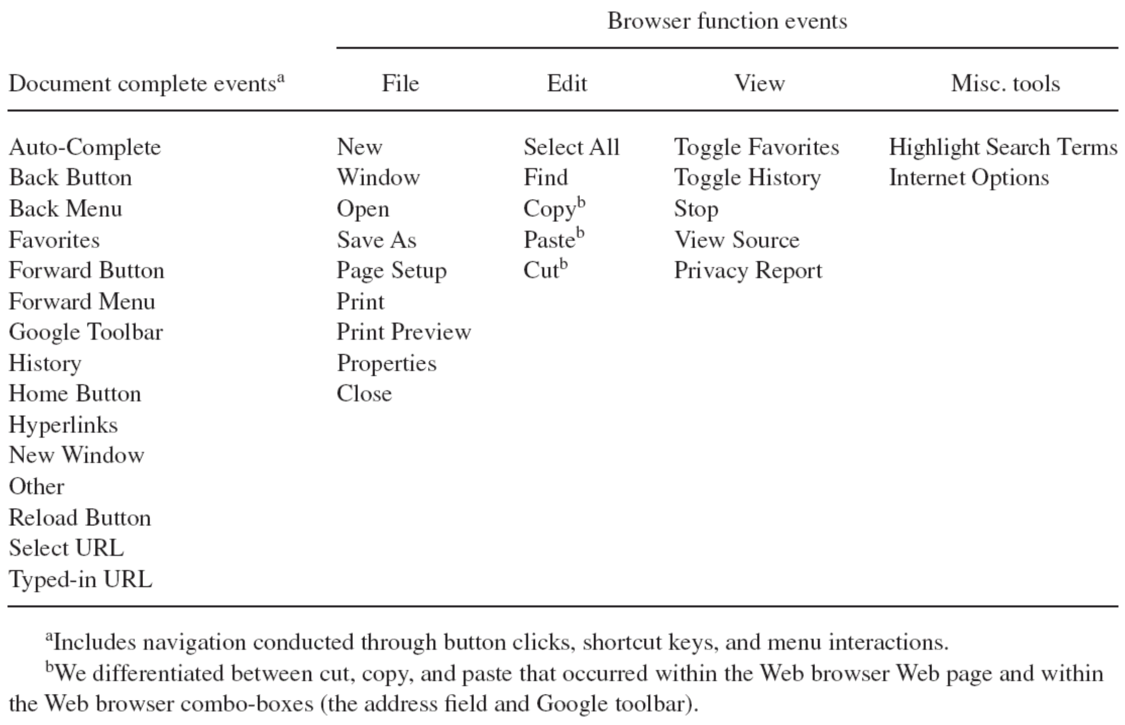
In 2008 Gutschmidt *et al.* [2008] studied different characteristics of user behavior regarding the execution of tasks of the task types similar to [Kellar & Watters, 2006; Kellar *et al.*, 2007]. They carried out a laboratory study with 20 students and employees of a university solving exemplary exercises on a single newspaper web page. The results showed that there were differences in the discriminating power for task types of the page view attributes *average page view duration*, *number of page views per minute*, *number of unique URLs in proportion to the total number of page views* as well as the *time spent on the start page of the newspaper web page.* In comparison to [Kellar & Watters, 2006] in which the Microsoft Internet Explorer was used, Gutschmidt *et al.* [2008] observed the user's interactions in the Mozilla Firefox browser via a self-developed extension. Mouse events, scrolling, keystrokes, tab events, browser events, page events were captured during the experiment sessions. The number of user context features captured were a subset of the ones observed by [Kellar *et al.*, 2007].

4.4 Task Detection on the Computer Desktop

One of the earliest systems for eliciting user goals was developed by Microsoft in the research project Lumiere [Horvitz *et al.*, 1998]. The Lumiere system is a predecessor of the Microsoft Office assistant application, nicknamed *"Microsoft Clippy"*, that assists the user in fulfilling her tasks based on a learned user model from user behavior. In Lumiere they used a Naïve Bayes user model derived from user interactions within the Microsoft Office applications to predict the user's goals as well as valuable learning content for the user. Lumiere is restricted to building user

[1]The message overlap is the overlap between the predicted and correct transitions' messages [Kushmerick & Lau, 2005].

[2]The *dwell time* is the amount of time participants spend reading and interacting with a particular web page [Kellar *et al.*, 2007]



	Browser function events			
Document complete events[a]	File	Edit	View	Misc. tools
Auto-Complete	New	Select All	Toggle Favorites	Highlight Search Terms
Back Button	Window	Find	Toggle History	Internet Options
Back Menu	Open	Copy[b]	Stop	
Favorites	Save As	Paste[b]	View Source	
Forward Button	Page Setup	Cut[b]	Privacy Report	
Forward Menu	Print			
Google Toolbar	Print Preview			
History	Properties			
Home Button	Close			
Hyperlinks				
New Window				
Other				
Reload Button				
Select URL				
Typed-in URL				

[a]Includes navigation conducted through button clicks, shortcut keys, and menu interactions.
[b]We differentiated between cut, copy, and paste that occurred within the Web browser Web page and within the Web browser combo-boxes (the address field and Google toolbar).

Figure 4.1: Implicit measures logged during Kellar et al. *[2007]'s field study about predicting tasks based on user interaction in web browsers. The figure is taken from [Kellar* et al.*, 2007].*

models from user interactions within Microsoft Office applications. Fenstermacher & Ginsburg [2002] argued that cross-application integration would be one of the key features of any user activity monitoring software.

In 2005 Fenstermacher [2005] worked on identifying processes based on low-level events and tasks derived from the complete user desktop by utilizing the TaskTracer system [Dragunov *et al.*, 2005]. The study participants were graduate students who were asked to write a summary of *Radio Frequency Identification* (RFID) technology and to create a corresponding brief Microsoft PowerPoint slide in 90 minutes. This study was *"simply a test of the technology and its installation, and not a carefully controlled experiment"* [Fenstermacher, 2005].

Czerwinski *et al.* [2004] carried out an in situ diary study for identifying task switching and interruption experiences of ten knowledge workers over the course of a week. They found out that (i) an average of 1.75 documents were utilized in an activity, (ii) the majority of tasks were described as routine tasks (27%), email related tasks (23%) and project-related (18%) tasks and (iii) more complex tasks, i.e., tasks that lasted longer and included more documents were difficult to switch to. They concluded that methods for capturing and remembering representations of tasks may be valuable in both reminding users about suspended tasks and in assisting users to switch among the tasks.

A selection of the most relevant task detection approaches to this thesis research is presented in greater detail in the following sections.

4.4.1 ActivityExtractor

The *ActivityExtractor* system [Mitchell *et al.*, 2006] extracts knowledge about the user's activities from emails based on raw workstation data via a clustering approach. Specific to the approach is that not only emails are considered for extracting activities or for classifying emails into activities but also the whole desktop content made accessible via the *Google Desktop Search* [3] index. Using this sophisticated search index the ActivityExtractor system can retrieve data about non email related sources like e.g., the online calendar of the user or the distribution of text terms in the index. This is also the reason why the ActivityExtractor system is mentioned in this section rather than in the previous email-based task detection section.

Mitchell *et al.* [2006]'s approach has three steps: (i) cluster emails based on the content of the email collection inclusive calendar meetings and person names, user-specified number of clusters and a set of activity names as input, (ii) perform social network analysis on each cluster based on the sender-recipient graph and (iii) construct a structured representation of each activity cluster as well as associating calendar meetings and person names with the activity. The output representation is an activity description consisting of (i) keywords, (ii) list of senders and recipient, (iii) fraction of user emails in this activity cluster, (iv) fraction of emails authored by the user in this activity cluster, (v) meetings as well as (vi) person names. They studied their approach based on a dataset from three user workstations. The three users A, B, C input a set of emails (A=2822,B=420,C=617), meeting text strings (A=1231,B=25,C=0), email addresses (A=2159,B=385,C=293), person names (A=470,B=76,C=56) and activity names (A=23,B=8,C=11). For each user they evaluated the performance of their approach separately. The accuracy values for correctly assigning emails to tasks were 73%, 80% and 83% for A, B and C respectively. They reported a positive influence of using social network analysis to refine the activity clusters initially formed based on email content clustering by an increase of 15% to 17% in the achieved accuracy. Furthermore they highlighted that the accuracy of the activity cluster also improved through utilizing the user activity names and the Google Desktop Search index.

4.4.2 APOSDLE Task Predictor

APOSDLE (Advanced Process-Oriented Self- Directed Learning Environment) is a framework that enhances the productivity of knowledge workers by integrating learning, teaching, and working [Lindstaedt *et al.*, 2005, 2008b]. In this project a task prediction component was developed that observed various user context features from the complete user's computer desktop in order to predict the active task of the user at any point in time [Lokaiczyk & Goertz, 2009; Lokaiczyk *et al.*, 2007]. An overview of all the observed user context features is given in Figure 2.4.

In [Lokaiczyk *et al.*, 2007] they studied the performance of decision trees, rule learners, Naïve Bayes, the incremental reduced error pruning (IREP) algorithm [Fürnkranz & Widmer, 1994] and support vector machines (SVM). The influence of the data preprocessing methods (i) splitting full text into textual features, (ii) information gain attribute selection and (iii) filtering frequent and noisy events as well as the number of used training samples were in focus of their investigations.

[3]Google Desktop Search: `http://desktop.google.com`

For real time task detection they developed a window slice algorithm that split the observed contextual features into a training instance for the classifiers every t seconds.

For evaluating their approach Lokaiczyk *et al.* [2007] designed an experiment with business tasks such as *market analysis*, *product design and specification*, *find and contact suppliers*, *contract placement*, and *triggering production*. These tasks were executed several times by one student in order to generate the evaluation dataset. The performance of the different classifiers were measured by applying stratified 10-fold cross-validation on the dataset. In [Lokaiczyk *et al.*, 2007] they reported a task detection accuracy of about 85% with SVMs. They suggest that the window size should not be lower than 5 seconds and should not exceed 30 seconds in order to achieve good accuracy values. Regarding the detection of window slices they noted that almost full coverage was achieved. From the paper Lokaiczyk *et al.* [2007] it is not clear in which ways the observed user context information is utilized to construct the training instances for the classifiers.

4.4.3 Dyonipos Task Recognizer

The Dyonipos [DYONIPOS, 2006; Granitzer *et al.*, 2009b; Tochtermann *et al.*, 2006] project aimed at supporting both knowledge workers and process engineers. For knowledge workers Dyonipos provided information need detection and proactive context-aware information delivery services [Rath *et al.*, 2007] as well as personal information management support [Rath *et al.*, 2008] in order to increase the productivity of knowledge workers. For the process engineer Dyonipos allowed to get insights into the knowledge workers working behavior and provided the process engineer with a support environment with advanced process modeling services, such as process visualization, standard process validation, and ad-hoc process analysis and optimization services [Rath *et al.*, 2006]. In the course of the Dyonipos project a task detection component was developed. Part of the following described research was also contributed by this dissertation effort and elaborated in detail by Granitzer *et al.* [2008] and Granitzer *et al.* [2009a]. An example is the conceptual model about the user interaction context, the *Semantic Pyramid*, on which also the ontology-based task detection approach described in Section 5 builds.

The goal of the task detection component of Dyonipos was to discover the current task of the user by utilizing classification algorithms and automatically observable user context features. User context features consisted of the *window title*, the *application name*, the *content* and the *semantic type*. The *application name* field contained the abbreviation of the application the user was working with. In the *content* field the content of the current window was stored. In the *window title* field the title of the current window of the application in focus was stored. The *semantic type* field contained a rule-based application depended type generated based on the observed context features, e.g., *read*, *write*, or *unknown*. For preprocessing these user context features they performed: (i) remove end of line characters, (ii) remove markup, e.g., `\&lg` and `![CDATA`, (iii) remove all characters but letters, (iv) remove German and English stopwords as well as formed *bag of words* for all fields and calculated the *tf/idf*[4] measure. For attribute selection the *information gain (IG)* measure was used.

Granitzer *et al.* [2008] evaluated classifiers and user context features for their applicability

[4]tf/idf: term frequency-inverse document frequency

to the task detection challenge by utilizing a dataset gathered from one user during a four week study in Austria's ministry of finance. They manually relabeled the training instances which were build on an event block basis, i.e., one training instance for the classifier resulted for each event block. By this method they constructed two datasets I and II. Dataset I and II contained 5 classes (email handling, paper writing, research, documentation, information collection) and 4 classes (communication, organization, writing, reading) respectively. This manual relabeling of the training data by a single expert based on the user assigned labels may have introduced wrong labels. In their evaluation the influence of three parameters was evaluated: (i) the number of attributes (ii) the classification model (Naïve Bayes, Support Vector Machines and k-Nearest Neighbor) and (iii) the field combination. On dataset I the k-Nearest Neighbor (k=1) performed best with the *semantic type* and *content* features with an accuracy of 74.51%, with 156 attributes, a precision of 0.91 and a 0.75 recall. On dataset II the k-Nearest Neighbor (k=1) performed best with the *window title* feature with an accuracy of 76.42%, with 188 attributes, a precision of 0.90 and a recall of 0.74.

4.4.4 Smart Desktop

The *Smart Desktop* system [Lowd & Kushmerick, 2009] is a desktop information management application for information workers. It was inspired by the TaskTracer system [Dragunov *et al.*, 2005]. In the Smart Desktop system there is a task detection component that allows to automatically infer the current project the user is working on. The hypothesis of their approach is that characteristics of the resources being accessed at a given point in time and the aspects of the user's ongoing activities (recent project, resources, etc.) are related.

Lowd & Kushmerick [2009] see the project prediction problem as a classical machine learning problem, more specifically a classification problem. For training the classification algorithms they engineer four different types of features based on automatically observed user context data: (i) resource features, (ii) past project features, (iii) salience features and (iv) shared salience features. *Resource features* consist of attributes about the utilized resources in a project including *full URI of the resource*, *meaningful sub-path of the URI*, *first 100 words in the body of a resource*, the *resource type* (e.g., web page, email, calendar entry, document, etc.) as well as specific fields of email messages. The *past project features* consist of attributes representing the last k projects the user worked on. The *salience features* result from the idea to recognize the similarity between current and recent user activities. A particular attribute of a resource feature which was last seen within a particular project represents a new attribute. In order to keep the number of generated attributes low only the feature types were used for building new attributes. The *shared salience features* were introduced to reduce the number of weights introduced by the set of salience features. A shared salience feature is constructed for each type of resource feature. The computation of the value of a shared salience feature is done by calculating the number of observed attributes of a particular type that was seen most recently in a particular project.

For evaluating the performance of their proposed features they recorded usage data from five employees over a two-week period in a real-world setting. They collected for each employee between 161-1181 resources distributed over 26-40 projects. This dataset also included 465-5036 time segments of activity data for each user. They evaluated four machine learning algorithms by a leave-one-out evaluation on a user basis: (i) the Naïve Bayes algorithm, (ii) the passive-aggressive algorithm, (iii) logistic regression and (iv) linear support vector machines. Their baseline was an expert system that combined expert knowledge and educated guesses. Users were asked to select from the engineered features a set of features to build the expert system. Lowd & Kushmerick [2009] reported that the Naïve Bayes algorithm was as competitive as the expert system and that the linear support vector machine made 10% fewer errors than the expert system. In conclusion they suggest to have data from several months and train logistic regression or linear support vector machine on the user's own data. From the paper by Lowd & Kushmerick [2009] it is unclear how high the detection accuracy, recall values or precision values were because only the error rates of the approaches were reported.

4.4.5 SWISH

SWISH [Oliver *et al.*, 2006] followed an unsupervised approach to identify the tasks of the user. The underlying assumption was that windows belonging to the same task share a common set of features. SWISH constantly monitors the user's desktop activities and utilizes the recorded stream of window events for extracting features for the clustering algorithm. Oliver *et al.* [2006] distinguished two types of context features on which they based their task detection on: (i) *semantic similarity of window titles* and (ii) *temporal closeness*.

The *semantic similarity* is computed based on word term vector representation of window titles. The window titles are preprocessed with stopword removal, splitting long words, and removing common application specific words as well as applying simple stemming and *tf/idf*-computation[5]. Analysis of patterns of window usage, i.e., how the windows were accessed by the user, constituted the *temporal closeness* feature. This features is based on the observation that related windows are used in a temporal sequence. Based on the transitions between the windows a *window switching matrix* is constructed. This matrix includes the number of switches between the windows and is used to construct a directed graph. In this graph the windows and transitions are represented by nodes and edges respectively. The weights of the edges are calculated based on the number of switches from the window where the edge starts to the window where it ends. Edges under a certain threshold are eliminated. The rest of the graph is used as input for the *Bron-Kerbosch* algorithm [Bron & Kerbosch, 1973] that groups the nodes related to each other together. The resulting groupings represent tasks. Based on these two types of features clustering was performed. SWISH does not need an advance labeling and hence no additional work on the user's side.

Oliver *et al.* [2006] reported task detection accuracy values of about 70% based on over 4 hours of real usage data observed from a single user. The dataset contained 5 tasks which were manually labeled for evaluating the clustering algorithm. In the experiments they employed a window size of 5 minutes and a threshold of 3 for constructing the directed graph of the window switching sequence. They also mentioned that 20% to 30% of the windows did not belong to any of the main 5 tasks. Furthermore they noted that applying the preprocessing steps had a high impact on the accuracy of their automatic task detection approach. With window title preprocessing

[5]tf/idf: term frequency versus inverse document frequency [Witten & Frank, 2005]

0.49 precision and 0.72 recall values were achieved in comparison to 0.39 precision and 0.65 recall values without applying the preprocessing. SWISH always used the full range of attributes and did not apply any attribute selection methods. Although the accuracy values with 70% for a clustering approach are good, the resulting clusters such as *"buying a book of Harry Potter"* do not seem to generalize well since the keywords representing the cluster were too specific to allow an abstraction for other books [Granitzer *et al.*, 2008] For future work they intended to look into other types of features that can be added in order to improve the task detection results.

4.4.6 TaskPredictor1

TaskTracer [Dragunov *et al.*, 2005] is an intelligent activity management system that supports the organization and re-finding of information based on activities, predicts folders the user is most likely to access and detects task switches.

For detecting task switches the *TaskPredictor 1* component [Shen *et al.*, 2006] of the TaskTracer system was developed. This component observes a sequence of events and constructs *window document sequences* (WDS) from it. For each WDS the *window title*, the *file path name* and the *URL of web pages* are extracted and represented as a set of words. This set of words is represented as a term vector in which each word is a binary variable that is set to 1 if the term appears in the WDS else to 0. For preprocessing a stopword list is used to eliminate very common words. Furthermore the Porter stemmer [Porter, 1997] is applied to stem English words to their roots. For attribute selection mutual information is employed. Mutual information, one of the most common measures of relevance for machine learning [Yang & Pedersen, 1997], is applied to select the 200 attributes with the highest predictive power. For the task switch detection a Naïve Bayes classifier is trained to decide whether to make a prediction or not. If the confidence of the classifier decision is over a certain threshold a prediction of the task switch is generated by using a support vector machine (SVM) [Joachims *et al.*, 2001].

Based on a dataset obtained from two users that included 177 tasks and about 10000 task switches *TaskPredictor 1* was able to attain a precision of 80% with a coverage of 10% for one user and a precision of 80% with a coverage of 20% for the other user [Shen *et al.*, 2006].

4.4.7 TaskPredictor2

The *TaskPredictor2* [Shen, 2009; Shen *et al.*, 2009] utilizes an online learning algorithm that only needs to take the last observation into account to update the classifier which reduces the computational costs. Driven by the disadvantages of the approach of *TaskPredictor 1* the focus of *TaskPredictor 2* was on increasing accuracy, memory and CPU costs, prediction delay and interruption cost as well as proactive association changes. This resulted in a new task switch detection system that computes an information vector for each minute that summarized the state of the user's desktop. Based on the comparison of two information vectors at time t and $t-1$ a prediction for a task switch is made. The information vector contains two types of features [Shen, 2009]: (i) *task-specific features* and (ii) *switch-specific features. Task-specific features* are the *strength of association of the active resource with a task*, the *percentage of open resources associated with a task* and the *importance of a window title word to a task. Switch-specific feature* are the *number of resources closed in the last s seconds*, the *percentage of open* *resources that were accessed in the last s seconds* and the *time since the user's last explicit task switch.*

For evaluating their approach they deployed their system in their research group and collected manually labeled data from two regular users. The dataset consisted of 304 different tasks, about 4000 task switches as well as about 70000 information vectors. It was unbalanced since one user contributed 4 months and the other one 6 days of usage data. From the figures presented in [Shen *et al.*, 2009] an accuracy increase of about 10% in comparison to *TaskPredictor1* can be observed. There are no precision and recall values directly given for the best detection performance in [Shen, 2009; Shen *et al.*, 2009] but instead figures presenting the precision as a function of the recall by varying the confidence threshold required to make a prediction are given.

4.4.8 Task Switch Detection Approach by Nair et al. in 2005

Nair *et al.* [2005] designed a software systems that is able to identify the task switches of a user based on low level window manipulation data. This approach developed their own task switch detection algorithm an does not utilize machine learning algorithms. The underlying assumption of the approach is that there is a difference in the user behavior for the user interaction activities involved for switching a task in comparison to normal work. More specifically, they hypothesized that a task switch results in window operations with a different frequency than for normal work. Window operation considered by their approach are create, activate, maximize, minimize, hide, show and destroy a window which are automatically captured by their software systems via hooking into the window system events on the Microsoft Windows operating system. Based on the observed window operations the detection algorithm calculates (i) *the average time between window events for a given task*, (ii) *the simple moving average of the time between the last k interactions* and (iii) *the ratio between the overall task average and the moving average.* If this ratio is above a certain threshold the system signals a task switch to the user via a pop-up window. The system hereby can detect task switches without requiring an identification of the underlying task itself.

In order to evaluate their approach Nair *et al.* [2005] designed a two week study with six participants consisting of two professors, three graduate students, one IT professional. At the end of this two study they did questionnaires and interviews for gathering user feedback about their system. A dataset of 1033 task switches were collected. The system managed to correctly identify 422 task switches which are 40.86% percent of the whole dataset. On a singular participant basis the detection accuracy ranged from 21.33% to 94.74% with a mean of 50.12%. Nair *et al.* [2005] also reported the following three insights about their results: (i) their system was not able to detect task switches if the user switches between tasks very quickly, (ii) the use of instant messaging decreases the detection accuracy (*"The Instant Messaging Effect"*), because the software system detected instant messaging usage as a task switch while users felt that this was not a new task but just a side channel of information, and (iii) considering the last 7 windows together with the threshold limits between 0.67 and 1.5 for the ratio provide a good balance between accurate task switch detection and the false positive alarm rate. In future work they planned to investigate additional features for the detection algorithm, e.g., the web browser URL information, as well as allowing the user to manually specify windows to ignore. Dynamically adapting window sizes, threshold values and user pop-up notifications were part of their future work.

4.5 Discussion about existing Task Detection Approaches

Task detection is classically seen as a machine learning problem. Some approaches focused on unsupervised learning approaches [Cselle *et al.*, 2007; Kushmerick & Lau, 2005; Oliver *et al.*, 2006] whereas some explore the applicability of supervised learning algorithms to the task detection challenge [Dredze *et al.*, 2006; Granitzer *et al.*, 2009a; Kellar *et al.*, 2007; Lokaiczyk *et al.*, 2007; Lowd & Kushmerick, 2009; Mitchell *et al.*, 2006; Oliver *et al.*, 2006; Shen *et al.*, 2006, 2009]. In [Shen *et al.*, 2006] also a combination of two supervised learning algorithms was utilized for the task prediction, namely the Naïve Bayes algorithm for deciding if a new prediction should be calculated and the support vector machine algorithm for the prediction itself. A special case among the presented approaches is the task switch detection approach by Nair *et al.* [2005] because it does not use a machine learning algorithm. Instead they developed an algorithm based on the frequency of the observed window operations of a user.

The task detection approaches for the computer desktop presented in Section 4.4 are especially relevant for this dissertation and hence discussed in further detail regarding features, class instance construction, algorithms, attribute selection, evaluation methods as well as used datasets and the corresponding evaluation results. A summary of these aspects is visualized in Table 4.1.

Features: All the presented approaches utilize text-based features. Among the most famous text-based features are the "window title" feature, the "file/web page URL" feature and the "content of a resource" feature. The APOSDLE and DYONIPOS systems only concentrate on text-based features. SWISH is the only task detection approach next to the task switch detection approaches that use sequence-based features. SWISH refers to this feature as the "temporal closeness" feature, that takes into account the temporal sequence of application windows.

The Activity Extractor and the TaskPredictor 2 approaches utilize structure-based features next to text-based features for clustering emails to activities and for identifying a task switch respectively. Structure-based features are ones that include some kind of relation between entities, e.g., task with resource, or person with person. The Activity Extractor approach constructs a sender-recipient graph and performs social network analysis on this graph in order to refine the activity clusters detected based on text-based features. TaskPredictor 2 includes the "strength of association of the active resource with a task" as a feature. Both approach report an accuracy increase through including structure-based features.

For assigning time-segments of user activity data to projects the Smart Desktop approach employs next to text-based features also sequence-based ones. It uses "past project features", "salience features" and "shared salience features" for recognize the similarity between current and recent user activities. However, the number of features that have been used for task detection is very small and hence possibly limited in adapting to new domains. The evaluation results of the approaches seem to rely mainly on the good performance of the "window title" feature and hence may not adapt well to situations in which this specific feature has a low task discriminating power (see Section 7.6). None of the approaches utilize an ontology model for engineering features or combinations of features. This dissertation will argue later that features engineered from an ontology model will outperform classical text-based features of Dyonipos, SWISH and TaskPredictor 1 (see Chapter 7).

Class Instance Construction: The task detection approaches on the computer desktop disagree on how to build training instances for classifiers. The result of this disagreement is that they are difficult to compare among each other[6]. Lokaiczyk *et al.* [2007] suggests using a sliding window approach with a window size of 5 to 30 seconds. In SWISH, TaskPredictor 1 and TaskPredictor 2 intervals of 300 seconds, 60 seconds and 20-60 seconds were used respectively. In comparison to a time-based boundary Dyonipos utilized the boundaries of an event block[7] for constructing training instances. When comparing the task detection performance measures reported by the mentioned approaches someone has to be aware that the results origin from training instances of different "size". By varying the size of a training instance the accuracy of the task detection can vary as well. None of the approaches compare the detectability of tasks on the "task level". Task level here means that a single training instance for the classifier is constructed based on the usage data of a fully executed task. In order to study the discriminative power of user context features for specific tasks one training instance should be constructed based on the usage data of one executed task.

Algorithms: Different influencing factors for task detection were evaluated for (i) the type of classifier [Granitzer *et al.*, 2009a; Lokaiczyk *et al.*, 2007], (ii) the number of attributes [Granitzer *et al.*, 2009a; Lokaiczyk *et al.*, 2007; Shen, 2009] (iii) different user context feature combinations [Granitzer *et al.*, 2009a; Lokaiczyk *et al.*, 2007] as well as (iv) various feature preprocessing methods [Lokaiczyk *et al.*, 2007; Oliver *et al.*, 2006]. The classification approaches to task detection agree to use standard machine learning measure like accuracy, precision and recall for evaluating task detection performance. However, they do not agree on the classifier that works best. Although, there is a slight tendency that the Naïve Bayes and the linear Support Vector Machine algorithms work well.

Attribute Selection: Some of the presented task detection approaches utilize an attribute selection algorithm for identifying the most discriminative attributes on which the classifier should be trained. The approaches agree that *Information Gain* [Witten & Frank, 2005] is one of the attribute selection algorithms of choice [Granitzer *et al.*, 2009a; Lokaiczyk *et al.*, 2007; Shen, 2009]. These approaches agree that suggested number of attributes for training the classifiers should be between 100 to 300 attributes. This can also be observed in Table 4.1.

Evaluation Method: The evaluation method used for assess the task detection or task switch detection performance varies. While APOSDLE and DYONIPOS use stratified 10-fold

[6]A detail discussion on this topic is given in Section 7.2.2.
[7]An event block represents all interactions with a single resource.

cross-validation, TaskPredictor 1 and 2 use train/test set evaluation. The evaluations of the Activity Extractor and the task switch detection system of Nair *et al.* [2005] involve the user in evaluating the approach's performance. In the Activity Extractor approach user determine which of their activities was best represented by a sub-cluster and then assign a label for the emails in the cluster [Mitchell *et al.*, 2006]. Nair *et al.* [2005]'s system asks the user each time it detects a task switch to confirm if a task switch has happened or not.

Dataset: The datasets used to evaluate the task detection approaches on the computer desktop mentioned in this section are rather small, i.e., origin only from experiments with one or two users. Furthermore the datasets gathered from real world experiments (i) might have introduced noise in the user labeling [Shen *et al.*, 2006] or expert relabeling process [Granitzer *et al.*, 2009a], (ii) do not generalize well [Oliver *et al.*, 2006], (iii) were done by non-domain experts [Lokaiczyk *et al.*, 2007] or (iv) were unbalanced regarding training instances [Mitchell *et al.*, 2006]. Details about the dataset, the number of users involved and the evaluation results are given in Table 4.1. In order to study the influence of user context features and machine learning algorithms balanced datasets are preferable. Such datasets are difficult to gather in a real world setting because of the working behavior of people and the continuous changing requirements of a real-world setting. A possible solution to get to a well-balanced task detection dataset from several domain experts of a domain a controlled setting, i.e., a laboratory setting, is required. In Chapter 7 the design of three laboratory experiments with multiple users in two different domains are described that fulfill the mentioned requirements and can be utilized to study the influence of features for task detection.

Evaluation Results: The accuracy values reported by the task detection approaches range from 70% to 85%. A direct comparison of these accuracy values has to be done with caution since the datasets on which these results are achieved differed in terms of (i) number of classes, (ii) number of class instances, (iii) number of users, (iv) type of studied tasks and (v) the used evaluation method.

In comparison to the mentioned existing approaches the ontology-based feature engineering and task detection approach introduced in this dissertation utilizes an automatically populated user interaction context ontology (see Chapter 3) for studying and enhancing automatic task detection on the computer desktop (see Chapter 5 and Chapter 7).

4.6 Summary

This chapter gave an overview of the related work about task detection. Task detection in emails, in web browser and on the complete computer desktop were described and discussed. A special focus was put on presenting task detection approaches for detecting tasks on the whole computer desktop. A discussion about the related task detection research in respect to the research of this dissertation rounded off this chapter.

Approach	Detection			Features			Class Instance Construction	Algorithms											Attribute Selection	Evaluation Method				Dataset							
	Task	Task Switch	Project	Text-based	Sequence-based	Structure-based		ID3	IREP	J48	KNN	Logistic-Regression	Naïve Bayes	PLSI	SMO	SVM	Clustering (Unknown)	Other		Cross-Validation	Leave One Out	Train/Test Set	Other	Real-World(r)/Lab(l)	#User(s)	#Classes	#Instances	#Attributes	Accuracy	Precision	Recall
Activity Extractor	x			x		x	Email										x	x					x[8]	r	1	23	2822	?	73%		
	x			x		x	Email										x	x					x[8]	r	1	8	420	?	80%		
	x			x		x	Email										x	x					x[8]	r	1	11	617	?	83%		
APOSDLE	x			x			30sec Sequence	x	x				x			x		x	x	x				l	1	5	591	200	85%	?	?
Dyonipos	x			x			Event Block				x		x			x			x	x				r	1	5	?	156	73.6%	0.91	0.75
	x			x			Event Block				x		x			x			x	x				r	1	4	?	188	76.4%	0.90	0.74
	x			x			Task[9]			x	x		x			x			x	x				l	14	5	218	300	83.5%	0.95	0.85
SWISH	x			x	x		5min Sequence							x			x					x		r	1	5	?	?	70%	0.49	0.72
Task Predictor 1	x			x			WDS[10]						x			x			x			x		r	1	96	5894	200	?	0.8	?
	x			x			WDS[10]						x			x			x			x		r	1	81	4151	200	?	0.8	?
Nair et al. 2005		x			x		Interaction Sequence[13]											x					x[11]	r	1	54/47[12]			94.7%		
		x			x													x					x[11]	r	1	43/76[12]			56.6%		
		x			x													x					x[11]	r	1	123/367[12]			33.5%		
		x			x													x					x[11]	r	1	117/217[12]			53.9%		
		x			x													x					x[11]	r	1	37/91[12]			40.7%		
		x			x													x					x[11]	r	1	48/225[12]			21.3%		
Task Predictor 2		x		x		x	Info-Vector[14]											x	?			x		r	1	3657	65049	?	?	?[15]	?[15]
		x		x		x	Info-Vector[14]											x	?			x		r	1	359	3641	?	?	?[15]	?[15]
Smart Desktop			x	x	x		1 User Interaction					x	x			x		x	x		x			r	4	26-40	465-5036	?	?	?	?

Table 4.1: This table gives an overview of the task detection approaches for the computer desktop presented in this chapter. The "x" and the " " signal that the criterion is met or not respectively. The question mark "?" indicates that no information about the criterion was available.

[8] User evaluation methodology: the emails were clustered to activities and compared to ground truth labels from each user. The users examined each cluster and *"determined which of their activities was best represented by the sub-cluster, then assigned this as true label for emails in the cluster"* [Mitchell *et al.*, 2006].

[9] These task detection results stem from the evaluations described in Section 7.4.2. They have also been published as part of [Granitzer *et al.*, 2009a].

[10] The window document sequence (WDS) is *"the maximal contiguous segment of time in which a particular window has focus and the name of the document in that window does not change.* [Shen *et al.*, 2006]

[11] User evaluation methodology: every time the system detects a task switch the user is asked to confirm if a task switch has happened or not.

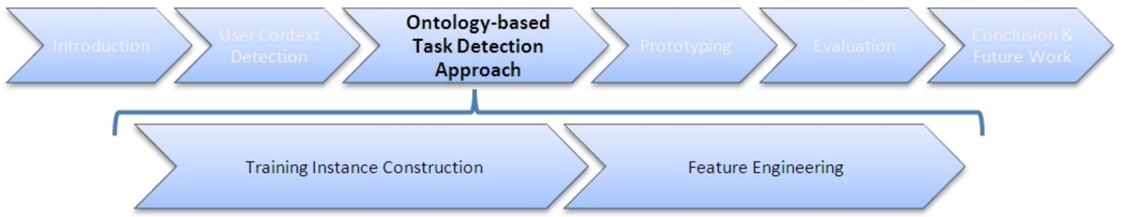
[12] The first number indicates how many task switches have correctly been detected, i.e., confirmed by the user. The second number indicates the total number of switches detected by the system.

[13] The Nair *et al.* [2005]'s detection algorithm is based on the observed window operations and calculates (i) the average time between window events for a given task, (ii) the simple moving average of the time between the last k interactions and (iii) the ratio between the overall task average and the moving average to make a prediction.

[14] An information vector is constructed every 20-60 seconds and based on (i) task-specific features and (ii) switch-specific features [Shen, 2009].

[15] There are no precision and recall values directly given for the best detection performance in [Shen, 2009; Shen *et al.*, 2009] but instead figures presenting the precision as a function of the recall by varying the confidence threshold required to make a prediction are given.

5 Ontology-Based Task Detection Approach



The purpose of this part of the thesis is to introduce the *ontology-based task detection approach* proposed in this dissertation research. It is a novel approach to task detection that combines semantic technologies with machine learning in order to improve task detection. This chapter elaborates on the utilization of the ontology-based user interaction context ontology (UICO) presented in Section 3.3 for enhancing the performance of the machine learning problem automatic task detection.

In this approach the UICO is utilized to engineer novel features and feature combinations for doing automatic task detection. 50 features classifiable into 5 feature categories are extracted from the UICO. On behalf of the *ontology-based task detection pipeline* this chapter explains the transformation of the user interaction context stored in the UICO to (training) class instances of machine learning algorithms (*training instance construction*) in Section 5.2 and elaborates on the construction and preprocessing of each feature (*feature engineering*) in Section 5.3.

5.1 Introduction

A classical approach to automatic task detection is to model it as a machine learning problem, more specifically a classification problem. This approach has been used for recognizing web based tasks [Gutschmidt *et al.*, 2008; Kellar & Watters, 2006], tasks within emails [Dredze *et al.*, 2006; Kushmerick & Lau, 2005] or from the complete user's desktop [Granitzer *et al.*, 2008; Lokaiczyk *et al.*, 2007; Oliver *et al.*, 2006; Shen *et al.*, 2007]. All these approaches are based on the following steps [Rath *et al.*, 2009d]: First, the contextual attention metadata has to be captured by context sensors. Second, it has to be chosen which parts of the data (features) are used for building the training instances for the machine learning part. Since these features can not directly be used as inputs for machine learning algorithms the third step is to transform the context features into attributes [Witten & Frank, 2005]. This transformation may also include data preprocessing operations. An example for the famous `window title` feature is the summarization of the words appearing in the window title into a *"bag of words"* then transformed into word vector format. For text-based features, preprocessing steps like removing stopwords [Granitzer *et al.*, 2008; Lokaiczyk *et al.*, 2007; Shen *et al.*, 2007] or application specific terms [Oliver *et al.*, 2006], are applied. The fourth step is to apply attribute selection algorithms [Granitzer *et al.*, 2008; Shen *et al.*, 2006] to select the most important attributes for the learning algorithms (optional). The fifth step is the training and testing of the learned model. A widely used method for testing the task detection performance of the mentioned approaches was *stratified 10-fold cross-validation* [Witten & Frank, 2005].

5.2 Training Instance Construction

Constructing training instances for the machine learning algorithms is done on the task level. This means that each task represents a training instance for a specific class to be learned. A class corresponds to a specific task model. Having multiple task models hence results in a *multi-class* classification problem. In Figure 5.1 the conceptual and the ontology view for constructing a training instance for a class is shown. A training instance for a class is built from features and feature combinations of the user interaction context of an instance of a `Task` concept. A sample task instance used in the training instance construction process is displayed in Figure 5.2. The process of constructing features representing a task instance and transforming them into attributes that can be used to train machine learning algorithms is referred to as *feature engineering.*

The 50 context features engineered for constructing the training instances can be grouped in six categories [Rath *et al.*, 2009a]: (i) *ontology structure*, (ii) *content*, (iii) *application* (iv) *resource*, (v) *action* and (vi) *switching sequences.* The *ontology structure category* contains features representing the number of instances of concepts and the number of datatype and objecttype relations used per task. The *content category* consists of the content of task-related resources, the content in focus and the text input of the user. The *application category* contains the classical *window title* feature [Granitzer *et al.*, 2008; Lokaiczyk *et al.*, 2007; Oliver *et al.*, 2006; Shen & Dietterich, 2007], the application name feature [Granitzer *et al.*, 2008] and the newly introduced graphical user interface elements features (accessibility objects [Microsoft, 2009]). The *resource category* includes the complete contents and URIs (URLs) [Shen & Dietterich, 2007] of the used, referenced and included resources, as well as a feature that combines all the metadata about the used resources in a 'bag of words'. The *action category* represents the user interactions and contains features about the interactions with applications [Granitzer *et al.*, 2008], resources types, resources, key input types (navigational keys, letters, numbers), the number of events and event blocks, the duration of the event blocks, and the time intervals between event blocks. The *switching sequences category* comprises features about switches between applications, resources as well as event types and resource types. Besides the ontology structure category, there are also new ontology-based features in the action and resource categories. These new features are constructed based on combinations of concepts with concepts as well as concepts with concept instances. An example for the first one is the combination of the `EventType` with the sub-concepts

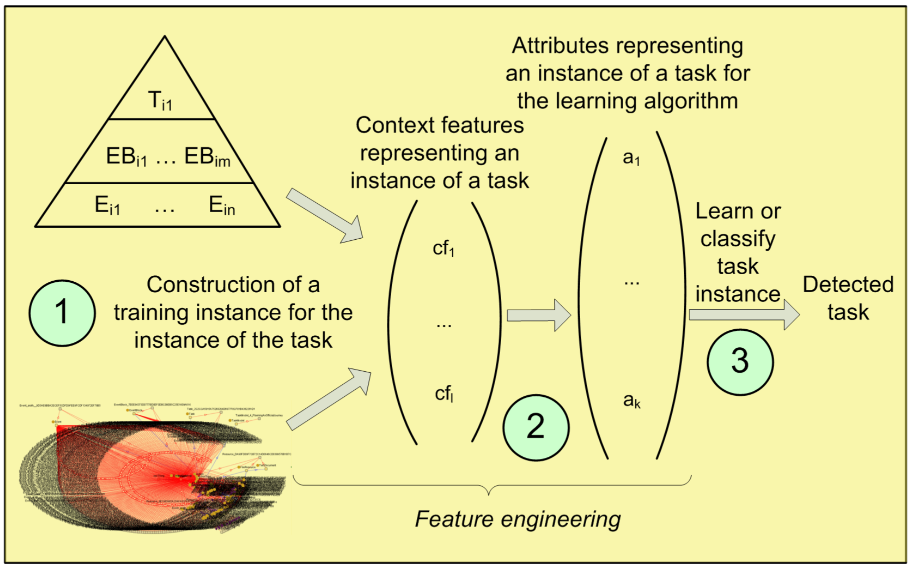

Attributes representing an instance of a task for the learning algorithm
T_{i1}
EB_{i1} … EB_{im}
E_{i1} … E_{in}
Context features representing an instance of a task
cf_1
…
cf_l
a_1
…
a_k
Learn or classify task instance
Detected task
1
Construction of a training instance for the instance of the task
2
3
Feature engineering

Figure 5.1: This figure visualizes the conceptual and the ontology model for constructing a training instance based on an instance of the `Task` *concept. An instance of the* `TaskModel` *concept serves as the class label for the machine learning algorithms. (1) shows the two populated models, (2) the feature engineering process for getting a representation of a task as a training instance for the feature selection and learning algorithms as well as (3) the detection of the task based on the learned classification model.*

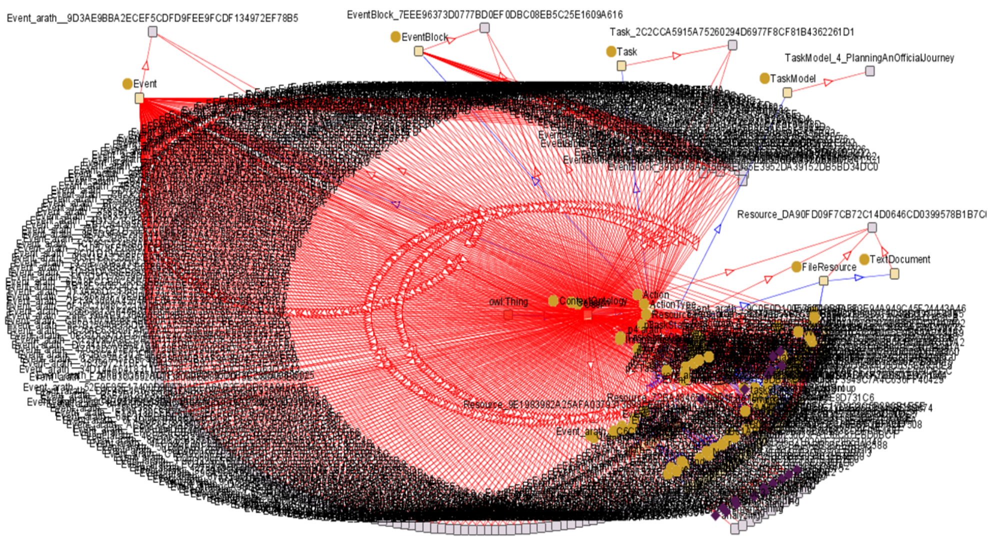

Event_arath__9D3AE9BBA2ECEF5CDFD9FEE9FCDF134972EF78B5
EventBlock_7EEE96373D0777BD0EF0DBC08EB5C25E1609A616
EventBlock
Task_2C2CCA5915A75260294D6977F8CF81B4362261D1
Task
TaskModel_4_PlanningAnOfficialJourney
TaskModel
Event
Resource_DA90FD09F7CB72C14D0646CD0399578B1B7C
FileResource
TextDocument
owl:Thing
ContextOntology
Action
ActionType

Figure 5.2: This figure shows the populated UICO of a task instance of a real task instance belonging to the task model "Planning a official journey" recorded during the task experiment described in Section 7.4. The graph visualization of the populated ontology was done in the Protégé ontology modeling tool [Protégé, 2009].

of the `Resource` concept. For the second one the combination of the `EventType` with an instance of the `TextDocument` concept is an example.

A detailed explanation of the six feature categories and the included features together with the preprocessing steps applied is given in the next Section 5.3.

5.3 Feature Engineering

50 features are engineered based on the concepts and relations of the user interaction context ontology (UICO). These features are detailed in this section. Furthermore preprocessing steps and the order in which they are performed are explained and listed for each feature separately. A complete listing of all features and their corresponding feature category is given in Table 5.1.

Nr.	Feature	Nr.	Feature	Nr.	Feature
	Action Category	20	semantic type	37	objecttype properties
1	EB duration	21	action type of event		**Resource Category**
2	res. types interact.		**Application Category**	38	used res. content
3	control input keys	22	acc. obj. name	39	resource content
4	nr. of E/EB	23	acc. obj. description	40	used res. metadata
5	included res. interact.	24	acc. obj. role	41	referenced resources
6	referenced res. interact.	25	acc. obj. role des.	42	used resources
7	res. interact.	26	acc. obj. value	43	included res. content
8	letter input keys	27	acc. obj. help	44	included res.
9	task duration	28	acc. obj. help topic	45	referenced res. content
10	applications interact.	29	application name		**Switching Sequence Category**
11	navigation input keys	30	window title	46	app. switch seq.
12	EB res. interact.	31	raw event source	47	E type switch seq.
13	mean EB duration		**Content Category**	48	E level res. switch seq.
14	mean time between EBs	32	content of EB	49	E&EB res. switch seq.
15	used res. interact.	33	content in focus	50	E&EB res. type switch seq.
16	action element of event	34	user input		
17	median time between EBs		**Ontology Category**		
18	number input keys	35	concept instances		
19	median EB duration	36	datatype properties		

Table 5.1: Overview of all features and their corresponding feature category engineered based on the user interaction context ontology (UICO).

5.3.1 Standard Text Preprocessing Steps

For standard text-based features the following preprocessing steps are applied in the following order:

1. fixing the German character encoding[1], e.g., `ü`, `ä` and `ö`.
2. remove end of line characters
3. remove multiple blank characters
4. remove `null` strings
5. remove markups, e.g.,`\&lg` and `![CDATA`
6. remove bracket characters, e.g., `(`, `\{` and `[`

[1]This step is required for the data in the first experiment since there was an encoding issue with special German characters called “Umlaute”.

7. remove URL encoded characters
8. remove all characters but letters
9. remove German and English stopwords
10. remove words shorter than three characters
11. multiple blanks
12. transform all strings to lower case
13. apply the *porter stemmer* [Porter, 1997] from the *Snowball* software package [Porter, 2009]
14. transform all strings to a word vector, where each dimension of the vector represents a word appearing in the string and the value the number of occupancies of this word
15. *tf/idf* computation for the values of the word vector according to the Weka *tf/idf* capabilities [Witten & Frank, 2005]

5.3.2 Action Feature Category

The feature category *action category* includes features that origin from interactions of the user with input devices, with resources and and with applications. This category includes 21 different features that are listed in Table 5.2. The construction procedure and the preprocessing steps applied to each feature are described in further detail in this section.

5.3.2.1 Feature 1: `EB duration`

The *event block duration* (*EB duration*) feature holds the information about the duration of interactions with a certain resource. The feature consists of nominal attributes that represent the 13 intervals of event block durations. The chosen intervals including examples of interactions for each interval are displayed in Table 5.3. Manual inspection of multiple event blocks from the data generated during the laboratory studies led to the boundaries of the intervals.

Applied Preprocessing Steps:

1. calculation of the event block duration and the corresponding interval
2. transforming the "interval category strings" into a word vector, i.e., for each category a new attribute is built and the number of event block durations that fall in this category is assigned to the attribute's value

5.3.2.2 Feature 2: `res. types interact.`

The *resource types interaction* (*res. types interact.*) feature represents the number of different types of combinations of the `EventType` and the `ResourceType` concept. These combinations can be interpreted as how often the user interacted in a specific way with a specific resource type. An example is a user who clicks on the search button of an online search engine four times. These interactions result into the event type `WebSearch` and the resource types `OnlineResource`. For each combination a new attribute is built, e.g., attribute `WebSearch_OnlineResource=4`.

Nr.	Feature	Brief Feature Description
1	EB duration	the duration of event blocks in seconds
2	res. types interact.	the number of combinations of `EventType` and `ResourceType`
3	control input keys	the number of control keys pressed by the user
4	nr. of E/EB	the number of events per event block
5	included res. interact.	the URIs of resources that are included in used resources
6	referenced res. interact.	the URIs of resources that are included in used resources
7	res. interact.	the number of combinations of `EventType` instances and the URI of the used resource
8	letter input keys	the number of letters in the keyboard input of the user
9	task duration	the duration of the task in seconds
10	applications interact.	the combinations of `EventType` instances and the application name
11	navigation input keys	the number navigation keys (keyboard and mouse) pressed by the user
12	EB res. interact.	the number of events of the event block that act on the used resource
13	mean EB duration	the mean duration of event blocks of a task in seconds
14	mean time between EBs	the mean of the duration between two event blocks of a task
15	used res. interact.	the URI of the used resource
16	action element of event	the text-based action element of an event sent by the sensor
17	median time between EBs	the median of the duration between two event blocks of a task
18	number input keys	the number of numbers in the input of the user
19	median EB duration	the median duration of event blocks of a task
20	semantic type	the automatically computed textual description of the user's interaction
21	action type of event	the text-based action type of an event sent by the sensor

Table 5.2: Overview of all features of the action feature category.

Interval	**Example**
<1s	a single interaction with an application or a resource
2s-3s	switch to an application and copying some text content
4s-10s	looking up a word in an online dictionary
11s-19s	writing an instant message
20s-50s	reading a short email
51s-100s	reading an online article
101s-200s	rephrasing a short text paragraph
201s-500s	writing an email
501s-750s	writing a long email
751s-1000s	calculating a schedule in a spreadsheet application
1001s-2000s	rewriting a text paragraph of a document
2001s-5000s	drawing a figure
>5000s	reading a document

Table 5.3: Intervals of the `EB duration` feature.

Applied Preprocessing Steps:

1. transforming the combinations into a word vector, i.e., for each combination a new attribute is built and the number of occurrences of each combination is assigned to the value of the attribute

5.3.2.3 Feature 3: `control input keys`

The *control input keys* feature counts the number of `control keys` the user has pressed during the execution of a task. Following keys are considered as control keys: `shift key`, `return key`, `F1-F* keys`, `tabulator key` and `numpad key`.

Applied Preprocessing Steps:

1. discretizing the number of pressed `control keys` by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.4 Feature 4: `nr. of E/EB`

The *number of events per event block* (*nr. of E/EB*) feature holds the information about the number of interactions with a certain resource. The feature consists of nominal attributes that represent the 12 intervals. The chosen intervals including examples of interactions for each interval are displayed in Table 5.4. Manual inspection of multiple event blocks from the data generated during the laboratory studies led to the boundaries of the intervals.

Applied Preprocessing Steps:

1. calculation of the number of events of an event block and the corresponding interval

2. transforming the intervals into a word vector, i.e., for each interval a new attribute is built and the number of events of an event block instance is assigned to the attribute's value

Interval	**Example**
1	web browsing, Microsoft Windows Explorer navigation
2-3	selecting and copying some text content
4-19	writing an instant message
20-50	writing a short email
51-100	reading a blog entry
101-200	editing a text paragraph
201-500	creating a single presentation slide
501-750	writing an email
751-1000	writing a paragraph in a document
1001-2000	writing a one page article
2001-5000	writing a long text document
>5000	writing and formatting a long text

Table 5.4: Intervals of the `nr. of E/EB` feature.

5.3.2.5 Feature 5: `included res. interact.`

The *included resource interaction* (*included res. interact.*) feature counts the number of interaction with a resources that *includes* another resource. An example is when a user interacts with a specific part of a web page multiple times. The web page and the part of the web page are represented as an instance of the `OnlineResource` concept and the relation `includesResource` exists between the two instances. For each included resource a new numeric attribute is created which holds the number of interaction as its attributes value.

Applied Preprocessing Steps:

1. retrieving the URIs of the included resources and storing them in a single string
2. transforming the URI string into a word vector, i.e., for each included resource a new attribute is built and the number of interactions is assigned to the attribute's value

5.3.2.6 Feature 6: `referenced res. interact.`

The *referenced resource interaction* (*referenced res. interact.*) feature counts the number of interaction with a resources that *references* another resource. An example is multiple user interactions with an email that includes a link to a web page. The email and the link are represented as instances of the `EmailResource` and `OnlineResource` respectively as well as connected with the `referencesResource` relation. For each referenced resource a new numeric attribute is created which holds the number of interactions as its attribute's value.

Applied Preprocessing Steps:

1. retrieving the URIs of the referenced resources and storing them in a single string
2. transforming the URI string into a word vector, i.e., for each referenced resource a new attribute is built and the number of interactions is assigned to the attribute's value

5.3.2.7 Feature 7: `res. interact.`

The *resource interaction* (*res. interact.*) feature counts the number of different type of combinations of the `EventType` and an instance of a `Resource` concept. These combinations can be interpreted as how often the user interacted in a specific way with a specific resource. An example is a user entering 20 characters into a text document. These interactions result into the event type `Push` and an instance of the resource type `TextDocument`. For each new event type and resource instance a new numeric attribute is built. In this example the attribute `Push_<URI of the text document>` with value 20 is created.

Applied Preprocessing Steps:

1. transforming the combinations into a word vector, i.e., for each combination a new attribute is built and the number of occurrences of the combination is assigned to the value of the attribute.

5.3.2.8 Feature 8: `letter input keys`

The *letter input keys* feature counts the number of *letter keys* the user has pressed. Following keys are considered as *letter keys* described as a regular expression: `[a-zA-ZäöüÄÖÜß]`.

Applied Preprocessing Steps:

1. discretizing the number of pressed *letter keys* by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.9 Feature 9: `task duration`

The *task duration* feature stands for the length of a task measured in seconds and is represented by a numeric attribute.

Applied Preprocessing Steps:

1. discretizing the number of seconds of the task duration by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.10 Feature 10: `applications interact.`

The *applications interactions* (*applications interact.*) feature counts the number of different types of combinations of the `EventType` concept and the name of the application. These combinations can be interpreted as how often the user interacted in a specific way with a specific application. An example is a user navigating through her folder hierarchy in the Microsoft Windows Explorer with 10 mouse clicks. These interactions result into the event type `Navigate` and the application name `explorer`. For each new event type and application name a new numeric attribute is built, e.g., attribute `Navigate_explorer` with value 10.

Applied Preprocessing Steps:

1. transforming the combinations into a word vector, i.e., for each combination a new attribute is built and the number of occurrences of the combination is assigned to the value of the attribute

5.3.2.11 Feature 11: `navigation input keys`

The *navigation input keys* feature counts the number of *navigation keys* the user has pressed. Following keys are considered as *navigation keys*: keyboard up/down/left/right/home/prior/end keys, the F1-F* keys[2] as well as left and right mouse buttons.

Applied Preprocessing Steps:

1. discretizing the number of pressed *navigation keys* by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.12 Feature 12: `EB res. interact.`

The *event block resource interactions* (*EB res. interact.*) feature counts the number of events of an event block that has a `isActionOn` relation to a specific instance of an used resource. An example is a user who writes an email in Microsoft Outlook. The events about the user's interactions with the email are associated with the same event block but not all events include the information that a specific user interaction has happened on this email because of the utilization of multiple sensors (see Section 3.4.1). For each used resource a new numeric attribute is created that stores the number of events of the event block as its attribute value.

Applied Preprocessing Steps:

1. retrieving the URIs of the used resource of the event block and storing them in a single string
2. transforming the URI string into a word vector, i.e., for each used resource a new attribute is built and the number of events of the event block is assigned to the attribute's value

5.3.2.13 Feature 13: `mean EB duration`

The *mean event block duration* (*mean EB duration*) feature calculates the mean length of event block durations of a task measured in seconds. It is represented by a numeric attribute.

[2]The F1-F* keys also count as *control keys* since the final interpretation of the action behind these keys is application and user configuration dependent.

Applied Preprocessing Steps:

1. discretizing the mean number of seconds of the duration of event blocks of a task by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.14 Feature 14: `mean time between EBs`

The *mean time between event blocks* (*mean time between EBs*) feature calculates the mean duration between the end time and the start time of two consecutive event blocks belonging to the same task. From a more general perspective, this feature can be seen as representing the amount of time required by a user to switch from one resource to another one. This duration is measured in seconds. The feature is represented by a numeric attribute.

Applied Preprocessing Steps:

1. discretizing the mean duration between the end time and the start time of two consecutive event blocks of a task by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.15 Feature 15: `used res. interact.`

The *used resource interactions* (*used res. interact.*) feature counts the number of interactions with a specific used resources. Examples are multiple user interactions with a web page. The web page is represented as an instance of the `OnlineResource` concept. For each used resource a new numeric attribute is created which holds the number of interactions as its attribute's value. An interaction here is an event that has an `isActionOn` relation with this resource.

Applied Preprocessing Steps:

1. retrieving the URIs of the used resources of all events of a task and storing them in a single string
2. transforming the URI string into a word vector, i.e., for each used resource a new attribute is built and the number of interactions (events) is assigned to the attribute's value

5.3.2.16 Feature 16: `action element of event`

The *action element of event* feature represents a concatenation of all `hasActionElement` properties of a sensor event. An example would be an event from the Novell GroupWise application when the user opens an email. The resulting event from the *DyGobs* sensor includes the string "email" as its `hasActionElement` value.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.2.17 Feature 17: `median time between EBs`

The *medium time between event blocks* (*median time between EBs*) feature calculates the median length of event block durations of a task measured in seconds and is represented by a numeric attribute.

<u>Applied Preprocessing Steps:</u>

1. discretizing the median number of seconds of the duration of event blocks of a task by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.18 Feature 18: `number input keys`

The *number input keys* feature counts the number of `number keys` the user has pressed. Following keys are considered as `number keys` described as a regular expression: `[0-9]`.

<u>Applied Preprocessing Steps:</u>

1. discretizing the number of pressed `number keys` by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.19 Feature 18: `median EB duration`

The *medium event block duration* (*median EB duration*) feature calculates the median length of event block durations of a task measured in seconds and is represented by a numeric attribute.

<u>Applied Preprocessing Steps:</u>

1. discretizing the median number of seconds of the duration of event blocks of a task by utilizing the `PKIDiscretize` filter of the Weka software package, which transforms the numeric attribute values into intervals.

5.3.2.20 Feature 20: `semantic type`

The *semantic type* stores a rule-based generated type describing the event in natural language text. Example values indicate whether a document is (i) read or (ii) edited by the user, or whether the state is (iii) unknown. The assignment of a semantic type is application dependent. Only MS Word, MS PowerPoint and MS Excel sensors are capable of allotting the semantic types (i) reading or (ii) writing. The discrimination is based on simple rules based on the keyboard input, the application and the interacted resource. The feature represents a concatenation of all the `hasSemanticType` relations of an event.

<u>Applied Preprocessing Steps:</u> Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.2.21 Feature 21: `action type of event`

The *action type of event* feature represents a concatenation of all `hasActionType` properties of a sensor event. An example is an event from the Novell GroupWise application when the user opens an email. The resulting event from the *DyGobs* sensor includes the string “open” as its `hasActionType` value.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3 Application Feature Category

The feature category *application category* includes features originating from user interactions with operating system elements and with application controls. This category includes 10 different features that are listed in Table 5.5. The construction procedure and the preprocessing steps applied for each feature are described in further detail in this section.

Nr.	**Feature**	**Brief Feature Description**
22	acc. obj. name	the name of the accessibility object
23	acc. obj. description	the text describing the visual appearance of the accessibility object
24	acc. obj. role	the role of the accessibility object
25	acc. obj. role des.	the description of the role of the accessibility object
26	acc. obj. value	the value of the accessibility object
27	acc. obj. help	the *Help* property string of an accessibility object
28	acc. obj. help topic	full path of the *WinHelp*[3] file associated with the accessibility object and the identifier of the appropriate topic within that file
29	application name	the name of the application in focus
30	window title	the title of the window in focus
31	raw event source	the unmodified source of the event sent by the sensors

Table 5.5: Overview of all features of the application feature category.

5.3.3.1 Feature 22: `acc. obj. name`

The *accessibility objects name* (*acc. obj. name*))feature represents a concatenation of all `hasAccName` relations of an event. An example is an event originating from a mouse click on the close button of a command window on a German Microsoft Windows operating system. The `hasAccName` value in this example is “Schließen”[4]

[4] “Schließen” in English is “close”.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3.2 Feature 23: `acc. obj. description`

The *accessibility objects description* (*acc. obj. description*) feature represents a concatenation of all `hasAccDescription` relations of an event. An example is an event originating from a mouse click on the close button of a command window on a German Microsoft Windows operating system. The `hasAccDescription` value in this example is "Schließt das Fenster"[5].

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3.3 Feature 24: `acc. obj. role`

The *accessibility objects role* (*acc. obj. role*) feature represents a concatenation of all `hasAccRole` relations of an event. An example is an event originating from a mouse click on the close button of a command window on a German Microsoft Windows operating system. The `hasAccRole` value in this example is "43" which is an internal number for the role of the accessibility object.

Applied Preprocessing Steps:

1. retrieving the role numbers of the accessibility objects of all events of a task and storing them in a single string
2. transforming the role number into a word vector, i.e., for each role number string a new attribute is built and the number of occuracies is assigned to the attribute's value

5.3.3.4 Feature 25: `acc. obj. role des.`

The *accessibility objects role description* (*acc. obj. role des.*) feature represents a concatenation of all `hasAccRoleDescription` relations of an event. An example is an event originating from a mouse click on the close button of a command window on a German Microsoft Windows operating system. The `hasAccRoleDescription` value in this example is "Schaltfläche"[6].

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3.5 Feature 26: `acc. obj. value`

The *accessibility objects value* (*acc. obj. value*) feature represents a concatenation of all `hasAccValue` relations of an event. These relations origin from the values of the accessiblity objects the user has interacted with. An example is an event originating from a mouse click on

[5] "Schließt das Fenster." in English is "Closes the window.".
[6] "Schaltfläche" in English means "button".

the filled in subject line of an email message in Microsoft Outlook. The `acc. obj. value` value in this example is the text content of this subject line.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3.6 Feature 27: `acc. obj. help`

The *accessibility objects help* (*acc. obj. help*) feature represents a concatenation of all `hasAccHelp` relations of an event. These relations origin from the help text of the accessibility objects the user has interacted with. An example is an event originating from a mouse click on the tab of the editor pane. The `acc. obj. help` value is the text content of this tab, i.e., the name of file opened in this tab for editing.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3.7 Feature 28: `acc. obj. help topic`

The *accessibility objects help topic* (*acc. obj. help topic*) feature represents a concatenation of all `hasAccHelpTopic` relations of an event. These relations origin from the help text of an accessibility objects the user has interacted with. This is the full path of the *WinHelp*[7] file associated with the accessibility object and the identifier of the appropriate topic within that file.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3.8 Feature 29: `application name`

The *application name* feature represents a concatenation of all `hasApplicationName` relations of an event. These relations hold the name of the application the user has interacted with. An example is a mouse click of a user within the Mozilla Firefox browser window. The value of the `hasApplicationName` in this example is "firefox".

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3.9 Feature 30: `window title`

The *window title* feature represents a concatenation of all `hasWindowTitle` relations of the events. This feature contains the title of the window of the application the user has interacted with. Depending on the application the window title may (e.g. Microsoft Word) or may not

[7] WinHelp: `http://msdn.microsoft.com/en-us/library/bb762267%28VS.85%29.aspx`

include (e.g. Novel Groupwise's new email window) the name of the application.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.3.10 Feature 31: `raw event source`

The *raw event source* feature represents a concatenation of the `hasEventSource` relation of the events. This feature contains the unmodified source of the event sent by the sensors.

Applied Preprocessing Steps:

1. remove end of line characters
2. remove multiple blank characters
3. remove all numbers
4. remove German and English stopwords
5. remove words shorter than three characters
6. transform all strings to lower case
7. remove multiple blanks
8. transform all strings to a word vector, i.e., for each string a new attribute is built and the number of occuracies of this string is assigned to the attribute's value

5.3.4 Content Feature Category

The feature category *content category* includes features that origin from interactions of the user with viewed and input text. This category consists of 3 different features that are listed in Table 5.6. The construction procedure and the preprocessing steps applied to each feature are described in further detail in this section.

Nr.	Feature	Brief Feature Description
32	content of EB	the automatically aggregated text content the user has interacted with
33	content in focus	the concatenated text content of the windows in focus
34	user input	the automatically aggregated keyboard input of the user

Table 5.6: Overview of all features of the content feature category.

5.3.4.1 Feature 32: `content of EB`

The *content of event block* (*content of EB*) feature represents a concatenation of all `hasContent` relations of an event block. This features consists of the automatically aggregated text content based on the `hasContent` relation of all events of the event block. It can also be interpreted that it stores the content of the current window, which can be the text on the current Microsoft PowerPoint slide or the content of the website the user is viewing. The raw text content includes also special characters, markup and end-of-line characters.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.4.2 Feature 33: `content in focus`

The *content in focus* feature is a concatenation of all `hasContent` relations of all events of an event block. The `hasContent` relation holds the text content of the window in focus.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.4.3 Feature 34: `user input`

The *user input* feature is a concatenation of all `hasPreprocessedInput` relations of all event blocks. It holds the input of the user after automatic preprocessing steps have been applied to the values of the `hasInput` relation of all events of the event block. An example are events with the following values for the `hasInput` relation: "`h`", "`e`", "`l`","`l`", "`i`", "`Back`", "`o`". These values are automatically preprocessed such that the `hasPreprocessedInput` relation of the event block only contains the word "`hello`".

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.5 Ontology Feature Category

The feature category *ontology category* contains features representing the number of instances of concepts as well as the number of datatype and objecttype relations. This category consists of 3 different features that are listed in Table 5.7. The construction procedure and the preprocessing steps applied to each feature are described in further detail in this section.

Nr.	**Feature**	**Brief Feature Description**
35	concept instances	the number of instances of a specific concept
36	datatype properties	the number of datatype properties used between concept instances and literals
37	objecttype properties	the number of objecttype properties used between concept instances

Table 5.7: Overview of all features and their corresponding feature category constructed from the concepts and relations of the User Interaction Context Ontology (UICO).

5.3.5.1 Feature 35: `concept instances`

The *concept instances* feature represents the number of instances for each concept of the UICO (see Section 3.3). For each concept a new attribute is constructed which holds the number of instances of this concept.

Applied Preprocessing Steps: No preprocessing steps are applied.

5.3.5.2 Feature 36: `datatype properties`

The *datatype properties* feature holds the number of datatype properties used by instances of the UICO (see Section 3.3). For each datatype property a new attribute is constructed which holds the number of times a datatype porperty is used.

Applied Preprocessing Steps: No preprocessing steps are applied.

5.3.5.3 Feature 37: `objecttype properties`

The *objecttype properties* feature holds the number of objecttype properties used by instances of the UICO (see Section 3.3). For each objecttype property a new attribute is constructed which holds the number of times a objecttype property is used.

Applied Preprocessing Steps: No preprocessing steps are applied.

5.3.6 Resource Feature Category

The feature category *resource category* contains features that include the complete content and URIs of the used, referenced and included resources, as well as a features that combine all the metadata about the used resources. This category consists of 8 different features that are listed in Table 5.8. The construction procedure and the preprocessing steps applied to each feature are described in further detail in this section.

5.3.6.1 Feature 38: `used res. content`

The *used res. content* feature represents the content of the used resources the user has directly interacted with. Examples are the content of a Microsoft PowerPoint slide, an email or a web page.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.6.2 Feature 39: `resource content`

The *resource content* feature concatenates the text content of all used, included and referenced resources. An example is an email that contains a link to an online resource as well as a copied & pasted text snippet from text document. The text content of the email, the online resource and the text document is concatenated as a string for this feature.

Nr.	Feature	Brief Feature Description
38	used res. content	the content of the resource the user has directly interacted with (used resource)
39	resource content	the text content of all used, included and referenced resources
40	used res. metadata	the concatenated text of all data and metadata about an used resource
41	referenced resources	the URIs of the referenced resources in the used resources
42	used resources	the URIs of the used resources
43	included res. content	the content of the included resources in an used resource
44	included res.	the URIs of the included resources
45	referenced res. content	the content of the referenced resources in an used resource

Table 5.8: Overview of all features of the resource feature category.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.6.3 Feature 40: `used res. metadata`

The *used resource metadata* (*used res. metadata*) feature concatenates the text content of all data and metadata about an used resource. An example for this feature is a concatenated string based on the content of a text document, the author of this document and the folder in which the document is stored.

Applied Preprocessing Steps:

1. fixing the German character encoding, e.g., `ü`, `ä` and `ö`.
2. remove end of line characters
3. remove multiple blank characters
4. remove `null` strings
5. remove markups, e.g.,`\&lg` and `![CDATA`
6. remove bracket characters, e.g., `(`, `\{` and `[`
7. remove URL encoded characters
8. remove remove everything except letters and German special characters
9. remove German and English stopwords
10. remove words shorter than three characters
11. multiple blanks
12. transform all strings to lower case

13. transform all strings to a word vector
14. compute *tf/idf* values for the vector entries

5.3.6.4 Feature 41: `referenced resources`

The *referenced resources* feature represents the URIs of the referenced resources in the used resources. An example is a text document that contains an email address. The text document and the email address result in instances of the `TextDocument` and `PersonResource` concepts respectively with the relation `isReferencedResourceFrom` from the `TextDocument` instance to the `PersonResource` instance. For each referenced resource a new attribute is created.

Applied Preprocessing Steps:

1. retrieving the URIs of the referenced resources and storing them in a single string
2. transforming the URI string into a word vector, i.e., for each referenced resource a new attribute is built and the number of interactions is assigned to the attribute's value

5.3.6.5 Feature 42: `used resources`

The *used resources* feature represents the URIs of the used resources. For each used resource a new attribute is created. An example are the interactions with a single web page. The web page results in an instance of the `OnlineResource` concept. The number of interactions on this web page are stored as the attribute's value.

Applied Preprocessing Steps:

1. retrieving the URIs of the used resources and storing them in a single string
2. transforming the URI string into a word vector, i.e., for each used resource a new attribute is built and the number of interactions is assigned to the attribute's value

5.3.6.6 Feature 43: `included res. content`

The *included resource content* (*included res. content*) feature concatenates the text content of the included resources of the used resources. An example is a text document in which the user has copied text passages from multiple web sites and presentations. Theses copied text passages result in instances of the `ClipboardSnippetResource` concept with the `isIncludedResourceFrom` relation. The concatenation of these text passages constitutes this feature.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.6.7 Feature 44: `included res.`

The *included resources* (*included res.*) feature represents the URIs of the included resources in the used resources. An example is a text document that contains a copied & pasted text paragraph from an email. The text document and the added text result in instances of the `TextDocument` and `ClipboardSnippetResource` concepts respectively with the relation `isIncludedResourceFrom` from the `TextDocument` instance to the `ClipboardSnippetResource` instance. For each included resource a new attribute is created.

Applied Preprocessing Steps:

1. retrieving the URIs of the included resources and storing them in a single string
2. transforming the URI string into a word vector, i.e., for each included resource a new attribute is built and the number of interactions with the used resource is assigned to the attribute's value

5.3.6.8 Feature 45: `referenced res. content`

The *referenced resource content* (*referenced res. content*) feature concatenates the text content of the referenced resources of the used resources. An example is a text document that contains a path to a file. The text document and the file path result in instances of the `TextDocument` and `FileResource` concepts respectively with the relation `isReferencedResourceFrom` from the `TextDocument` instance to the `FileResource` instance. The text content of the referenced files are concatenated which consitutes this feature.

Applied Preprocessing Steps: Standard text processing steps are applied to this feature (see Section 5.3.1).

5.3.7 Switching Sequence Feature Category

The feature category *switching sequence category* includes features that origin from switching between applications, event types, resources and resource types. This category includes 5 different features that are listed in Table 5.9. The construction procedure and the preprocessing steps applied to each feature are described in further detail in this section.

Nr.	**Feature**	**Brief Feature Description**
46	app. switch seq.	the number of switches between two specific applications
47	E type switch seq.	the number of switches between combinations of two `EventType` instances
48	E level res. switch seq.	the combinations of switches from one used resource to another one (event level)
49	E&EB res. switch seq.	the combinations of switches from one used resource to another one (event and event block level)
50	E&EB res. type switch seq.	the combinations of switches from one type of an used resource to another type of an used resource (event and event block level)

Table 5.9: Overview of all features of the switching sequence feature category.

5.3.7.1 Feature 46: `app. switch seq.`

The *application switching sequence of length 2* (*app. switch seq.*) feature represents the number of switches between two applications. An application switch occurs if two consecutive events are captured from two different applications. An example for this is the switch between the Microsoft Internet Explorer and the Microsoft Word application. This results in the string "iexplore_winword". For each combination of an application switch a new attribute is created. The order counts, which means that switching from Microsoft Internet Explorer to Microsoft Word results in a different attribute than vice versa.

Applied Preprocessing Steps:

1. calculating the combinations of application switches and storing them in a single string as tokens, like e.g., "iexplore_winword"
2. transforming the string into a word vector, i.e., for each token a new attribute is built and the number of tokens appearing in the string is assigned to the attribute's value

5.3.7.2 Feature 47: `E type switch seq.`

The *event type switching sequence of length 2* (*E type switch seq.*) feature represents the number of switches between two event types, i.e., two consecutive events belong to two different instances of the `EventType` concepts. An example for this is a user entering a letter in an email and then clicking on the send email button. The event type resulting from the first interaction is an instance of the `Push` concept. The second one is an instance of the `Send` concept. As a result the following numeric attribute is built: "push_send". For each combination of an event type switch a new attribute is created. The order counts similar to the `app. switch seq.` feature.

Applied Preprocessing Steps:

1. calculating the combinations of event type switches and storing them in a single string as tokens, like e.g., "push_send"
2. transforming the string into a word vector, i.e., for each token a new attribute is built and the number of tokens appearing in the string is assigned to the attribute's value

5.3.7.3 Feature 48: `E level res. switch seq.`

The *event level resource switching sequence of length 2* (*E level res. switch seq.*) feature represents the combinations of switches from one used resource to the another one on the event level. On the event level means that two consecutive events have to have the `isActionOn` relation with different URIs associated with them. An example is a single interaction with a text document and then another one with a web page. Both events have the `isActionOn` relation but one targets the instance of the `TextDocument` concept and the other one an instance of the `OnlineResource` concept and hence has another URI as well. The resulting attribute is "<URI of text document>_<URI of web page>". For each combination of resource switches on the event level a new attribute is created. The order counts similar to the `app. switch seq.` feature.

Applied Preprocessing Steps:

1. calculating the combinations of used resource switches and storing them in a single string as tokens, like e.g., "<URI of text document>_<URI of web page>"
2. transforming the string into a word vector, i.e., for each token a new attribute is built and the number of tokens appearing in the string is assigned to the attribute's value

5.3.7.4 Feature 49: `E&EB res. switch seq.`

The *event and event block level resource switching sequence of length 2* (*E&EB res. switch seq.*) feature encapsulates the combinations of switches from one used resource to another one combining the event and the event block level. The event level is the same as for the `E level res. switch seq.` feature. The event block level is different because the detection of a switch between two different used resources is more coarse granular. A switch is detected if two consecutive event blocks have two different used resources associated with them. An event block can have zero or one used resource associated. Such an association takes place if one event of the event block has a `isActionOn` relation. The attribute construction and the preprocessing is similar to the `E level res. switch seq.` feature. The only exception is that for switches on the event level a the "eventlevel_" prefix and for the one on the event block level a "eventblocklevel_" prefix are used.

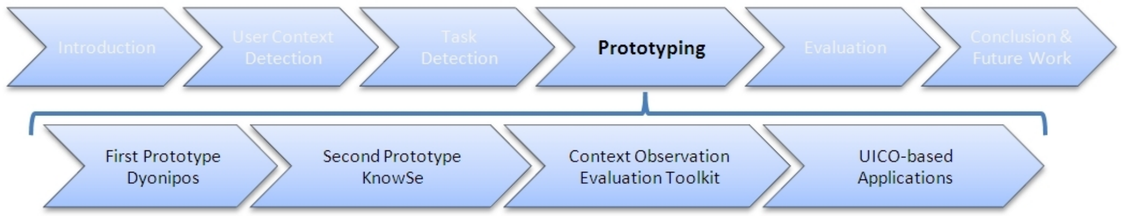
5.3.7.5 Feature 50: `E&EB res. type switch seq.`

The *event and event block level resource type switching sequence of length 2* (*E&EB res. type switch seq.*) feature is constructed similar to the `E&EB res. switch seq.` feature with the difference that the resource type is considered instead of the URI of the used resource. The applied preprocessing steps are the same. An example for an event level attribute is "eventlevel_OnlineResource_TextDocument" and for an event block level attribute is "eventblocklevel_OnlineResource_EmailResource".

5.4 Summary

This chapter introduced the ontology-based task detection approach proposed in this research effort to do task detection. The engineering of 50 distinguished features based on the user interaction context ontology (UICO) including the corresponding preprocessing steps for transforming these features into attributes. These attributes can then be used as input for machine learning algorithms. The implementation and the evaluation of the ontology-based task detection approach is reported in Chapter 6 and in Chapter 7 respectively.

6 Prototyping



This chapter gives an overview about the prototypes that were built in course of this dissertation research effort. The first prototype which is described in Section 6.2 was developed in the frame of the national funded research project *Dyonipos* [Tochtermann *et al.*, 2006]. The second prototype based on the user interaction context ontology (UICO)(see Section 3.3) is described and visualized in Section 6.3. The task detection evaluation environment's architecture and features are presented in Section 6.4. Application prototypes developed based on the UICO on top of the second prototype are highlighted in Section 6.5. These application prototypes concentrate on the areas of user context visualization, context-aware information retrieval and user interruptions.

6.1 Introduction

In the course of this dissertation effort two prototypes were designed and implemented. These prototypes featured context sensors (see Section 3.4) as well as aggregation and resource discovery algorithms (see Section 3.5). Both prototypes were based on the conceptual model referred to as the *semantic pyramid* (see Section 3.2) but only the second prototype includes the user interaction context ontology (UICO) and automatic ontology population mechanisms.

The first prototype described in Section 6.2 was developed in the frame of the national funded research project *Dyonipos* [Tochtermann *et al.*, 2006]. This prototype was utilized to record the dataset for the task detection experiment in the course of the Dyonipos task detection experiment in Austria's ministry of Finance. Dyonipos and the experiment are highlighted in Section 4.4.3 and further described in [Granitzer *et al.*, 2009a, 2008].

The second prototype is based on the UICO and includes mechanisms for automatically populating the UICO as well as a user interface for the task recording and user interaction context exploration. It was used during the three laboratory experiments explained in Chapter 7. The second prototype features a fully service-oriented architecture. The service which is responsible for recording the usage data during the task executions is called *Context OBservation Service* (COBS). An architectural overview of COBS as well as screenshots of the second prototype is given in Section 6.3.

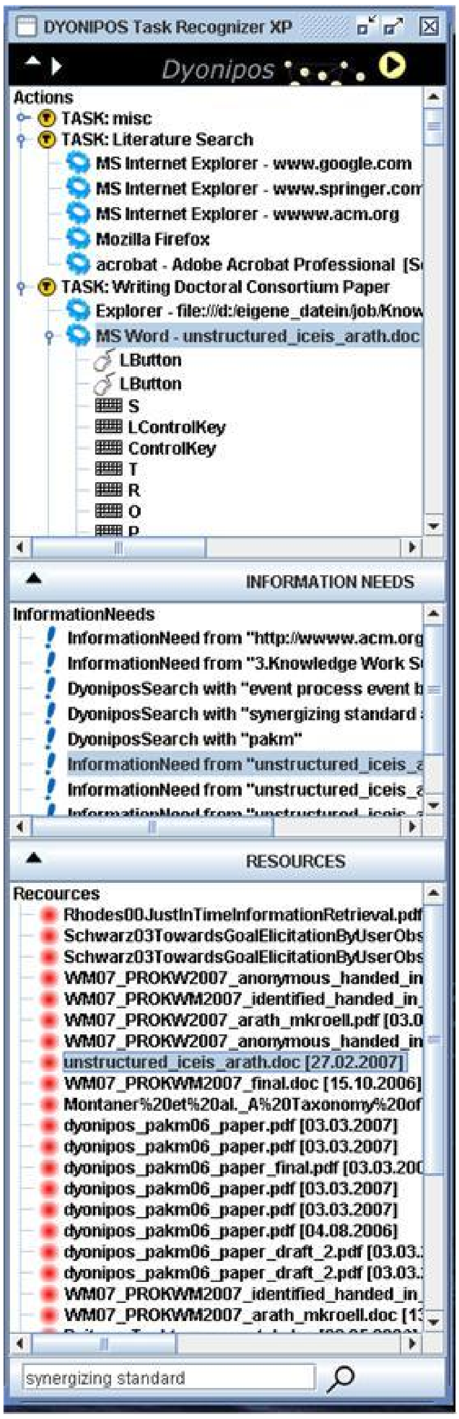
The *Context OBservation Evaluation Toolkit* (COBET) is a service (i) for evaluating the ontology-based task detection approach and (ii) for comparing it to existing approaches. COBET is part of the second prototype. Its architecture is presented in Section 6.4. COBET can be seen as a flexible pipeline for evaluating features and classifiers of task detection approaches starting from training instance construction including feature engineering to evaluating different attribute selection methods and classifiers. COBET also featured several visual and textual presentations of the evaluation results for assessing the classification models.

A variety of applications building on top of the second prototype of this research effort, more specifically the context observation service, are introduced and visualized in Section 6.5. These applications illuminate what kind of applications are possible to create on top of an automatically real-time populated ontology-driven user interaction context model. The application areas span from (i) user interaction context visualizations, (ii) context-aware proactive information retrieval and (iii) measuring the user's interruptibility.

6.2 First Prototype - Dyonipos

The first prototype was developed as part of the research project Dyonipos. This prototype has a client-server architecture constituting of the following four components: (i) a context observer, (ii) a context compiler, (iii) a context processor and (iv) a context analyzer. The context observer targets the Microsoft Windows XP environment (keystrokes and mouse clicks) and standard Microsoft Office applications. It is fine-tuned to the domain of the testing and evaluation environment (Austria's Ministry of Finance). It is able to observe user interactions like keyboard input, mouse clicks and clipboard events, the handling of files in Microsoft Word, Microsoft Excel and Microsoft PowerPoint and the visited web pages in the Microsoft Internet Explorer 6. The context compiler is responsible for receiving the captured user interaction context data (events) from the context observer, transforms it to objects and hand these over to the context analyzer and context processor for further processing. Further processing includes the aggregation to blocks of related events (event blocks) as well as resource discovery and analysis. Resource discovery results in an indexing of the found resources and the extraction of metadata based on the *KnowMiner* knowledge discovery framework [Granitzer, 2008].
The first prototype was tested and evaluated by 10 users of the Austria's Federal Ministry of Finance in a productive environment for approximately 5 weeks. The observed usage data from one user was used to carry out the first task detection experiment highlighted in Section 4.4.3 and further described in [Granitzer *et al.*, 2009a, 2008]. The graphical user interface can be observed in Figure 6.1. The handling of the prototype and its visual components is explained via a short use case scenario in the following paragraphs. For a more detailed description about the first prototype and it's architecture the reader is referred to [Rath, 2007].

Use Case Scenario: After a user has started the *Dyonipos Task Recognizer* application, she begins her daily work on her computer desktop. The task recognizer observes her actions,


DYONIPOS Task Recognizer XP

Dyonipos

Actions
- TASK: misc
- TASK: Literature Search
  - MS Internet Explorer - www.google.com
  - MS Internet Explorer - www.springer.com
  - MS Internet Explorer - wwww.acm.org
  - Mozilla Firefox
  - acrobat - Adobe Acrobat Professional [S
- TASK: Writing Doctoral Consortium Paper
  - Explorer - file:///d:/eigene_datein/job/Know
  - MS Word - unstructured_iceis_arath.doc
    - LButton
    - LButton
    - S
    - LControlKey
    - ControlKey
    - T
    - R
    - O
    - P

INFORMATION NEEDS

InformationNeeds
- InformationNeed from "http://wwww.acm.org
- InformationNeed from "3.Knowledge Work S
- DyoniposSearch with "event process event b
- DyoniposSearch with "synergizing standard
- DyoniposSearch with "pakm"
- InformationNeed from "unstructured_iceis_a
- InformationNeed from "unstructured_iceis_a
- InformationNeed from "unstructured_iceis_a

RESOURCES

Recources
- Rhodes00JustInTimeInformationRetrieval.pdf
- Schwarz03TowardsGoalElicitationByUserObs
- Schwarz03TowardsGoalElicitationByUserObs
- WM07_PROKW2007_anonymous_handed_in
- WM07_PROKWM2007_identified_handed_in
- WM07_PROKW2007_arath_mkroell.pdf [03.0
- WM07_PROKW2007_anonymous_handed_in
- unstructured_iceis_arath.doc [27.02.2007]
- WM07_PROKWM2007_final.doc [15.10.2006]
- Montaner%20et%20al._A%20Taxonomy%20of
- dyonipos_pakm06_paper.pdf [03.03.2007]
- dyonipos_pakm06_paper.pdf [03.03.2007]
- dyonipos_pakm06_paper_final.pdf [03.03.200
- dyonipos_pakm06_paper.pdf [03.03.2007]
- dyonipos_pakm06_paper.pdf [03.03.2007]
- dyonipos_pakm06_paper.pdf [04.08.2006]
- dyonipos_pakm06_paper_draft_2.pdf [03.03.
- dyonipos_pakm06_paper_draft_2.pdf [03.03.
- WM07_PROKWM2007_identified_handed_in
- WM07_PROKWM2007_arath_mkroell.doc [1

synergizing standard

Figure 6.1: The graphical user interface of the first prototype developed as part of the research project Dyonipos. The top area lists the user's tasks including the corresponding event blocks and events. The middle section shows the detected information needs. The resources for fulfilling the user's information need are visualized in the bottom area.

categorizes and displays them in the first area of the graphical user interface - the *action view*. This section shows the user's activities with the corresponding event blocks and events. The yellow button with the "Play" symbol enables the context sensors, i.e., starts the context observer. A person can provide feedback to the system by moving the wrongly matched event blocks to other task entries or confirm the correctly matched ones. Additionally it is possible to create new tasks and event blocks to allow specifying "offline" activities, e.g., meeting in the conference room or telephone conference. The user can delete the observed actions and stop the context sensor process at any time. The *information need view* visualizes the recognized information need that origin from the user interaction context. By selecting an information need the resources intended to fulfill the information need are displayed. In the *resources view* relevant information sources for the actual task, event block, event or information need, are visualized. Resources are not limited to documents or links as showed in Figure 6.1 but can also include other resources available on the user's computer desktop. At the bottom of the graphical user interface there is a search field that allows the user to search through the iteratively built resource repository.

6.3 Second Prototype - KnowSe

The second prototype was built based on the lessons learned from the first prototype. This prototype is called *KnowSe* which stands for "Knowledge Services". *"KnowSe forms the basis for dynamically orchestrating a large variety of intelligent knowledge services, highly contextualized to a persons work context and interconnected with a multitude of knowledge sources* [KnowSe, 2009]. KnowSe's services can be utilized for example to support personalized learning [Rath *et al.*, 2009b] and improve computer supported collaborative work [Rath *et al.*, 2009c]. KnowSe is a service-oriented *Eclipse Rich Client Platform* (Eclipse RCP) application comprised of several *OSGI*-services [OSGI2008, 2008].

Two screenshot of the KnowSe application utilized in the observation of the task executions in the three laboratory experiments described in Chapter 7 is visualized in Figure 6.2 and in Figure 6.3. Figure 6.2 illuminates the user interaction context observation capabilities for various resources and for various applications. The second screenshot in Figure 6.3 displays the implementation and realization of the software requirements derived from the results of the user questionnaires discussed in Section 7.4. The *Task History* and the *Triple View* allows the user to explore their recorded usage data on various granularity levels. Furthermore these views provides the users with the functionality to do text-based search and to easily delete UICO concept instances and relations. The *SPARQL* view is an easy to use testbed on the one hand and on the other hand to show the great potential of the populated UICO model for user profiling, information and expert recommendation, relation analysis as well as a complementary part of semantic desktop systems.

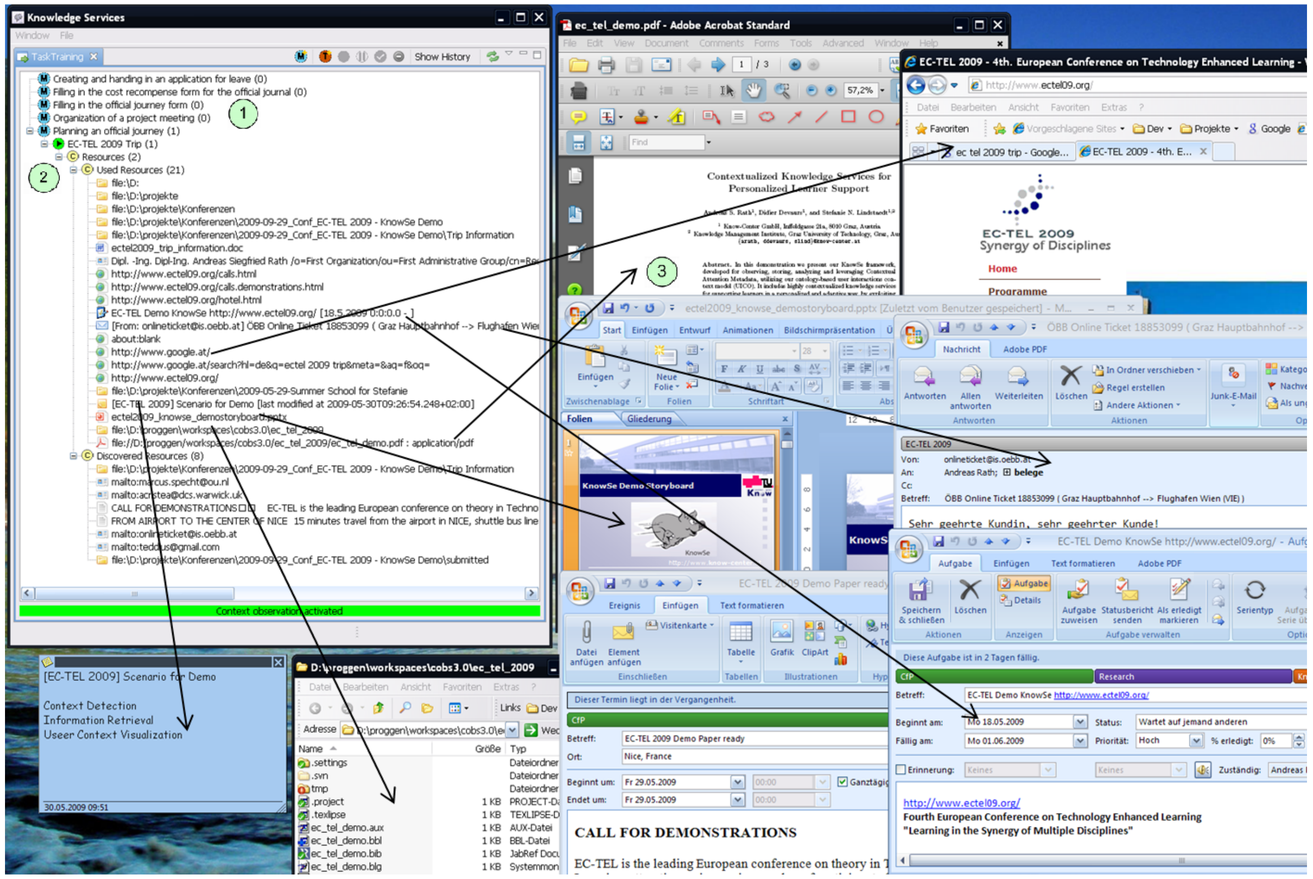
The OSGI services were designed and implemented as part of this dissertation research effort together with the help and support of several up-to-date technology enthusiastic students: (i) Daniel Resanovic implemented the mapping algorithms between the user interaction context model and the graphical user interface as well as a connection to the APOSDLE platform [Resanovic, 2008], (ii) Georg Kompacher did the graphical interface components for the task experiment [Kompacher, 2008], (iii) Stefanie Wechtitsch worked on the integration of user interruptibility measures [Wechtitsch, 2008] and (iv) Thomas Pichler contributed the timeline visualization[Pichler, 2010]. Didier Devaurs, a colleague also working in the KnowSe project, designed and implemented the OSGI service that is responsible for interfacing the KnowMiner knowledge discovery framework [Granitzer, 2008].

Two exceptions to this are the **C**ontext **OB**servation **S**ervice (COBS) and the **C**ontext **OB**servation **E**valuation **T**oolkit (COBET) which were fully designed and implemented as part of this dissertation research effort. It followed a service-oriented architecture approach and is realized as an OSGI service. COBS utilizes the *OpenAnzo* framework[1] [OpenAnzo.org, 2008] as a semantic middleware for storing and manipulating the automatically populated user interaction context model (see Section 3.3). The reasons why the OpenAnzo framework was chosen for COBS are manifold and listed bellow:

- named graph support
- tracking of graph modifications
- offline storage and synchronization of graphs
- versioning of named graphs
- notify/update mechanism
- SPARQL query support on named graphs and sets of named graphs
- user roles and policy support
- open source and freely modifiable
- automatic Java class generation support based on an ontology

A global overview of the architecture of COBS is given in the next section.

[1]The *OpenAnzo* framework - an open source RDF store, query engine and related middleware for the development of semantic applications.


Knowledge Services

Creating and handing in an application for leave (0)
Filing in the cost recompense form for the official journal (0)
Filing in the official journey form (0)
Organization of a project meeting (0)
Planning an official journey (1)
EC-TEL 2009 Trip (1)
Resources (2)
Used Resources (21)
file:\D:
file:\D:\projekte
file:\D:\projekte\Konferenzen
file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo
file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo\Trip Information
ectel2009_trip_information.doc
http://www.ectel09.org/calls.html
http://www.ectel09.org/calls.demonstrations.html
http://www.ectel09.org/hotel.html
EC-TEL Demo KnowSe http://www.ectel09.org/ [18.5.2009 0:0:0.0 -]
[From: onlineticket@is.oebb.at] ÖBB Online Ticket 18853099 (Graz Hauptbahnhof --> Flughafen Wien
about:blank
http://www.google.at/
http://www.google.at/search?hl=de&q=ectel 2009 trip&meta=&aq=f&oq=
http://www.ectel09.org/
file:\D:\projekte\Konferenzen\2009-05-29-Summer School for Stefanie
[EC-TEL 2009] Scenario for Demo [last modified at 2009-05-30T09:26:54.248+02:00]
ectel2009_knowse_demostoryboard.pptx
file:\D:\proggen\workspaces\cobs3.0\ec_tel_2009
file://D:/proggen/workspaces/cobs3.0/ec_tel_2009/ec_tel_demo.pdf : application/pdf
Discovered Resources (8)
file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo\Trip Information
malto:marcus.specht@ou.nl
malto:acristea@dcs.warwick.uk
CALL FOR DEMONSTRATIONS EC-TEL is the leading European conference on theory in Techno
FROM AIRPORT TO THE CENTER OF NICE 15 minutes travel from the airport in NICE, shuttle bus line
malto:onlineticket@is.oebb.at
malto:teddcus@gmail.com
file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo\submitted

Context observation activated

[EC-TEL 2009] Scenario for Demo
Context Detection
Information Retrieval
Useer Context Visualization
30.05.2009 09:51

ec_tel_demo.pdf - Adobe Acrobat Standard

Contextualized Knowledge Services for Personalized Learner Support

EC-TEL 2009 - 4th. European Conference on Technology Enhanced Learning
http://www.ectel09.org/
EC-TEL 2009 Synergy of Disciplines

ectel2009_knowse_demostoryboard.pptx
KnowSe Demo Storyboard

ÖBB Online Ticket 18853099 (Graz Hauptbahnhof --> Flughafen Wien (VIE))
Von: onlineticket@is.oebb.at
An: Andreas Rath; belege
Betreff: ÖBB Online Ticket 18853099 (Graz Hauptbahnhof --> Flughafen Wien (VIE))
Sehr geehrte Kundin, sehr geehrter Kunde!

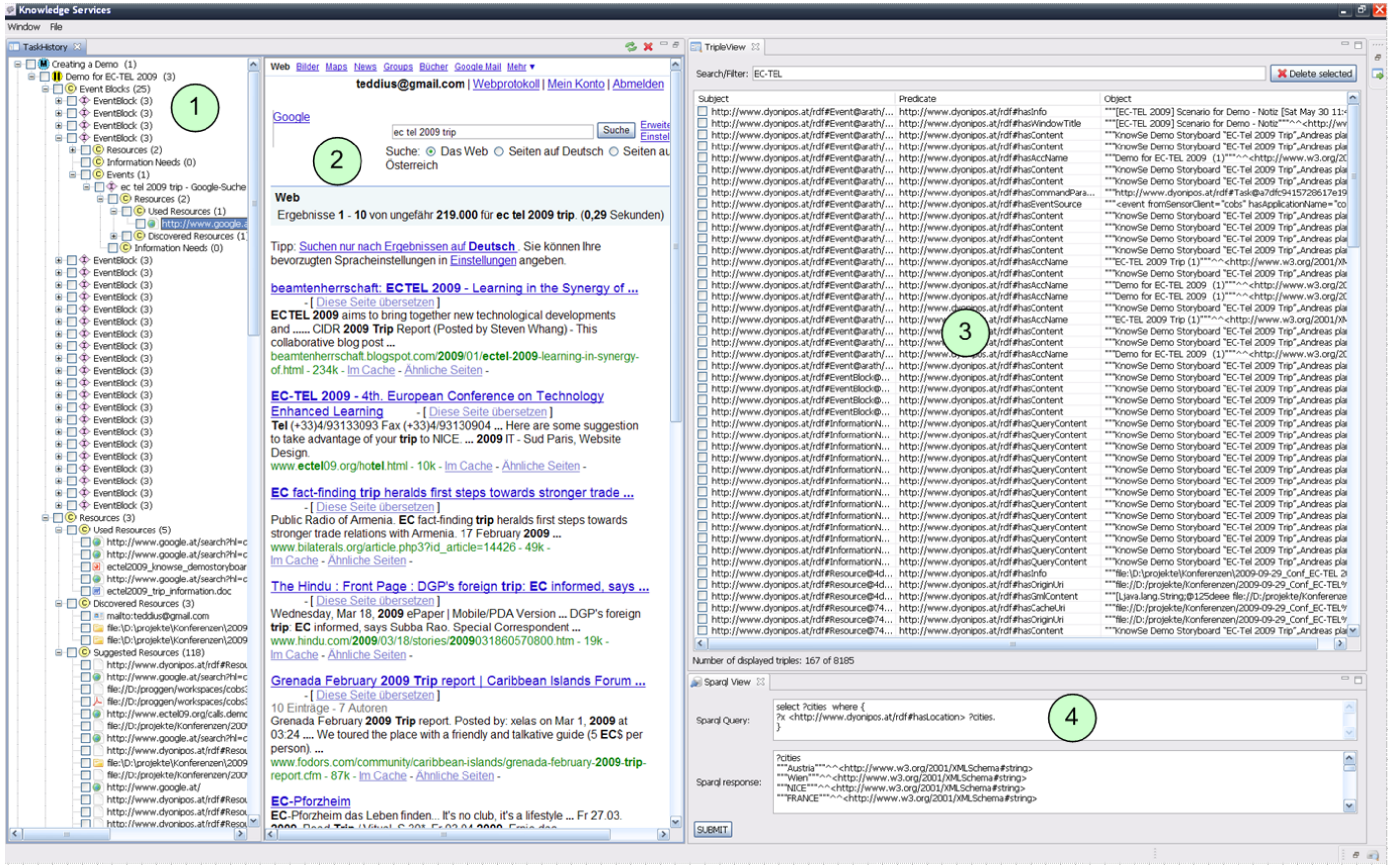
EC-TEL 2009 Demo Paper ready
Dieser Termin liegt in der Vergangenheit.
Betreff: EC-TEL 2009 Demo Paper ready
Ort: Nice, France
Beginnt um: Fr 29.05.2009
Endet um: Fr 29.05.2009
CALL FOR DEMONSTRATIONS
EC-TEL is the leading European conference on theory in T

EC-TEL Demo KnowSe http://www.ectel09.org/ - Aufgabe
Diese Aufgabe ist in 2 Tagen fällig.
Betreff: EC-TEL Demo KnowSe http://www.ectel09.org/
Beginnt am: Mo 18.05.2009 Status: Wartet auf jemand anderen
Fällig am: Mo 01.06.2009 Priorität: Hoch % erledigt: 0%
http://www.ectel09.org/
Fourth European Conference on Technology Enhanced Learning
"Learning in the Synergy of Multiple Disciplines"

D:\proggen\workspaces\cobs3.0\ec_tel_2009

Name	Größe	Typ
.settings		Dateiordner
.svn		Dateiordner
tmp		Dateiordner
.project	1 KB	PROJECT-D
.texlipse	1 KB	TEXLIPSE-D
ec_tel_demo.aux	1 KB	AUX-Datei
ec_tel_demo.bbl	1 KB	BBL-Datei
ec_tel_demo.bib	1 KB	JabRef Doc
ec_tel_demo.blg	1 KB	Systemmon

Figure 6.2: A screenshot of the KnowSe prototype which was developed as part of this research effort and utilized in the observation of the task executions in the three laboratory experiments is displayed in this figure. On the top left side in (1) the task models are listed from which the user can choose to instantiate a task. Also on the left the current task is shown including the captured used and computed discovered resources (2). On the right and bottom part of this figure the applications are displayed from which the user interaction context were automatically captured.



Sparql Query:

```
select ?cities where {
?x <http://www.dyonipos.at/rdf#hasLocation> ?cities.
}
```

Sparql response:

```
?cities
"""Austria"""^^<http://www.w3.org/2001/XMLSchema#string>
"""Wien"""^^<http://www.w3.org/2001/XMLSchema#string>
"""NICE"""^^<http://www.w3.org/2001/XMLSchema#string>
"""FRANCE"""^^<http://www.w3.org/2001/XMLSchema#string>
```

Figure 6.3: *A screenshot of the user interaction context exploration views. On the left the figure shows the Task History view (1) with the UICO concept instances as well as a detailed view about the selected used resource (2), in this case the Google results web page. In the right area there is the Triple view (3) which lists all statements/triples about the selected concept, in this case the used resource. The SPARQL view (4) in the bottom right area illuminates the SPARQL query for all automatically recognized locations in the populated UICO. Beneath the query the resulting locations are listed.*

6.3.1 Architecture

The base architecture of COBS consists of six components: (i) the `ContextObservationService`, (ii) the `UserMonitor`, (iii) the `ContextObservers`, (iv) the `TaskManagemnent`, the (v) `OntologyAccess` and (vi) the `ContextAnalysis` component. The component diagram of COBS is visualized in Figure 6.4. An overview of the components is given in the following paragraphs.

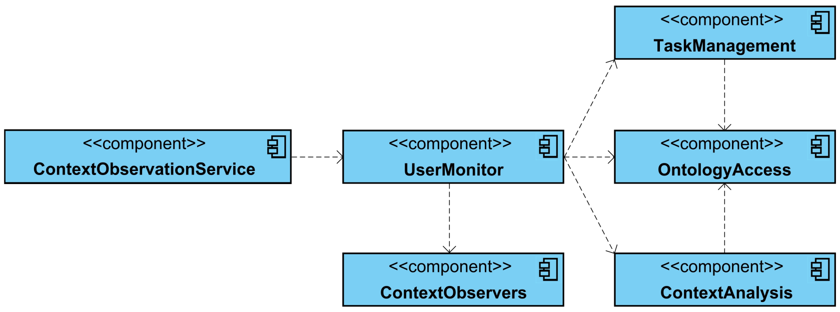


Figure 6.4: The component diagram of the base components consisting of (i) the `ContextObservationService`, (ii) the `UserMonitor`, (iii) the `ContextObservers`, (iv) the `TaskManagemnent`, (v) the `OntologyAccess` and (vi) the `ContextAnalysis` component for automatically observing and detecting the user interaction context as well as automatically populating the user interaction context ontology (UICO).

`ContextObservationService`: The `ContextObservationService` is the main service component which exposes a wide range of functionalities of the central `UserMonitor` component to other application and services. The service interface provides methods for starting and stopping the context observation, configuring the information need detection mechanism, controlling the task handling and most importantly querying the UICO with the SPARQL semantic web query language.

`UserMonitor`: The `UserMonitor` is the central component which controls the task management (`TaskManagement`), the access to the UICO via the `OntologyAccess` component, the starting and stopping of the external context sensors (`ContextObservers`) including the receiving and parsing of the raw sensor data as well as the forwarding of this data to the `ContextAnalyis` component.

ContextObservers: Multiple sensors were developed for observing the user's interactions with applications and resources as well as retrieving data and metadata about the acted upon resources as described in Section 3.4. The `ContextObservers` component stands for all the sensors. A listing of all developed application and operating system sensors is given in Section 3.4.1. The sensors are implemented in different programming languages in respect to the applications they monitor. The Microsoft Office sensor pack is implemented in C# and C whereas the Mozilla Thunderbird and Firefox extensions are written in the *XML User Interface*

Language (XUL) [2] and JavaScript. The `UserMonitor` acts as a server and the sensors as clients that send their observed usage data in form of an *event xml* format also developed as part of this dissertation research. The sensors are *"lightweight"* implementations, which means that they do not have additional functionality except observing the user interaction context and sending it to the `UserMonitor`. The advantage of this design is that other context sensors can be rapidly developed independently in the most suitable programming language, since there is no sophisticated preprocessing required on the context sensor side. This allows an easy and fast extensibility of the sensing capabilities.

TaskManagement: The `TaskManagement` is responsible for the appropriate handling of tasks including checking the preconditions for a task state transition. This component is inspired by the *NEPOMUK Task State Model* [Grebner *et al.*, 2007] and implements the NEPOMUK task states in form of a finite-state automaton. This task state model allows the users to modify their user interaction context manually by creating, executing, interrupting, finishing, aborting, approving and archiving a task. The connection to the `OntologyAccess` allows the `TaskManagement` to directly map the resulting tasks, task states and task transitions into the UICO. All the manipulations of a task, e.g., create, delete, suspend and so on, are reflected directly in the context model which means that the state of the UICO is synchronous with the state of the tasks at all times. This synchronization is especially important for other services when querying the UICO for task or task switch information at a certain point in time. The renaming and the labeling as well as the assignment of a task to a task model is one of the `TaskManagement` component's responsibilities.

OntologyAccess: The `OntologyAccess` component is the key component that manages all manipulations of the UICO. The UICO is implemented as a quadstore. A quadstore in comparison to a triple store has an additionally value that stores the URI of the associated graph, called named graph, in which a triple/statement exists. The UICO consists of multiple named graphs. Named graphs make it easier to implement relations between graphs like for example, versioning, user policies, change tracking, imperfection of data or trust [Sintek *et al.*, 2007]. Each application session, each task, the UICO ontology itself and the combined task models have their own named graph. The advantage of the separation into multiple named graphs is to decrease the number of total triples in one graph. This speeds up the performance of the SPARQL queries. The `OntologyAccess` component has its internal graph management functionality which handles the association of the UICO concept instances and properties to the correct graph. The graphs can be queried together and separately. A serialization, synchronization, versioning and restricting the access rights of the graphs is also possible. The utilized semantic middleware is the *OpenAnzo* framework [OpenAnzo.org, 2008]. Following advantages come with the ontology-based user interaction context model in combination with the used semantic framework:

- Analysis of the user interaction context at a specific time or time frame (e.g., a task).

[2] XML User Interface Language (XUL), `https://developer.mozilla.org/en/XUL`

- Tracking and analyzing changes of the UICO instances and relations on the graph and triple level.
- Actuality of the UICO: the observed contextual data and the derived information are stored directly in the user interaction context ontology model.
- Maintaining consistency of the UICO during automatic population, i.e., following the specifications of the ontology by having the Java classes automatically generated from the ontology. The access to the ontology instances through Java classes is realized through automatically generated SPARQL queries.

ContextAnalysis: The events originating from the context sensors are analyzed by the `ContextAnalysis` component. The analysis steps include (i) the abstraction and analysis of user interactions (see Section 3.3.1), (ii) the detection of information needs [Rath *et al.*, 2007] and (iii) the discovery of used, references as well as included resources (see Section 3.5.2). These analysis steps are performed by the `UserInteractionAnalysis`, the `InformationNeedDetection`, the `ResourceDiscovery` and `ResourceRetriever` components. The `ContextAnalysis` component also indexes all resources the user has interacted with and retrieves relevant resources as a fulfillment mechanisms to the user's information needs. The `ContextAnalysis` component brings together the information about the resources and the user interactions as well as the retrieved resources.

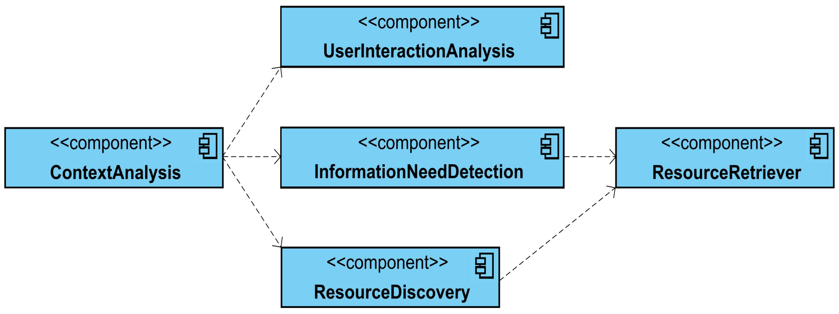


Figure 6.5: The component diagram of the `ContextAnalysis` component consisting of the `UserInteractionAnalysis`, the `InformationNeedDetection`, the `ResourceDiscovery` and the `ResourceRetriever` components for analyzing the user interaction context.

UserInteractionAnalysis: Abstracting and aggregating events to event blocks as well as the computation of event types as described in Section 3.5 are situated in this component. Events are grouped together to event blocks based on static rules depending on the application in which the event are observed. These rules are implemented in Java and detailed in Section 3.5.3.

ResourceDiscovery: The `ResourceDiscovery` component identifies resources in the event data stream with various techniques as elaborated in Section 3.5.2. Next to regular expressions and direct resource identification the knowledge discovery framework, the *KnowMiner* framework [Granitzer, 2008], is utilized for information extraction. Persons, locations and organization are recognized, related to the resources in which they are detected and directly mapped to concept instances and relations of the UICO and stored in the UICO via the `OntologyAccess` component. All the discovered resources are indexed by the KnowMiner framework for search, retrieval and clustering. The URIs of a specific resource in the index, in the Java object model and in the UICO are equal and unique.

InformationNeedDetection: The detection of information needs of the user is handled by this component. Static rules are defined and implemented to recognize a user's information need. The rules utilize the current user interaction context of the user for query expansion as well as search space narrowing. The detection of information needs is based on various context features, e.g., application name, input characters, browser's request URL, current resource in focus, or last accessed resources. A more elaborated description about this component is given in [Kröll *et al.*, 2006; Rath *et al.*, 2007] and about the exploitation in [Rath *et al.*, 2008].

6.4 Context OBservation Evaluation Toolkit (COBET)

The **C**ontext **OB**servation **E**valuation **T**oolkit (COBET) was designed and implemented as an important part of this research. COBET is the key component for evaluating the performance of the ontology-based task detection approach as well as for comparing it to already existing approaches. Here the features, feature combinations and the reported preprocessing steps for the *Dyonipos* [Granitzer *et al.*, 2008], the *SWISH* [Oliver *et al.*, 2006], and the *TaskPredictor 1* [Shen, 2009; Shen *et al.*, 2006] approaches were implemented based on the explanations available in the existing literature.

6.4.1 Architecture

The architecture of COBET can be seen as a flexible and an easy to use pipeline based system for evaluating machine learning algorithm performance on a given dataset. In the machine learning part the machine learning toolkit Weka [Witten & Frank, 2005], which is well-known and widely used in the machine learning research community, is used for various operations regarding data preprocessing, filtering, attribute selection, classification and performance measurements. The focus of COBET is not on implementing new attribute selection methods or new classifiers. COBET's main focus is on exploring and discovering the best features, feature combinations, preprocessing steps and classifiers for a machine learning task, i.e., studying the classification problem *"task detection"*. The base components of the architecture are displayed in Figure 6.6 and described in the following paragraphs.

Pipeline: The `Pipeline` component is the central part which combines data loading, training instance construction, context feature to attributes transformation, performance evaluation and finally the graphical and text-based output generation. For the UICO, the Dyonipos, the *SWISH* and the *TaskPredictor 1* approaches own instances of a pipeline were built in order to evaluate the task detection performance on the experiment's datasets.

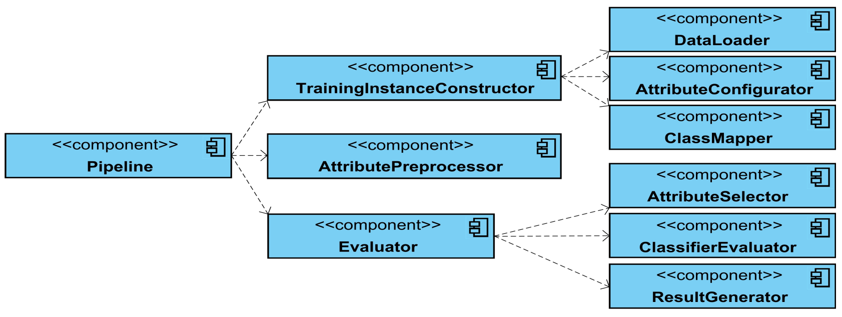


Figure 6.6: This figure shows the component diagram of the Context OBservation Evaluation Toolkit (COBET) designed and implemented to evaluate the ontology-based task detection approach proposed in this research for the experiments described in Chapter 7.

TrainingInstanceConstructor: Constructing the training/test instances from the task execution data based on a predefined configuration is the responsibility of the `TrainingInstanceConstructor` component. It uses the `DataLoader` component for retrieving the data from the UICO or from *arff* files[3]. Based on the configuration of the attributes provided by the `AttributeConfigurator` component the `TrainingInstanceConstructor` constructs the training/test instances for the classifiers. The labeling of the training/test instances is flexible and configurable through the `ClassMapper` component. Regarding the construction of the training and test instances, it is important to note here, that the construction of each training and each test instance is done separately and independently in order to assure that there are no influencing factors introduced in the construction and later in the evaluation process of the task detection approaches.

DataLoader: This component handles the loading of task execution data from (i) an existing *arff* file and (ii) is able to retrieve task execution data from a populated UICO graph. In the latter case, both retrieving data from a locally stored UICO graph in a file as well as retrieving the data via querying a remote UICO stored on an OpenAnzo server are supported.

AttributeConfigurator: The configuration of the attributes that constitute a training/test instance is specified in the `AttributeConfigurator` component. The configuration of an attribute includes the type of the attribute, e.g., *nominal*, *string*, *boolean* or *numeric* attribute, as well as the name of the attribute.

ClassMapper: This component allows the specification of rules how a training/test instance is labeled. An example for this is the labeling of the training/test instances with the corresponding task model, with routine or knowledge-intensive task or with laboratory or personal workstation task.

[3]The *arff* file format is the application specific file format of Weka for storing training instances, and learned classification models.

AttributePreprocessor: The preprocessing steps are configured in advanced for each attribute or group of attributes separately. Since the order of the preprocessing steps influence the classification result, the order in which the preprocessing steps are performed on the attributes are important to follow. Based on the configuration of the attributes they are preprocessed. Both, the configuration and the preprocessing of the attributes happen in the `AttributePreprocessor` component. Some of the attribute transformation methods[4] used are provided by Weka, such as `StringToWordVector` or `Discretize`. Other preprocessing methods were newly implemented, such as a specialized form of *StopwordRemoval* or string manipulation methods. The attribute transformation methods used for the UICO approach are described in Section 5.3.

Evaluator: The `Evaluator` component evaluates the the performance of a task detection pipeline. For this it utilizes the following three components which are (i) `AttributeSelector` for selecting a specific number of attributes for the classifier, (ii) the `ClassifierEvaluator` for classifier performance evaluation and (iii) the `ResultGenerator` for text-based and graphical output generation.

AttributeSelector: The `AttributeSelector` selects a specific number of attributes for the classifiers in order to evaluate on how many attributes a classifier performs best.

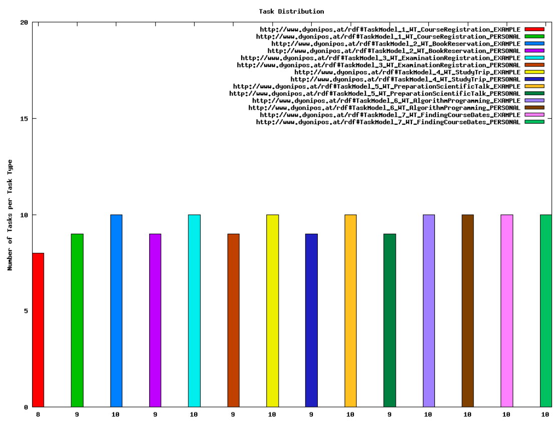
ClassifierEvaluator: The evaluation of the classifier performance can be done in two ways: (i) stratified 10-fold cross-validation and (ii) train/test set evaluation. In both cases the Weka functionalities are utilized by the `ClassifierEvaluator` component to evaluate the classification performance.

ResultGenerator: The `ResultGenerator` component creates graphical as well as text-based output based on the classifiers' performance results. The graphical output is based on the functionalities provided by *Gnuplot*[5], a function plotting utility.

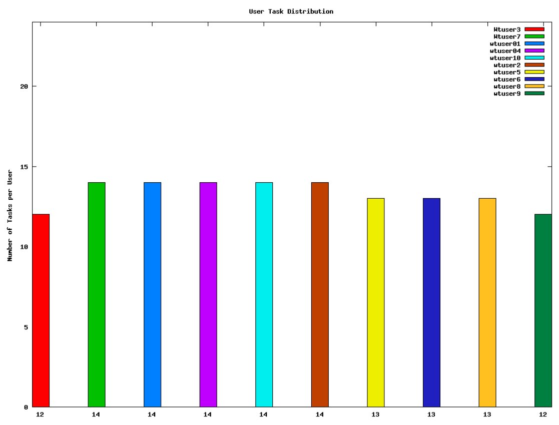
Implemented graphical outputs are visualized in Figure 6.7 and include:

- dataset statistic about the distribution of task models, tasks and users
- graph plots about classification accuracy values of classifiers versus the number of attributes with and without including the standard definition of the cross-validation folds
- top 10 and top 20 best performing configurations of a specific number of attributes and classifiers in respect to accuracy, micro precision, micro recall as bar charts

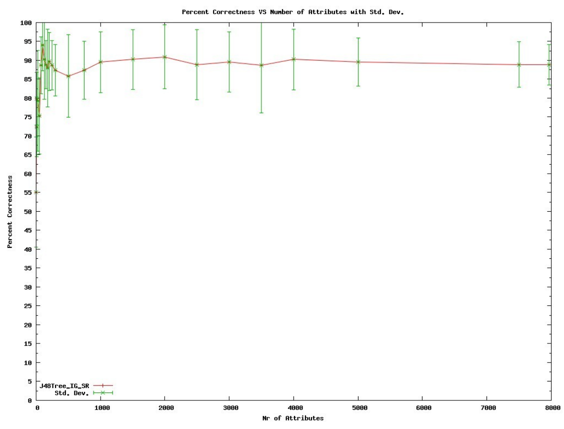
[4]In Weka the attribute transformation methods are called *Filters*
[5]Gnuplot, `http://www.gnuplot.info`



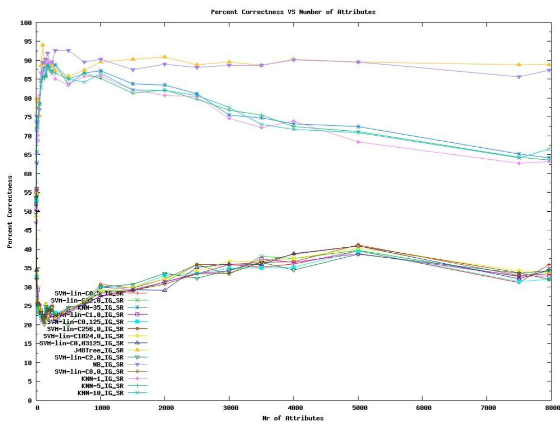
(a) Task instance distribution in respect to the task models.



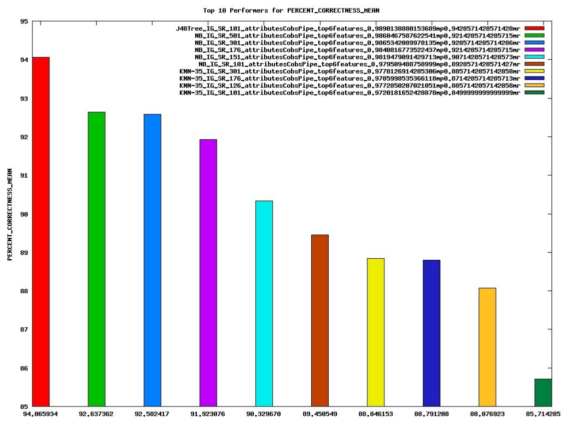
(b) Task instance distribution in respect to the users.



(c) J48 classifier accuracy values including the standard deviation across the cross-validation folds.



(d) Classification accuracy values of the classifiers varying the number of used attributes.



(e) Top 10 best performing pipeline configurations in terms of mean classification accuracy of the stratified 10-fold cross-validation folds

Figure 6.7: Examples of graphical outputs automatically generated by COBET during the evaluation of the task detection performance of the UICO pipeline on the laboratory experiment 2 dataset (see Section 7.5).

6.5 UICO-based Applications

On the top of the *Context OBservation Service* (see Section 6.3) including the user interaction context ontology (UICO) as an integrated part, several application prototypes were built in the frame of the *KnowSe* project. This section allows the reader to get a short glance what kind of applications are possible to create based on this research efforts.

6.5.1 User Interaction Context Visualization

The visualization of the user interaction context is not only useful for users to inspect what kind of information was recorded about them (see Section 7.4) but also to enable a reflection possibility on the tasks performed and the utilized resource. The timeline visualization shows the tasks and the utilized resources during a work day, week, or month. The graph-based visualization allows an identification of similar tasks of the user and of colleagues. This view also supports the finding of possible collaborators based on topics, resources and tasks.

6.5.1.1 Timeline Visualization

A *timeline* visualization was created on the basis of the user interaction context. This visualization allows the users to reflect on their performed tasks including the utilized resources. A timeline visualization of the automatically captured and computed user interaction context of a work day was developed together with Pichler [2010]. A sample timeline is visualized in Figure 6.8. The *SIMILE Widgets* toolkit available at [MIT, 2009] served as a base for this visualization.

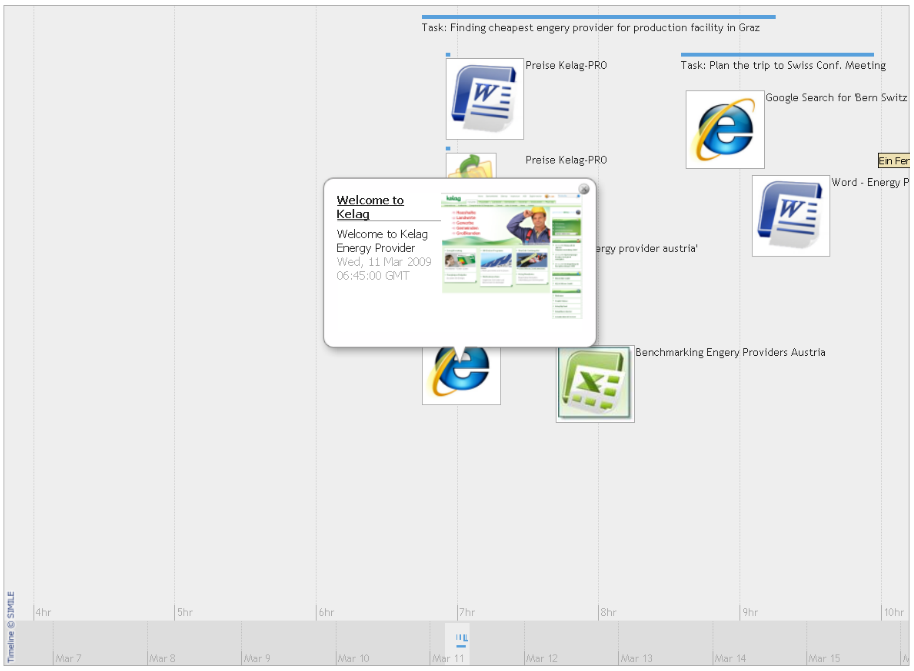
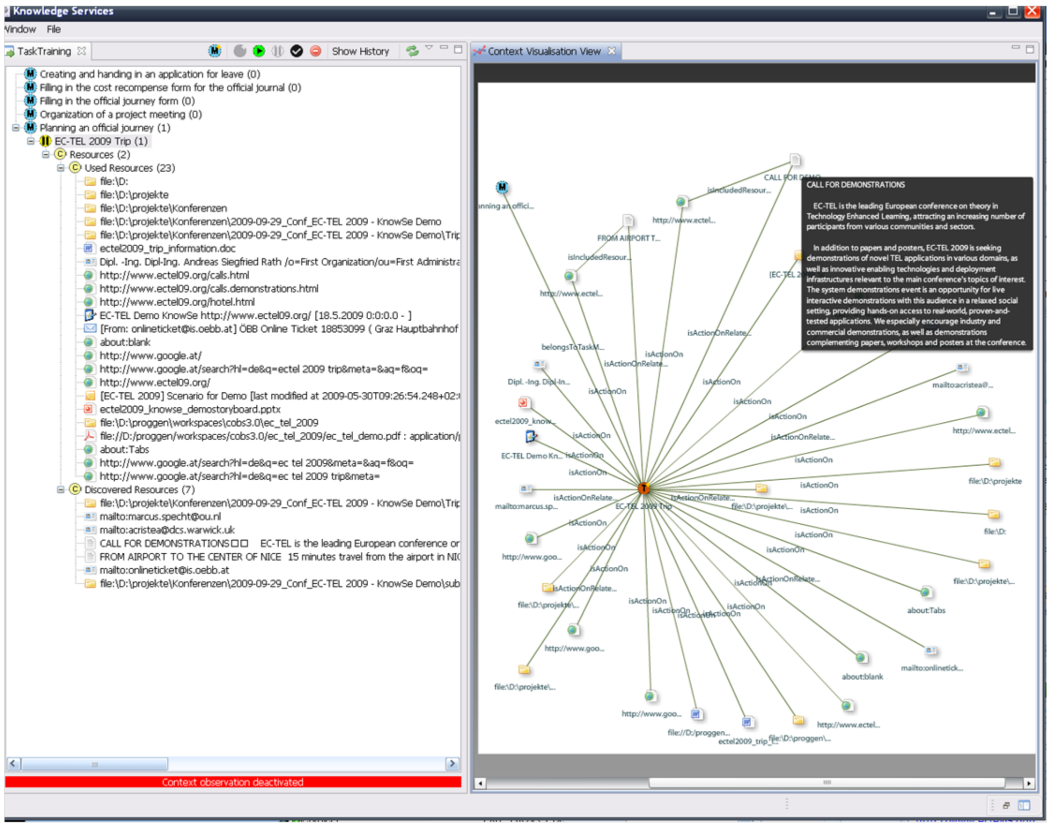

Task: Finding cheapest engery provider for production facility in Graz
Preise Kelag-PRO
Task: Plan the trip to Swiss Conf. Meeting
Google Search for 'Bern Switz
Preise Kelag-PRO
Ein Fer
Word - Energy P
Welcome to Kelag
Welcome to Kelag Energy Provider
Wed, 11 Mar 2009 06:45:00 GMT
ergy provider austria'
Benchmarking Engery Providers Austria
Timeline © SIMILE
4hr
5hr
6hr
7hr
8hr
9hr
10hr
Mar 7
Mar 8
Mar 9
Mar 10
Mar 11
Mar 12
Mar 13
Mar 14
Mar 15

Figure 6.8: This figure shows a timeline visualization of the user interaction context comprising of tasks and resources of a work day.

6.5.1.2 Graph Visualization

The UICO can also be seen as a graph representing the relations between users, user interactions, resources, tasks and applications. The graph visualizations were developed together with Georg Kompacher and are shown in Figure 6.9 and Figure 6.10. They utilize the open source *RaViz Relational Analysis Component* [RaVis, 2009] library for rendering the concept instances and the corresponding relations of the user interaction context. In Figure 6.9 a user's task is displayed with the utilized resources during a task execution. Figure 6.10 shows the relations between tasks of multiple users.



```
Creating and handing in an application for leave (0)
Filling in the cost recompense form for the official journal (0)
Filling in the official journey form (0)
Organization of a project meeting (0)
Planning an official journey (1)
  EC-TEL 2009 Trip (1)
    Resources (2)
      Used Resources (23)
        file:\D:
        file:\D:\projekte
        file:\D:\projekte\Konferenzen
        file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo
        file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo\Trip
        ectel2009_trip_information.doc
        Dipl. -Ing. Dipl-Ing. Andreas Siegfried Rath /o=First Organization/ou=First Administra
        http://www.ectel09.org/calls.html
        http://www.ectel09.org/calls.demonstrations.html
        http://www.ectel09.org/hotel.html
        EC-TEL Demo KnowSe http://www.ectel09.org/ [18.5.2009 0:0:0.0 - ]
        [From: onlineticket@is.oebb.at] ÖBB Online Ticket 18853099 ( Graz Hauptbahnhof
        about:blank
        http://www.google.at/
        http://www.google.at/search?hl=de&q=ectel 2009 trip&meta=&aq=f&oq=
        http://www.ectel09.org/
        [EC-TEL 2009] Scenario for Demo [last modified at 2009-05-30T09:26:54.248+02:
        ectel2009_knowse_demostoryboard.pptx
        file:\D:\proggen\workspaces\cobs3.0\ec_tel_2009
        file://D:/proggen/workspaces/cobs3.0/ec_tel_2009/ec_tel_demo.pdf : application/
        about:Tabs
        http://www.google.at/search?hl=de&q=ec tel 2009&meta=&aq=f&oq=
        http://www.google.at/search?hl=de&q=ec tel 2009 trip&meta=
      Discovered Resources (7)
        file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo\Trip
        mailto:marcus.specht@ou.nl
        mailto:acristea@dcs.warwick.uk
        CALL FOR DEMONSTRATIONS□□   EC-TEL is the leading European conference or
        FROM AIRPORT TO THE CENTER OF NICE  15 minutes travel from the airport in NI
        mailto:onlineticket@is.oebb.at
        file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo\sub
```

CALL FOR DEMONSTRATIONS

EC-TEL is the leading European conference on theory in Technology Enhanced Learning, attracting an increasing number of participants from various communities and sectors.

In addition to papers and posters, EC-TEL 2009 is seeking demonstrations of novel TEL applications in various domains, as well as innovative enabling technologies and deployment infrastructures relevant to the main conference's topics of interest. The system demonstrations event is an opportunity for live interactive demonstrations with this audience in a relaxed social setting, providing hands-on access to real-world, proven-and-tested applications. We especially encourage industry and commercial demonstrations, as well as demonstrations complementing papers, workshops and posters at the conference.

Figure 6.9: This figure displays a graph visualization of the user interaction context observed during a "Plan a trip" task focusing on resources related to this task.

6.5.2 User Interruptibility

Handling user interruptions and notifications is one of the challenges a user has to deal with during work on a computer desktop. The user is constantly interrupted by instant messaging applications or email notifications. These interruptions and notifications are disruptive, have negative impact on the productivity, annoy and distract the user.

KnowSe also uses notifications about task detection and proactive information delivery. To motivate users to work with the application instead of switching it off, a mediator is required

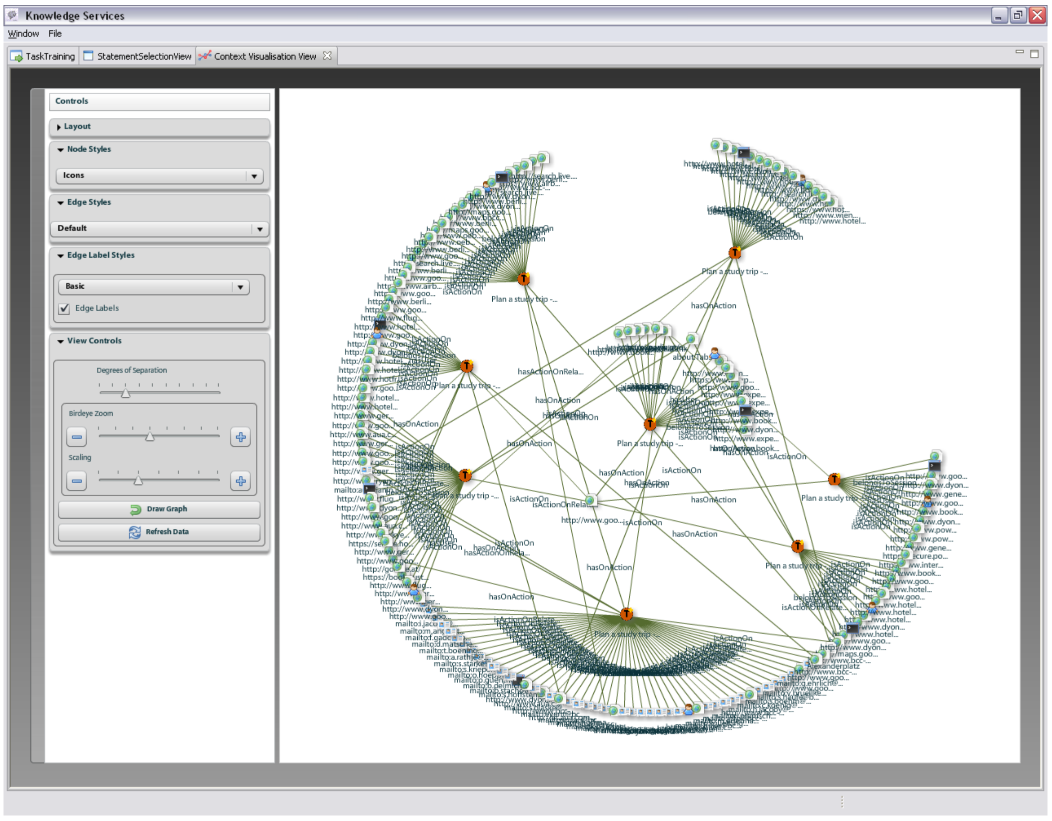


Figure 6.10: A graph visualization of user interaction context of multiple users with a focus on the utilized resources during the task executions.

to decide when and how to interrupt the user [McFarlane, 2002]. The computation of how interruptible a user is, is based on the user context [Gievska & Sibert, 2004].

Together with Stefanie Wechtitsch [Wechtitsch, 2008] and Didier Devaurs a mediator was implemented that utilizes the automatically captured user interaction context by COBS in order to (i) suppress notifications, (ii) trigger task detection and (iii) decide upon the visual representation of a notification. Following characteristics of the current context were taken into account [Horvitz *et al.*, 2004]: the application on focus, the last switch of applications, the number of window title switches, the number of mouse clicks and keystrokes, and the duration a user was inactive. Two illustrative examples of notifications are shown in Figure 6.11 and Figure 6.12 that display a notification bubble (balloon) and a tray icon notification respectively for signaling the user that a new task was detected.

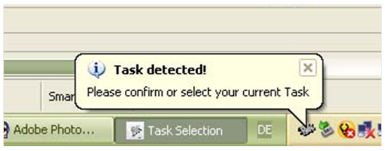

Task detected!
Please confirm or select your current Task

Figure 6.11: This figure visualizes the KnowSe's notification bubble that notifies the user about the detection of her current task.

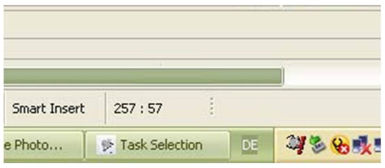
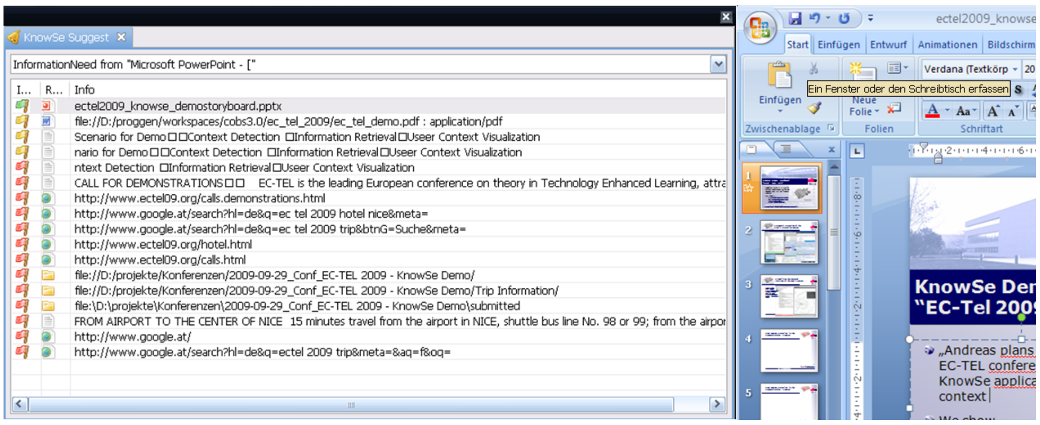


Figure 6.12: This figure highlights the KnowSe's tray icon notification in the bottom right area that notifies the user about the detection of her current task.

6.5.3 Context-Aware Proactive Information Delivery

For users it is worthwhile to automatically have the currently needed resources at hand because it reduces the time spent on searching and navigating through their file collections [Rath *et al.*, 2007]. In order to deliver these valuable resources their work patterns are utilized for context-aware information retrieval [Fuhr, 2005]. A proactive context-aware delivery agent on the basis of the automatically observed user interaction context was realized that retrieves relevant resources from the UICO. The automatically observed user's interactions constitute the base for discovering information needs. An information need has a priority that indicates the need for fulfillment. The priority is based on the accuracy of the detection mechanism and to what extend the fulfillment is time-critical. The value takes the last accessed resource into account in order to broaden and to narrow the search space as well as to enhance the ranking of the delivered resources. The overall objective of just-in-time information delivery is that users actually use more information than they would with search engines since there is no additional effort in obtaining resources, e.g., finding key words or entering the query into a search box. The effortless accessibility enables the users to incorporate additional resources into their work and thus improving the overall work quality and performance. The same policy as JITIR agents [Rhodes, 2000] is followed, i.e., proactively, presentation of retrieved information in an accessible yet non intrusive manner and context-awareness. The *KnowSe Suggest* prototype is shown in Figure 6.13. The visual components of the prototype were developed together with Georg Kompacher. For further details about information need detection and fulfillment the reader is referred to [Rath *et al.*, 2007].

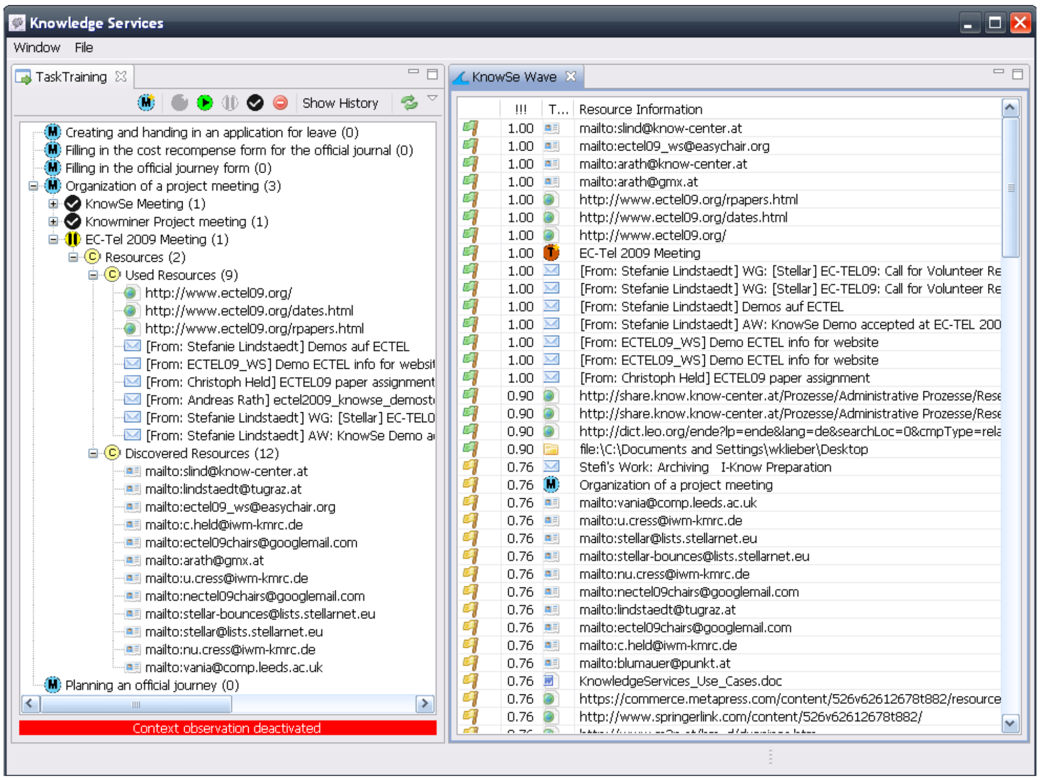


InformationNeed from "Microsoft PowerPoint - ["

I...	R...	Info
		ectel2009_knowse_demostoryboard.pptx
		file://D:/proggen/workspaces/cobs3.0/ec_tel_2009/ec_tel_demo.pdf : application/pdf
		Scenario for Demo☐☐Context Detection ☐Information Retrieval☐Useer Context Visualization
		nario for Demo☐☐Context Detection ☐Information Retrieval☐Useer Context Visualization
		ntext Detection ☐Information Retrieval☐Useer Context Visualization
		CALL FOR DEMONSTRATIONS☐☐ EC-TEL is the leading European conference on theory in Technology Enhanced Learning, attra
		http://www.ectel09.org/calls.demonstrations.html
		http://www.google.at/search?hl=de&q=ec tel 2009 hotel nice&meta=
		http://www.google.at/search?hl=de&q=ec tel 2009 trip&btnG=Suche&meta=
		http://www.ectel09.org/hotel.html
		http://www.ectel09.org/calls.html
		file://D:/projekte/Konferenzen/2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo/
		file://D:/projekte/Konferenzen/2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo/Trip Information/
		file:\D:\projekte\Konferenzen\2009-09-29_Conf_EC-TEL 2009 - KnowSe Demo\submitted
		FROM AIRPORT TO THE CENTER OF NICE 15 minutes travel from the airport in NICE, shuttle bus line No. 98 or 99; from the airpor
		http://www.google.at/
		http://www.google.at/search?hl=de&q=ectel 2009 trip&meta=&aq=f&oq=

Figure 6.13: This figure displays the proactive context-aware information retrieval prototype KnowSe Suggest utilizing the user interaction context for detecting the user's information needs including the information need fulfillment (resource delivery) in the left area. The right area visualizes a part of the Microsoft PowerPoint slide which was the source of the detected information need.

The UICO with its concepts and relations can also be seen as a big graph thus enabling the utilization of graph based algorithms for context-aware information retrieval. *KnowSe Wave* is based on the UICO and utilizes spreading activation algorithms to identify relevant resources, tasks and task models. Spreading activation [Anderson, 1983] has its roots in the cognitive psychology and is used as an explanatory model to understand the operation method of the human brain. In the cognitive psychology the idea is about neurons and synapses which interconnect the neurons. In the case of the *KnowSe Wave* application (i) the concepts represent the neurons and (ii) the synapses represent the relationship between two concepts (object type properties). The *KnowSe Wave* application is visualized in Figure 6.14. The visual components of the prototype were developed together with Georg Kompacher. The mechanisms and algorithms for identifying relevant concepts of the UICO were contributed by Kompacher [2010].



```
Creating and handing in an application for leave (0)
Filling in the cost recompense form for the official journal (0)
Filling in the official journey form (0)
Organization of a project meeting (3)
    KnowSe Meeting (1)
    Knowminer Project meeting (1)
    EC-Tel 2009 Meeting (1)
        Resources (2)
            Used Resources (9)
                http://www.ectel09.org/
                http://www.ectel09.org/dates.html
                http://www.ectel09.org/rpapers.html
                [From: Stefanie Lindstaedt] Demos auf ECTEL
                [From: ECTEL09_WS] Demo ECTEL info for websi
                [From: Christoph Held] ECTEL09 paper assignment
                [From: Andreas Rath] ectel2009_knowse_demost
                [From: Stefanie Lindstaedt] WG: [Stellar] EC-TELO
                [From: Stefanie Lindstaedt] AW: KnowSe Demo a
            Discovered Resources (12)
                mailto:slind@know-center.at
                mailto:lindstaedt@tugraz.at
                mailto:ectel09_ws@easychair.org
                mailto:c.held@iwm-kmrc.de
                mailto:ectel09chairs@googlemail.com
                mailto:arath@gmx.at
                mailto:u.cress@iwm-kmrc.de
                mailto:nectel09chairs@googlemail.com
                mailto:stellar-bounces@lists.stellarnet.eu
                mailto:stellar@lists.stellarnet.eu
                mailto:nu.cress@iwm-kmrc.de
                mailto:vania@comp.leeds.ac.uk
Planning an official journey (0)
```

	!!!	T...	Resource Information
🟩	1.00	▤	mailto:slind@know-center.at
🟩	1.00	▤	mailto:ectel09_ws@easychair.org
🟩	1.00	▤	mailto:arath@know-center.at
🟩	1.00	▤	mailto:arath@gmx.at
🟩	1.00	●	http://www.ectel09.org/rpapers.html
🟩	1.00	●	http://www.ectel09.org/dates.html
🟩	1.00	●	http://www.ectel09.org/
🟩	1.00	T	EC-Tel 2009 Meeting
🟩	1.00	✉	[From: Stefanie Lindstaedt] WG: [Stellar] EC-TEL09: Call for Volunteer Re
🟩	1.00	✉	[From: Stefanie Lindstaedt] WG: [Stellar] EC-TEL09: Call for Volunteer Re
🟩	1.00	✉	[From: Stefanie Lindstaedt] Demos auf ECTEL
🟩	1.00	✉	[From: Stefanie Lindstaedt] AW: KnowSe Demo accepted at EC-TEL 200
🟩	1.00	✉	[From: ECTEL09_WS] Demo ECTEL info for website
🟩	1.00	✉	[From: ECTEL09_WS] Demo ECTEL info for website
🟩	1.00	✉	[From: Christoph Held] ECTEL09 paper assignment
🟩	0.90	●	http://share.know.know-center.at/Prozesse/Administrative Prozesse/Rese
🟩	0.90	●	http://share.know.know-center.at/Prozesse/Administrative Prozesse/Rese
🟩	0.90	●	http://dict.leo.org/ende?lp=ende&lang=de&searchLoc=0&cmpType=rela
🟩	0.90	📁	file:\C:\Documents and Settings\wklieber\Desktop
🟨	0.76	✉	Stefi's Work: Archiving I-Know Preparation
🟨	0.76	M	Organization of a project meeting
🟨	0.76	▤	mailto:vania@comp.leeds.ac.uk
🟨	0.76	▤	mailto:u.cress@iwm-kmrc.de
🟨	0.76	▤	mailto:stellar@lists.stellarnet.eu
🟨	0.76	▤	mailto:stellar-bounces@lists.stellarnet.eu
🟨	0.76	▤	mailto:nu.cress@iwm-kmrc.de
🟨	0.76	▤	mailto:nectel09chairs@googlemail.com
🟨	0.76	▤	mailto:lindstaedt@tugraz.at
🟨	0.76	▤	mailto:ectel09chairs@googlemail.com
🟨	0.76	▤	mailto:c.held@iwm-kmrc.de
🟨	0.76	▤	mailto:blumauer@punkt.at
🟨	0.76	W	KnowledgeServices_Use_Cases.doc
🟨	0.76	●	https://commerce.metapress.com/content/526v62612678t882/resource
🟨	0.76	●	http://www.springerlink.com/content/526v62612678t882/

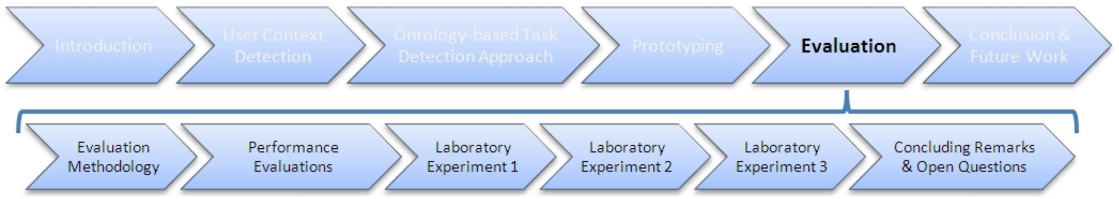
Figure 6.14: This figure displays the proactive context-aware information retrieval prototype KnowSe Wave utilizing the graph structure of the user interaction context ontology for identifying relevant concepts (e.g., resources, tasks, task models). The KnowSe Wave view is on the right side and shows (i) a flag indicating the relevance of a concept, (ii) the activation value as a numerical value for the relevance of a concept, (iii) the type of the suggested concept and (iv) a short description of the identified concept.

6.6 Summary

This chapter gave an overview of the prototyping efforts that were part of this research. It showed the first prototype which was built during the Dyonipos project. A global overview of the architecture of the second prototype referred to as *KnowSe* including the *Context OBservation Service* (COBS) which was used in the latter three laboratory experiments was given. The *Context OBservation Evaluation Toolkit* (COBET) developed in this research for evaluating the performance of the ontology-based task detection approach including a comparison with existing task detection approaches was presented. A presentation of the prototype applications created on top of COBS in the frame of the *KnowSe* prototype rounded off this chapter.

7

Evaluation of the Ontology-Based Task Detection Approach



This chapter's purpose is to present the evaluations of the ontology-based task detection approach performed on three independent datasets obtained from three large-scale laboratory user experiments. In Section 7.2 the methodology of the evaluations is explained. Section 7.3 describes the measures used to assess the performance of the ontology-based task detection approach. The design and execution of the experiments as well as the conducted task detection evaluations are explained for the laboratory experiment 1 in Section 7.4, for the laboratory experiment 2 in Section 7.5, and for the laboratory experiment 3 in Section 7.6. Section 7.7 elaborates on the attempt of generalizing the findings and concludes this chapter with open questions.

7.1 Introduction

For evaluating a task detection approach a dataset consisting of user context data of recorded task executions is required. The more datasets are used to evaluate the task detection approach the more reliable the findings are. Ideally, evaluation datasets consist of usage data (i) of different task executions, (ii) different tasks (task models/task categories), (iii) from various users, (iv) from various domains as well as (v) contains no or less noise. Such kind of datasets are not available by the knowledge of the author. Furthermore there are no standard datasets available for the evaluation of the task detection approaches.

Datasets collected and utilized in other task detection approaches contains usage data from only 1 user to 4 users of only a single domain (see Section 4.4. Examples of tasks are *"buying a book"* or *"final review"* tasks [Oliver *et al.*, 2006], business tasks [Lokaiczyk *et al.*, 2007], or tasks as in [Granitzer *et al.*, 2008] including *"email handling"*, *"paper writing"*, *"research"*, *"documentation"* or *"information collection"*.

7.2 Evaluation Methodology

The methodology chosen for evaluating the proposed ontology-based task detection approach (see Chapter 5) is explained based on the following aspects: (i) experiment design and dataset collection, (ii) level of training (class) instance construction and (iii) performance measurements.

7.2.1 Experiment Design and Dataset Collection

Since no standard public datasets are available for evaluating task detection approaches, three independent laboratory experiments with several users from two different domains executing different kind of tasks were designed and performed. By the nature of laboratory experiments the experimenters have control of the experimental setting and conditions. This power of control is useful in laboratory experiments to (i) reduce the possibilities of the participants to introduce noise into the recorded task usage data as well as (ii) varying the experiments conditions (experiment design) and controlling the execution of the experiment. In the experiments of this dissertation research effort different conditions were varied, e.g., the task itself, the type of task or the computer environment. In the execution of the experiment, the sequence in which the participants executed their tasks were also randomized to reduce bias as well.

From the author's previous work in the Dyonipos project reported in [Granitzer *et al.*, 2009a, 2008] regarding dataset collection in real-world settings following experiences were gained:

Free User Task Labeling: The labeling of the task was done by users during their task executions. The users were free in how they labeled their tasks but were limited in terms of providing a description about the task. There was no possibility for the users to add a description for a task. In the experiment evaluation the labels by an expert were used to manually group the tasks into *task clusters*. Although the task clusters were coarse granular, the manual grouping could have introduced a bias because of the lack of knowledge about the executed tasks originating from the missing task descriptions. In the performed laboratory experiments of this dissertation research a set of tasks belonging to a specific type of task or task classification of a domain were selected during workshops by domain experts in order to reduce the bias. Furthermore real domain experts executed the selected tasks in a controlled setting.

Noise in the Usage Data: The tasks were collected from users during their real work in the user's domain, i.e., "in the wild of a working day". This might have introduced unpredictable noise and false task labels in the observed usage data. External interruptions, e.g., people entering the room, phone calls, leaving the desk, checking or receiving emails and instant messages etc., or switching to another task without telling the recording tool that a switch has happened are typical examples for noise in the observed usage data.

Further reasons why laboratory experiments were chosen for the dataset collection over real-world settings, were the lack of existing datasets from laboratory experiments with more than two users. It is also very difficult to vary experiment conditions in a real-world setting.

7.2.2 Level of Training/Class Instance Construction

As summarized in Section 4.5 existing task detection approaches vary in terms of how they construct training/class instances for the machine learning algorithms. Possibilities are a sliding window approach, a certain time interval (5 to 300 seconds) or grouping all interactions with a single resource ("event block level"). These approaches have limitations for finding the best discriminating features for tasks because a training instance corresponds to just a part of a task and not to a single task. Hence, these approaches study the discriminative power of features among parts of tasks and not the whole tasks themselves. The same applies to the performance of the studied machine learning algorithms. However, for real-time task detection it can be useful to know which features discriminate parts of tasks best. In case of this dissertation research, one of the goals is to find the best discriminative feature between tasks and hence it is argued to build the training instances on a task level, i.e., one task execution corresponds to one training/class instance.

From the author's previous work in the Dyonipos project reported in [Granitzer *et al.*, 2009a, 2008] and the author's follow up work regarding the level of the training/class instance construction used, it seems that event block level training instance construction can introduce a not unimportant bias in the task detection evaluation results:

Event Block Level Training Instance Construction: Task detection on the event block level with stratified 10-fold cross-validation has limits when interpreting the detectability of a task. If there are long lasting tasks or tasks with a lot of event blocks then the chance of randomly choosing one or more event block/s of the same task in a cross-validation step is high. Event blocks of the same task, especially event blocks constructed based on the interaction with the same resource, are easy to detect because of the similar content or the same window title. Evaluations showed that when taking away the event block boundaries and constructing events based on a sliding window approach an arbitrarily high or low accuracy of task detection can be reached through varying the size of the window.

A variable window size relates to (i) the rules how event blocks are constructed and (ii) to the user's interaction behavior with applications and resources. Since event block rules group events based on the interaction with a single resource, the user's behavior directs the number of constructed event block based on the user's resource switching behavior. If the user switches a lot between multiple resources, small event blocks are the outcome. In contrary, if the user interacts with a single resource for a long time, only one single event block is created. Rephrasing this effect in terms of a machine learning problem means that the number of resulting training/class instances for a class could vary based on the task and could be kind of "unbalanced" (number of events per event block). In both cases, (i) if the event blocks are too small or too large[1] and (ii) if a task has a big amount of or just a few event blocks, it influences the resulting task detection accuracy.

[1]Here "large" refers to the number of events per event block

For evaluating the detectability of tasks and finding well discriminating context features, a construction of training instances on the task level, i.e., the construction of a training instance for the classifier based on a single task execution, seems to be more reliable than an "event block level" construction or a sliding window approach.

7.2.3 Performance Measurements

Task detection is classically seen as a machine learning problem in task detection research. In case of this dissertation research task detection is seen as a classification problem because supervised machine learning algorithms, more specifically classification algorithms, are studied for their applicability for the task detection challenge. Standard machine learning measure [Witten & Frank, 2005] such as *accuracy*, *precision*, *recall* and the *f1 measure* are suggested to measure the performance of the task detection results by the task detection literature. This dissertation goes along with their suggestions and calculates standard machine learning measures as described in Section 7.3.

7.3 Performance Evaluations

In all the ontology-based task detection experiments the performance of well-known learning algorithms in the area of text classification were studied The same evaluation methodology was applied to ensure comparability across the results from the evaluations of the different experiment's datasets. The ontology-based task detection approach is referred to as the *UICO* approach.

7.3.1 Learning Algorithms

Evaluated learning algorithms were the Naïve Bayes (NB), Linear Support Vector Machine (SVM) with cost parameter $c \in \{2^{-5}, 2^{-3}, 2^{-1}, 2^{0}, 2^{1}, 2^{3}, 2^{5}, 2^{8}, 2^{10}\}$, J48 decision tree (J48) and k-Nearest Neighbor (KNN-k) with $k \in \{1, 5, 10, 35\}$ algorithms. The interval for the cost parameters for the SVM were chosen according to the libSVM practical guide at [Chang & Lin, 2001]. The variations of the number of neighbors k were introduced in order to explore the task detection performance with different decision boundaries for the KNN learning algorithms.

7.3.2 Attribute Selection

For each classifier/learning algorithm $l \in L$, for each feature category $\phi \in \Phi$ and each feature $f \in F$ the g attributes having the highest *Information Gain* (IG) value to obtain the dataset were selected. Information gain attribute selection was used because (i) it is one of the most popular and fastest algorithms in text classification and because (ii) "pre-evaluations" with more advanced attribute selection methods showed that the influence of using more advanced ones is rather small. An extensive study of different attribute selection methods was not in focus. As values for g, 50 different measure points were used. Half of them were equally distributed over the available number of attributes with an upper bound of 5000 attributes. The other half was defined by $G = \{3, 5, 10, 25, 50, 75, 100, 125, 150, 175, 200, 250, 300, 500, 750, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 5000, 7500, 10000\}$. If the maximum number of attributes of a dataset was less than 10000 attributes than the performance was evaluated on the maximum number of attributes available. On the other hand if the maximum number of attributes of the dataset exceeded 10000 attributes then 10000 attributes constituted the last measuring point. The interval as well as the special measuring points were chosen because of two reasons: (i) several evaluations of task detection performance research reported that they achieved good results for a lower number of attributes (200-300 attributes) [Granitzer *et al.*, 2008; Lokaiczyk *et al.*, 2007; Shen *et al.*, 2006], and hence a special focus was put on these number of attributes and (ii) to also investigate the performance of the learning algorithms for a higher number of attributes.

From a practical evaluation point of view only a few measuring points were chosen because (i) the training of classifiers with a high number of features take a long time, (ii) requires high computational power and (iii) is not possible to perform on standard desktop computers in a productive scenario. The focus was to find a good combination of classifiers and features that work well on a standard desktop computer. Which attributes were finally used for training the classifiers depended on the *attribute selection* algorithm (*IG*).

7.3.3 Algorithm Evaluation Methods

Two types of learning algorithm performance evaluations were done: (i) stratified 10-fold cross-validation and (ii) training and test set evaluation. In (i) statistical values for each fold were computed as well as the mean and standard deviation of all values across all folds calculated. In (ii) a training and a test set were constructed based on the investigated research question. Each learning algorithm was trained on the training set and evaluated on the test set. The training and test set instances were strictly parted which means constructed and preprocessed independently to ensure that there was no bias or influence what so ever.

The Weka machine learning toolkit [Witten & Frank, 2005] provided the necessary tools set to measure the performance of the Naïve Bayes (NB), Linear Support Vector Machine (SVM), J48 decision tree (J48) and k-Nearest Neighbor (KNN-k) algorithm. The Weka integration [EL-Manzalawy & Honavar, 2005] of the libSVM [Chang & Lin, 2001] allowed hereby the evaluation of a SVM as well.

7.3.4 Algorithm Performance

In all the experiments the influence of five parameters on the task detection performance were evaluated: (i) the number of features, (ii) the classification model, (iii) the feature category, (iv) single features and (v) the combination of the *Top k* best performing single features with $k \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20\}$. For each research question of each experiment's dataset the task detection performance of the feature and feature combinations of the *UICO approach* was compared to the features and feature combinations of the existing approaches of Dyonipos [Granitzer *et al.*, 2008], *SWISH* [Oliver *et al.*, 2006] and *TaskPredictor 1* [Shen, 2009; Shen *et al.*, 2006]. For the comparison with the three already existing approaches the features and the corresponding feature preprocessing steps were implemented based on the description and details given in the published papers. For the Dyonipos and *TaskPredictor 1* also the same stopword list was used in the feature preprocessing steps. For SWISH a new stopword list has been created based on the description in [Oliver *et al.*, 2006].

7.3.5 Dominance Matrices and Significance Tests

This section describes the dominance matrix computations and the significance tests. The computation of dominance matrices for the feature/feature combinations and classifiers were performed. Significance tests were done in order to produce a statistically significant ranking for (i) features/feature combinations as well as (ii) for classifiers for each experiment's dataset separately.

7.3.5.1 Finding the Best Features/Feature Categories

Studying the best features and feature combinations was done twofold: (i) *dominance matrices* and (ii) *paired t-tests* [Bortz & Döring, 2006]. Similar cross-dataset comparison methods were proposed by [Yang & Liu, 1999] for learning algorithm evaluation for text categorization and by [Zhao *et al.*, 2005] for hierarchical clustering algorithm evaluation for multiple document datasets.

Feature Dominance Matrix: The *feature dominance matrix* shows how often a feature/feature combination outperforms another one. The rows and columns of this matrix correspond to the features/feature combinations whereas its values correspond to the number of times the feature/feature combinations of the row outperforms the one in the column. A feature/feature combination outperforms another one according to the following order: (i) higher accuracy (ii) higher micro precision, (iii) higher micro recall and (iv) lower number of attributes.

Paired T-Tests for Features/Feature Categories: The *paired t-test* tests if the performance of a features/feature combination was statistically significantly better than another one. Testing was performed with the *Apache Commons Mathematics Library* [The Apache Software Foundation, 2009] for the three significance levels $\alpha = 0.005$, $\alpha = 0.01$ and $\alpha = 0.5$. The following notion explains the symbols used to indicate the different significance levels in tables:

- $\gg$ indicates that the feature/feature combination in the row achieved a significantly higher value than the feature/feature combination in the column with a significantly level of $\alpha = 0.005$
- $\triangleright$ indicates that the feature/feature combination in the row achieved a significantly higher value than the feature/feature combination in the column with a significantly level of $\alpha = 0.01$
- $>$ indicates that the feature/feature combination in the row achieved a significantly higher value than the feature/feature combination in the column with a significantly level of $\alpha = 0.05$
- $\ll$ indicates that the feature/feature combination in the column achieved a significantly higher value than the feature/feature combination in the row with a significantly level of $\alpha = 0.005$
- $\triangleleft$ indicates that the feature/feature combination in the column achieved a significantly higher value than the feature/feature combination in the row with a significantly level of $\alpha = 0.01$
- $<$ indicates that the feature/feature combination in the column achieved a significantly higher value than the feature/feature combination in the row with a significantly level of $\alpha = 0.05$
- $\sim$ indicates that there is no significance at any of the three levels $\alpha \in \{0.005,0.001,0.05\}$, i.e., the p-value [Bortz & Döring, 2006] $p \geq 0.05$.

Ψ the number of times the feature/feature combination outperformed other features/feature combinations at a $\alpha = 0.05$ significance level. (This value is *not* corresponding to the sum of the values of the row of the tables because only the top 15 features are listed.)

The paired t-test for features and feature combinations was computed with and without rank transformation. When rank transformation was performed the test is referred to as $T - Test_{rt}^{f}$ and $T - Test^{f}$ otherwise. The significance tests with rank transformation were done to reduce the influence introduced by the difference of the performance values when comparing across datasets.

Rank Transformation: The rank transformation [Sachs & Hedderich, 2006] computes the ranks of each value of two vectors. The values of both vectors are pooled together and sorted. Each value is substituted with the appropriate rank based on its position in the sorting. If two values are the same then this situation is called *binding* and is handled in the following way: Let v_p be a value v of one of the vectors, p the position in the sorted array and v^r the rank r of the value v. The same values $v_j, v_{j+1}, \ldots, v_{j+i}$ will result in $v_j^r, v_{j+1}^r, \ldots, v_{j+i}^r$ with $r = \frac{\sum_{k=0}^{i} j+k}{j+i}$.

7.3.5.2 Finding the Best Learning Algorithms

Dominance matrices and significance tests were computed for comparing the classifier's performances on each experiment's dataset. These two methods are explained in this section and the results are discussed in the respective experiment's sections.

Learning Algorithm Dominance Matrix: The *learning algorithm dominance matrix* shows how often the best run of a learning algorithm outperforms another one for all features, feature combinations, and *Top k* single performing feature combinations. The rows and columns of this dominance matrix correspond to the learning algorithms whereas its values correspond to the number of times the learning algorithm of the row outperforms the one in the column. A learning algorithm outperforms another one according to the following order: (i) higher accuracy (ii) higher micro precision, (iii) higher micro recall and (iv) lower number of attributes.

Paired T-Tests for Learning Algorithms: The *paired t-test* tests if the performance of a classifier was statistically significantly better than another one. Testing was performed with the *Apache Commons Mathematics Library* [The Apache Software Foundation, 2009] for the three significance levels $\alpha = 0.005$, $\alpha = 0.01$ and $\alpha = 0.5$. The paired t-tests were also computed based on the micro f-measures for having a second view on the classifier performance next to the accuracy measure. The following notion explains the symbols used to indicate the different significance levels in tables:

$\gg$ indicates that the classifier in the row achieved a significantly higher value than the classifier in the column with a significantly level of $\alpha = 0.005$

$\triangleright$ indicates that the classifier in the row achieved a significantly higher value than the classifier in the column with a significantly level of $\alpha = 0.01$

$>$ indicates that the classifier in the row achieved a significantly higher value than the classifier in the column with a significantly level of $\alpha = 0.05$

$\ll$ indicates that the classifier in the column achieved a significantly higher value than the classifier in the row with a significantly level of $\alpha = 0.005$

$\lhd$ indicates that the classifier in the column achieved a significantly higher value than the classifier in the row with a significantly level of $\alpha = 0.01$

$<$ indicates that the classifier in the column achieved a significantly higher value than the classifier in the row with a significantly level of $\alpha = 0.05$

$\sim$ indicates that there was no significance at any of the three levels $\alpha \in \{0.005,0.001,0.05\}$, i.e., the p-value [Bortz & Döring, 2006] $p \geq 0.05$.

Ψ the number of runs the classifier outperformed other classifiers.

The paired t-tests for the learning algorithms were computed with and without rank transformation (see Section 7.3.5.1). When rank transformation was performed the test is referred to as $T - Test^c_{rt}$ and $T - Test^c$ otherwise. The significance tests with rank transformation were done to reduce the influence introduced by the difference of the performance values when comparing across multiple datasets.

7.4 Laboratory Experiment 1 - Know-Center GmbH.

The *task experiment 1* is an experiment to get insights into the capabilities of automatic context detection in a controlled setting. In this experiment task executions were recorded on a single laboratory computer and on the personal employees' workstations. 14 subjects participated in the experiment and made their usage data available for evaluation. The experiment lasted for about two weeks.

Following questions were investigated:

- What are the requirements and conditions for the users and the software to for a user interaction context observation experiment?
- Can the task model of a task instances be automatically detected?
- Can task models of task instances from personal workstations be detected based on laboratory task executions for training the classifier?
- Can task models of task instances from personal task executions be detected based on predefined standard task executions for training the classifier?
- Is there a difference in automatically detecting tasks on a laboratory computer or on a personal workstation?
- Can the type of task be automatically detected when distinguishing routine and knowledge-intensive tasks?
- Can the task model of a task instance be automatically detected based on task instances from only one expert user?
- Can the task model of a task instance of a single user be automatically detected based on task instances from multiple expert users?
- Which context features are most discriminative for the studied tasks?
- Which learning algorithm performs best in terms of automatic task detection on the collected dataset?

7.4.1 Experiment Design

The comparison was *within subjects* [Bortz & Döring, 2006] and the manipulations were achieved by (i) the computer environment (laboratory or personal workstation), (ii) the type of task (standard or personal) and (iii) the task to be executed (5 different tasks). The experiment was designed in three phases. Phase 1 was the phase before the execution of the recording of user interaction context observations. Phase 2 was the user interaction context observation phase. This phase was followed by Phase 3, which included the evaluation of the task detection approaches of the proposed UICO approach and the existing approaches based on the recorded task usage dataset. A description of the steps of the phases of the experiment as well as the obtained results are discussed in the following sections.

Manipulation 1: Laboratory and Personal Workstation

The first manipulation was achieved by varying the work environment, i.e., the computer desktop environment the experiment's participants utilized to perform the tasks. Half of the participants started performing the tasks on a *laboratory computer* on which a set of standard software used in the company was installed. The other half began working on their company's *personal workstation* with their personal computer desktop settings and access to their personal files, folder, bookmarks, emails and so on. The assignment of the experiment's participants to "starting at the laboratory computer" or "starting at the personal workstation" was randomized.

Manipulation 2: Standard and Personal tasks
Tasks that have a specific goal are referred to as *standard tasks.* An example of a *specific goal* is "Bill Adams plans a journey to the CHI 2010 conference". By having multiple users executing a task with the same specific goal very similar task instances were expected. On behalf of an artificial person (*persona*) called "Bill Adams" who also worked in the same company the tasks were executed (standard tasks). The tasks were also performed on behalf of the experiment's participants themselves (*personal tasks*). The order to start with a *standard* or a *personal task* was randomized.

Manipulation 3: Tasks
The third manipulation resulted from varying the tasks themselves. Five tasks were studied. In a proceeding workshop the participants of the experiment agreed on the selection of five tasks typically for the company to execute during the experiment. The tasks had different characteristics, like for example, complexity, estimated execution time, number of involved resources or granularity. A short questionnaire was issued before starting the experiment to make sure that the subjects understood the tasks to perform. Another reason for this questionnaire was to have the subjects think about the tasks before they actually started executing them.
The five task models are listed bellow. A detailed description of the task models and example tasks is given in Section 8.4.

1. *Routine Tasks:*

Task 1: Filling in the official journey form
Task 2: Filling in the cost recompense form for the official journal
Task 3: Creating and handing in an application for leave

2. *Knowledge-intensive Tasks:*

Task 4: Planning an official journey
Task 5: Organization of a project meeting

The order in which the subjects were asked to execute the tasks was randomized. The fifth task *"Organization of a project"* meeting was only executed on the personal workstation and not on the laboratory computer. The reason was the outcome of the held workshop. The experiment's participants agreed that they would only be able to perform this kind of task if they had access to their personal files, bookmarks, as well as their email, notes and calendar program. However, it was decided from the experiment design perspective to keep this task since it was such a typical work task for the studied domain.

Example of a task model:

[Task 1] Filling in the official journey form

Task Model (Description):
The employee fills in the details about the planned journey in the official journey form, prints it and gives it to her division manager.

Task Instance (Task Example):
Suppose you are a researcher named Bill Adams, who is working in the Knowledge Services division on the KnowSe project. You are traveling to the Computer-Human Interaction (CHI 2009) conference in Boston, USA, with a colleague of yours. The conference starts at the 4th April 2009 and ends at the 9th April. Your secretary has already compiled some information about your trip. Here are the details:

- Project *KnowSe*
- Your division: Knowledge Services
- Your name: Bill Adams
- Your colleague's name: Mary Jones
- Flight 3rd April from Graz (Austria) to Frankfurt (Germany) at 06:00
- Flight 3rd April from Frankfurt (Germany) to Boston (USA) at 12:25
- Flight 10th April from Boston (USA) to Frankfurt (Germany) at 21:40
- Flight 11th April from Frankfurt (Germany) to Graz (Austria) at 09:15
- CHI 2009 conference fees are 1000 Euros per person
- Hotel costs are 500 Euros
- Traveling costs are 950 Euros per person (flight and bus)
- All expenses will have to be payed in advance by yourself and will be refunded after the journey.

7.4.1.1 Laboratory Experiment 1 - Phase 1 - Before the Experiment

Phase 1 included the analysis of requirements for a research experiment in which user interaction context data is automatically observed from the user's computer desktop. The instrument of the requirement analysis was informal interviews. The interviews, carried out with 8 subjects, had two parts, an explorative part and a semi-structured part. In the explorative part following question was asked *"What are the conditions for you to contribute the data about your user interaction context for a research experiment?"*. It was intended to ask such an open question to not limit or lead the subjects in any direction. The semi-structured interview part dealt with 19 specific areas whereas 11 areas focused on the design requirements and the setup for the experiment. The questions regarding the software requirements focused on 8 specific areas that generated the most discussions in the workshop.

7.4.1.1.1 Requirement Analysis

Requirements for the design of the experiment:

- In advance information about the capabilities of the context sensors as well as the aggregation and representation of the user interaction context.
- A formal or an informal agreement to the specific purposes of the analysis of the captured usage data during the experiment.
- The goal of the data gathering and the evaluation has to be clear to the subjects and must not be modified without explicit allowance of the subjects.
- A little manual user effort on the subject's side is ok during the period of the experiment. Especially the creation, the start, the suspending, the deletion of task instances is ok. Furthermore it is ok that the subject's computer is a bit slower than normally during the experiment.
- The captured data is only allowed to be distributed among a predefined circle of persons. This circle has to be defined before the experiment.
- The evaluation results have to be made anonymous.
- Manual inspection and automatic evaluation of the captured user interaction context is allowed. Manual inspection means that the predefined circle of persons is approved to look at every single item of the captured user interaction context. Automatic evaluation here stands for statistic, heuristic and algorithm based evaluations of the captured usage data.

The software requirements for the prototype used in the experiment were derived from the explorative and the semi-structured parts of the interviews. These are listed below in Table 7.1 and sorted based on the number of times the subjects named the specific requirement in the interviews. All the requirements for the design of the experiment and for the used prototype were followed. Furthermore all wishes of the subjects mentioned during the interviews were carefully respected.

Context Sensing and User Interaction Context Information
Phase 1 further included the compilation of an informative document that described the capabilities of the context sensors, i.e., what they observe about the user's behavior, the representation of the user interaction context as the result of the sensor data processing and a short description about the architecture behind the software used in the experiment. This document was compiled and distributed to the experiment's subjects before the start of the user interaction context observation phase.

Two Week Testing Period
A two week software testing period was planned to allow the subjects to get familiar with the prototype and to reduce the bias of insecurity and unfamiliarity in handling the software during the experiment. This period also gave the subjects the possibility to ask questions and to get clarification about prototype handling issues. The *KnowSe* prototype was used in this experiment. It is described in Section 6.3 as well as visualized in Figure 6.2 and Figure 6.3.

7.4.1.2 Laboratory Experiment 1 - Phase 2 - User Interaction Context Observation

The test design was a *within subjects design* [Bortz & Döring, 2006] which means that each subject carried out the experiment for every condition of the tasks. This test design was chosen because

SR	**Description**	**#Votes**
SR1	Information about the data that will be recorded (sensor capabilities)	8
SR2	Displaying the data that has been observed from my usage behavior	7
SR3	Deleting data stored about my user behavior on a concept and triple level	6
SR4	Switching the context observation on and off	5
SR5	Performance criterion: Software should allow reasonable working speed with the computer	4
SR6	Knowing/Showing which data will leave my computer	3
SR7	No automatic transfer of the usage data to the server /asking user before submission	3
SR8	No mixing of my usage data with the data of other users (clear separation)	2
SR9	Clear indication when context observation is active or when not	1
SR10	Software uninstallation routine (everything has to be cleanly removed after the experiment)	1
SR11	Searching and navigating through the recorded and derived user interaction context data (concept and triple level)	1
SR12	Local storage of observed usage data in log files	1
SR13	Easy installation	1

Table 7.1: This table shows the software requirements (SR) derived from explorative and semi-structured interviews with the 8 subjects. The last column (#Votes) indicates the number of people who mentioned the software requirement stated in the row as an important one.

of the number of subjects available (14 subjects). With this test design it was also possible to illuminate *dependency* and *difference hypothesis* [Bortz & Döring, 2006].

In phase 2 the subjects executed the *Tasks 1-4* on a remote computer with a standard software installation (Know-Center standard software installation) and the *Tasks 1-5* on their own company computer system. The subjects were asked to execute an example task for each task model and a free task in respect to the corresponding task model. To reduce the bias of a training effect, half of the subjects started on the remote computer and the other half on their company computer. The selection of the subjects who started on which environment and the order in which the subjects were asked to execute the tasks were randomized to reduce bias and the *training effect* [Bortz & Döring, 2006]. The reasons why the subjects had to execute the tasks on different computers were that the computer system with the standard software installed provided a more controlled environment and reduced disturb factors. Examples for such possible factors are incoming instant messaging requests, Skype calls, personal information management issues, or new email notifications.

A further reason for the different task execution environments was the ability to test the hypothesis that the task execution on a familiar and non-familiar computer environment differs. Possible differences expected were for example the task length, the number of resources used, the number or types of user interactions, navigation patterns and so on. The explorative study of the observed user interaction context data led to more discriminating criteria between tasks.

Task 5 which was about the organization of a project meeting was special because it was the most complex and the longest task for the subjects. Since it was not easy to complete *Task 5* without the subject's personal computer environment, i.e., files, folders, email etc., this task was only executed on their personal workstations. This decision was made beforehand in the task selection workshops by the domain experts.

7.4.1.3 Laboratory Experiment 1 - Phase 3 - Task Detection

The evaluation and the analysis of the data plays the main part of phase 3 of the experiment. The well known machine learning toolkit *Weka* [Witten & Frank, 2005] in combination with the Weka integration of the *libSVM* [Chang & Lin, 2001] were utilized to study appropriate parameters and algorithms for attribute selection and automatic task detection performance. The training instances for the learning algorithms were built based the user interaction context ontology (UICO) as described in Section 5.

7.4.2 Research Question: Can the task model of the task instances be automatically detected?

The goal of this evaluation was to answer the question *"Can the task model of the task instances be automatically detected?"*. The dataset on which this question was investigated contained 218 tasks from 14 users. The same evaluation was done in [Rath *et al.*, 2009d] except the fact that two task instances were removed from the 220 tasks dataset because of their short duration of the user interaction context observation for these two tasks. These tasks had less than three event blocks.

The distribution of the task instances in respect to the classes and the computer environment is shown in Table 7.2. Laboratory workstation tasks and personal workstation tasks were not distinguished in this evaluation but are listed in Table 7.2 to provide the reader with a complete picture about the task distribution in the dataset. An overview of all results about the performance of detecting the tasks (*Task 1-5*) is given in Table 7.3. The results were achieved by applying stratified 10-fold cross-validation on the training instances. A training instance was built for each task instance independently.

Classes	**Laboratory Workstation**	**Personal Workstation**	**Sum**
Task 1	30	25	55
Task 2	26	19	45
Task 3	26	25	51
Task 4	24	28	52
Task 5	0	15	15
Dataset CV	106	112	**218**

Table 7.2: This table shows the distribution of the stratified 10-fold cross-validation training/test instances for the different task classes ranging from Task 1 to Task 5 recorded on the laboratory computer and on the personal workstations.

Feature Categories: The feature category which achieved the highest accuracy values was the combination of all 50 features of all categories (l=J48, a=86.71%, g=750, p=0.96, r=0.85). Close behind with the same algorithm was the *application category* with the same accuracy but with a 0.01 lower micro recall value with 50 attributes. The *resource category* obtained an accuracy of 66.49% (l=NB, g=3000, p=0.89, r=0.72) which was only sufficient for the global rank R_G=17. The number of attributes of the best runs for the feature categories were between 50 and 4000 attributes.

Single Features: The best performing single feature was the *acc. obj. name* feature (l=J48, a=86.21%, g=50, p=0.96, r=0.86). Only 5.05% less accurate was the *window title* feature with the second highest accuracy (l=J48, a=81.26%, g=75, p=0.94, r=0.81). The *acc. obj. value* feature achieved the third rank of the best performing single features with an accuracy of 71.15% on 25 attributes (l=J48, p=0.90, r=0.69) with the same algorithm as the top two single performing features. The range of the numbers of attributes for the best runs of the top

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	J48	750	86.71	0.96	0.85	10
	2	Application Cat.	J48	50	86.71	0.96	0.84	11
	3	Resource Cat.	NB	3000	66.49	0.89	0.72	17
	4	Content Cat.	NB	4000	65.15	0.87	0.65	18
	5	Ontology Str. Cat.	J48	359	63.38	0.86	0.64	21
	6	Action Cat.	J48	750	58.23	0.85	0.61	24
	7	Switching Seq. Cat.	NB	300	49.59	0.80	0.53	32
Single Feat.	1	acc. obj. name	J48	50	86.21	0.96	0.86	13
	2	window title	J48	75	81.26	0.94	0.81	15
	3	acc. obj. value	J48	25	71.15	0.90	0.69	16
	4	content in focus	J48	1500	64.72	0.87	0.64	19
	5	content of EB	KNN-1	75	63.74	0.87	0.66	20
	6	used res. metadata	J48	125	62.94	0.87	0.65	22
	7	datatype properties	J48	150	62.38	0.86	0.64	23
	8	acc. obj. role des.	J48	5	57.77	0.82	0.52	25
	9	used res. content	NB	100	56.04	0.83	0.57	26
	10	acc. obj. role	NB	10	55.17	0.80	0.56	27
	11	resource content	KNN-5	75	54.63	0.81	0.56	28
	12	applications interact.	J48	69	52.79	0.81	0.56	29
	13	res. types interact.	J48	36	50.93	0.81	0.56	30
	14	concept instances	J48	81	50.37	0.79	0.52	31
	15	E type switch seq.	J48	45	47.21	0.78	0.50	33
Top k Feat.	1	Top $k = 4$	J48	5000	88.55	0.97	0.87	1
	2	Top $k = 5$	J48	1000	88.53	0.97	0.87	2
	3	Top $k = 2$	KNN-10	25	88.12	0.96	0.89	3
	4	Top $k = 20$	J48	500	88.07	0.96	0.87	4
	5	Top $k = 3$	J48	3500	87.66	0.96	0.88	5
	6	Top $k = 7$	J48	300	87.19	0.96	0.85	6
	7	Top $k = 6$	J48	250	87.16	0.96	0.86	7
	8	Top $k = 8$	J48	3000	87.14	0.96	0.84	8
	9	Top $k = 10$	J48	300	86.73	0.96	0.85	9
	10	Top $k = 15$	J48	500	86.28	0.96	0.85	12
	11	Top $k = 9$	J48	300	85.76	0.96	0.83	14

Table 7.3: Overview of the best accuracy values (a) for each feature (f) and the Top k best performing single features from the UICO for detecting the task model (Task 1-5) of the task instances by stratified 10-fold cross-validation. The learning algorithm (l), the number of attributes (g), the micro precision (p) and the micro recall (r) are also given.

15 single performing features was between 5 and 150 attributes except for the feature *content in focus* which had its best run with 1500 attributes. In comparison with the *Top k* feature combinations, all *Top k* feature combinations outperformed each single feature except the worst *Top k* feature combination which was the *Top* $k = 9$.

Top *k* Features: The *Top k* feature combinations achieved accuracy values ranging from 85.76% to 88.55%. The highest accuracy resulted from the *Top k=4* feature combination (l=J48, a=88.55%, p=0.97, r=0.87). The range of the number of attributes of the best runs of the classifiers of the *Top k* feature combinations were between 25 and 5000 attributes.

Comparison with existing approaches: A comparison with the existing approaches showed that the best overall tested feature combination of the UICO features, the *Top k=4* feature combination (l=J48, a=88.55%, p=0.97, r=0.87), outperformed all the existing approaches by at least 5.04% in terms of accuracy. The *SWISH* , the *TaskPredictor 1* and the best Dyonipos feature combination $\mathcal{ACW}$ were outperformed by 9.2%, 9.13% and 5.04% respectively. The *SWISH* approach had its best run with 150 attributes and the J48 algorithm (a=79.35%, p=0.93, r=0.78). The *TaskPredictor 1* approach also performed best with the J48 algorithm (a=79.42%, p=0.94, r=0.81) but required 1387 attributes. Among the existing approaches the Dyonipos $\mathcal{ACW}$ resulted in the highest accuracy (l=J48, a=83.51%,p=0.95, r=0.85).

A detailed comparison of the feature and classifier performance evaluation of the existing approaches is given in Table 7.4. The Dyonipos evaluations were also published as part of [Granitzer *et al.*, 2009a].

Set	R_G	f	l	g	a	p	r
	1	$\mathcal{ACW}$	NB	300	83.51	0.95	0.85
	2	$\mathcal{CW}$	NB	750	82.58	0.95	0.85
	3	$\mathcal{AW}$	J48	100	80.28	0.94	0.80
Dyonipos	6	$\mathcal{W}$	NB	100	78.87	0.93	0.80
	7	$\mathcal{AC}$	J48	150	68.27	0.89	0.69
	8	$\mathcal{C}$	KNN-1	125	63.38	0.86	0.63
	9	$\mathcal{A}$	J48	24	48.20	0.78	0.52
SWISH	5		J48	150	79.35	0.93	0.78
TaskPredictor 1	4		J48	1387	79.42	0.94	0.81

Table 7.4: Overview of the best results about the performance of detecting the task model (Task 1-5) of the task instances for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Concluding Remarks: The evaluation results showed that the task model of the task instances were detected with an accuracy of 88.55% on this task dataset. The UICO *Top k=4* best performing single feature combination outperformed all the existing approaches between 5.04% to 9.20%.

7.4.3 Research Question: Can task models of task instances from personal workstations be detected based on laboratory task executions for training the classifier?

The goal of this evaluation was to answer the question *"Can task models of task instances from personal workstations be detected based on laboratory task executions for training the classifier?"*. The dataset contains 203 task instances from 14 users: 106 tasks from the laboratory computer and 97 from personal workstations. The distribution of the task instances in respect to the classes (Task 1 to Task 4) as well as in respect to the computer environment is shown in Table 7.5. The same question has also been studied on a similar dataset with additional measuring points in the interval between 5000 and 10000 attributes in [Rath *et al.*, 2009a]. An overview of all results about the performance of detecting real workstation tasks by training on task executions observed on a laboratory workstation is given in Table 7.6.

Classes	**Laboratory Workstation**	**Personal Workstation**	**Sum**
Task 1	30	25	55
Task 2	26	19	45
Task 3	26	25	51
Task 4	24	28	52
Dataset (Train/Test)	106	97	**218**

Table 7.5: This table shows the distribution of training instances (laboratory workstation tasks) and test instances (personal workstation tasks) for the different task classes. Training and test instances were constructed based on the usage data recorded on the laboratory computer and on the personal workstations respectively.

Feature Categories: The best feature category was the application category that correctly identified 91.75% of the real tasks (l=NB, g=500, p=0.97, r=0.92). Approximately 5% behind in terms of accuracy was the content category (l=NB, a=86.60%, g=500, p=0.95, r=0.87). Using all 50 features resulted in a 82.47% accuracy, which was about 9% worse than the best performing feature category.

Single Features: The performance of each single feature was evaluated separately and confirmed that the *window title* [Granitzer *et al.*, 2008; Lokaiczyk *et al.*, 2007; Oliver *et al.*, 2006; Shen *et al.*, 2007] was the best discriminative feature: it obtained an accuracy of 85.57% (l=J48, g=100, p=0.95, r=0.87). Of great interest were the good performances of accessibility object features: the *acc. obj. name* with a=80.41% (l=J48, g=100, p=0.92, r=0.81) and the *acc. obj. value* with a=71.13% (l=J48, g=150, p=0.89, r=0.72). Simply counting the number of UICO datatype relations (a=70.10%) was also quite efficient.

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	Application Cat.	NB	175	90.72	0.97	0.91	3
	2	Content Cat.	NB	50	85.57	0.94	0.85	9
	3	All Categories	NB	750	82.47	0.93	0.83	16
	4	Resource Cat.	NB	4435	68.04	0.87	0.68	21
	5	Ontology Str. Cat.	J48	359	65.98	0.86	0.67	23
	6	Action Cat.	SVM-$C2^5$	10	59.79	0.81	0.58	25
	7	Switching Seq. Cat.	NB	1500	44.33	0.71	0.44	32
Single Feat.	1	window title	J48	100	85.57	0.95	0.87	7
	2	content in focus	NB	10	84.54	0.94	0.84	11
	3	acc. obj. name	J48	50	80.41	0.92	0.81	17
	4	content of EB	NB	200	73.20	0.89	0.76	18
	5	acc. obj. value	J48	10	71.13	0.89	0.72	19
	6	datatype properties	J48	221	70.10	0.88	0.71	20
	7	used res. metadata	J48	1000	68.04	0.86	0.68	22
	8	used res. content	NB	125	62.89	0.84	0.65	24
	9	resource content	NB	200	58.76	0.81	0.61	26
	10	acc. obj. role	J48	31	54.64	0.79	0.55	27
	11	acc. obj. role des.	NB	25	51.55	0.77	0.52	28
	12	E type switch seq.	NB	25	47.42	0.74	0.48	29
	13	res. types interact.	SVM-$C2^{10}$	10	45.36	0.69	0.42	30
	14	concept instances	SVM-$C2^{-2}$	3	44.33	0.71	0.45	31
	15	acc. obj. help topic	KNN-10	1	44.33	0.70	0.44	33
Top k Feat.	1	Top $k = 6$	NB	250	94.85	0.98	0.95	1
	2	Top $k = 5$	NB	150	92.78	0.97	0.94	2
	3	Top $k = 4$	NB	500	89.69	0.96	0.91	4
	4	Top $k = 3$	NB	300	89.69	0.96	0.90	5
	5	Top $k = 2$	NB	50	86.60	0.95	0.87	6
	6	Top $k = 20$	NB	7637	85.57	0.95	0.86	8
	7	Top $k = 15$	NB	7265	84.54	0.94	0.85	10
	8	Top $k = 9$	NB	750	82.47	0.93	0.84	12
	9	Top $k = 8$	NB	750	82.47	0.93	0.84	13
	10	Top $k = 7$	NB	750	82.47	0.93	0.84	14
	11	Top $k = 10$	NB	750	82.47	0.93	0.84	15

Table 7.6: Overview of the best results about the performance of detecting real workstation tasks by training on task executions from a laboratory setting for each feature category, each single feature as well as the Top k performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

Top k Features: The performance for different k was studied and obtained with the NB classifier at g=250 attributes with the *Top* $k = 6$ features the highest accuracy (a=94.85%), precision (p=0.98) and recall (r=0.95), among all studied features, feature categories and *Top* k combinations. This was an accuracy increase of 9.28%, a precision increase of 0.03 and a recall increase of 0.08 compared to the performance of the *window title feature*. The *Top k* feature combinations achieved accuracy values between 82.47% and 94.85% with the Naïve Bayes algorithm.

Comparison with existing approaches: The best performance of the already existing approaches was 88.66% accuracy with the Naïve Bayes algorithm with 50 features by the Dyonipos feature combinations $\mathcal{CW}$ and $\mathcal{ACW}$. This was 5.22% and 8.25% better than the *SWISH* and the *TaskPredictor 1* approach respectively. In comparison with the best UICO result, the *Top* $k = 6$ feature combination (l=NB, a=94.85%, g=250, p=0.98, r=0.95) outperformed the best Dyonipos approach by 6.16%, the *SWISH* approach (l=NB, a=80.41%, g=50, p=0.93, r=0.82) by 13.41% and the *TaskPredictor 1* approach (l=NB, a=80.41%, g=100, p=0.92, r=0.81) by 14.44%. All the feature combinations of the existing approaches had their best run with the Naïve Bayes learner like the *Top* $k = 6$ UICO feature combination. A detailed comparison of the feature and classifier performance evaluation of the existing approaches is given in Table 7.7.

Set	R_G	f	l	g	a	p	r
	1	$\mathcal{CW}$	NB	50	88.66	0.96	0.90
	2	$\mathcal{ACW}$	NB	50	88.66	0.96	0.90
	3	$\mathcal{W}$	NB	75	85.57	0.95	0.87
Dyonipos	4	$\mathcal{AW}$	NB	100	85.57	0.95	0.87
	5	$\mathcal{AC}$	NB	50	85.57	0.95	0.86
	8	$\mathcal{C}$	NB	500	69.07	0.87	0.72
	9	$\mathcal{A}$	KNN-35	3	36.08	0.62	0.35
SWISH	6		NB	50	81.44	0.93	0.82
TaskPredictor 1	7		NB	100	80.41	0.92	0.81

Table 7.7: Overview of the best results about the performance of detecting real workstation tasks by training on task executions from laboratory setting for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Concluding Remarks: The good performance of the classical *window title* feature was confirmed but significantly outperformed by a specific combination of the *Top* $k = 6$ UICO features. The positive influence of specific context features on task detection performance could be an indication that it is not necessary to sense “everything” about the user’s interactions with her computer desktop but only some relevant elements. This would have an impact on what kind of sensors have to be developed, i.e. which context features have to be sensed, to achieve a reasonable task detection performance. It would also impact the user's system performance because capturing less data normally leads to less CPU (central processing unit) requirements. Furthermore if we know which features are performing well for supervised machine learning algorithms in laboratory settings, it could provide a first indication on which features could be used in an unsupervised learning approach and in real world settings. However, this would require further experiments in laboratory and real world settings.

7.4.4 Research Question: Can task models of task instances from personal task executions be detected based on predefined standard task executions for training the classifier?

The goal of this evaluation was to answer the question *"Can task models of task instances from personal task executions be detected based on predefined standard task executions for training the classifier?"* This question studies the relation of a task instance to its task model and to what degree the *"specific goal"* of a task instance influences the detectability. An example of a specific goal is "Bill Adams is planning a journey to the CHI 2010 conference". Having multiple users executing a task with the same specific goal means that the task instances only vary in terms of how the user performs the task, i.e., her user interactions with applications and resources. The specific is the same. The answer to this research question is important because it will give an insight if it was sufficient to train on task instances with one specific goal in order to detect similar task instances with other specific goals. In order to answer this research question the data from the first laboratory task experiment described in Section 7.4 was used. The participants of the experiment were asked to pretend to be the artificial employee "Bill Adams" while performing predefined standard task executions. This data was used to train different classification algorithms. The subjects were also asked to execute the same tasks as themselves. The order in which the tasks were executed was randomized. It was also randomized if the subjects started performing the task on behalf of Bill Adams or themselves. The training instance construction, the preprocessing steps, the attribute selection and the classification algorithms were the same as described in Section 7.3.

Classes	**Standard Tasks**	**Personal Tasks**	**Sum**
Task 1	28	27	55
Task 2	23	22	45
Task 3	26	24	50
Task 4	27	24	51
Task 5	9	8	17
Dataset (Train/Test)	**113**	**105**	218

Table 7.8: This table shows the distribution of the training instances (standard tasks) and test instances (personal tasks) for the different task classes ranging from Task 1 to Task 5 recorded on the laboratory and on the personal workstations.

In Table 7.8 the distribution of the task instances in respect to the classes (*Task 1* to *Task 5*) as well as in respect to the standard and personal tasks is shown. The dataset contained 218 tasks from 14 users. Among these 218 tasks there were 113 *standard tasks* and 105 *personal tasks.* Standard tasks are tasks that have a specific goal, as described in Section 7.4.1. Personal tasks are tasks the subjects performed on behalf of themselves.

An overview of all results about the performance of detecting the tasks (Task 1-5) is given in Table 7.9. A training instance was built for each standard task and a test instance for each personal task independently. For the evaluation of the task classification the evaluation method train and test set (see Section 7.3.3) was used whereas the standard task instances constituted the training set and the personal task instances the test set.

Feature Categories: The best feature category was the *application category* that correctly identified 75.24% of the personal tasks (l=NB, g=200, p=0.92, r=0.77). The same accuracy was achieved by the combination of all 50 features of all categories (l=NB, a=75.24%, g=5000, p=0.92, r=0.76). The next place in the category ranking went to the *ontology structure category* with an accuracy of 61.90%, which was 13.34% worse than the best performing feature category. The best two accuracy values were achieved by the Naïve Bayes algorithm and third best result by the J48 decision tree learner on 250 attributes.

Single Features: The best performing single feature was the *acc. obj. name* feature which obtained an accuracy of 66,67% (l=KNN-35, g=25, p=0.88, r=0.66). The *window title* feature was a little bit worse with the Naïve Bayes classifier in terms of accuracy (a=65.71%), precision (p=0.88). Simply counting the number of UICO datatype relations (a=60.00%) was also quite efficient and only 6.67% worse than the best performing single feature. The range of attributes for the best runs of the single performing features was between 3 and 360 attributes except for the *used res. metadata* and *content of EB* features which had their best runs with 1000 and 750 attributes respectively.

Top k Features: The task detection performance for the different k best performing single feature combinations resulted between 69.52% and 77.14% accuracy. The *Top* $k = 2$ features obtained the best result with the KNN-35 classifier on only 25 attributes (p=0.93,r=0.75). This combination also achieved the highest accuracy in comparison to all evaluated single features and feature categories. The precision and recall values of the *Top k* features were close together between p=0.90 and p=0.92 and r=0.73 and r=0.75 respectively. The Naïve Bayes classifier was the best performing learning algorithm for all *Top k* feature combinations except for the one with the highest accuracy which was the *Top* $k = 2$. The number of attributes required to achieve the best results was quite high. The *Top k* with $k = 5, 7, 8, 9, 10, 15, 20$ had their best runs with attributes from a range between 2500 and 10000 attributes.

Comparison with existing approaches: The context features utilized by already existing approaches showed a similar performance in terms of accuracy. They all obtained an accuracy above 70%, more specifically the best Dyonipos feature combination $\mathcal{CW}$ had the highest accuracy among the existing approaches with 75.24% followed by the *SWISH* approach with 72.38% and

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	Application Cat.	NB	200	75.24	0.92	0.77	2
	2	All Categories	NB	5000	75.24	0.92	0.76	3
	3	Ontology Str. Cat.	J48	250	61.90	0.86	0.59	16
	4	Resource Cat.	J48	150	59.05	0.85	0.62	18
	5	Content Cat.	KNN-35	125	58.10	0.85	0.61	19
	6	Action Cat.	J48	300	56.19	0.83	0.54	24
	7	Switching Seq. Cat.	J48	2000	44.76	0.73	0.40	33
Single Feat.	1	acc. obj. name	KNN-35	25	66.67	0.88	0.66	14
	2	window title	NB	360	65.71	0.88	0.64	15
	3	datatype properties	KNN-35	224	60.00	0.85	0.61	17
	4	used res. metadata	NB	1000	58.10	0.85	0.59	20
	5	content of EB	NB	750	58.10	0.83	0.54	21
	6	content in focus	KNN-35	75	56.19	0.84	0.58	22
	7	acc. obj. value	J48	50	56.19	0.84	0.57	23
	8	acc. obj. role	KNN-35	35	50.48	0.79	0.47	25
	9	used res. content	KNN-35	50	49.52	0.79	0.50	26
	10	resource content	KNN-35	25	49.52	0.79	0.50	27
	11	acc. obj. role des.	KNN-35	61	48.57	0.77	0.46	28
	12	applications interact.	J48	50	46.67	0.76	0.45	29
	13	objecttype properties	KNN-35	25	45.71	0.77	0.46	30
	14	nr. of E/EB	NB	3	45.71	0.75	0.42	31
	15	res. types interact.	KNN-35	25	44.76	0.75	0.43	32
Top k Feat.	1	Top $k = 2$	KNN-35	25	77.14	0.93	0.75	1
	2	Top $k = 3$	NB	75	74.29	0.92	0.78	4
	3	Top $k = 8$	NB	2000	73.33	0.92	0.76	5
	4	Top $k = 7$	NB	2000	73.33	0.92	0.76	6
	5	Top $k = 10$	NB	2500	73.33	0.92	0.76	7
	6	Top $k = 20$	NB	10000	73.33	0.92	0.74	8
	7	Top $k = 15$	NB	7500	73.33	0.92	0.74	9
	8	Top $k = 5$	NB	1500	72.38	0.91	0.75	10
	9	Top $k = 9$	NB	2500	72.38	0.91	0.73	11
	10	Top $k = 6$	NB	750	69.52	0.90	0.73	12
	11	Top $k = 4$	NB	500	69.52	0.90	0.73	13

Table 7.9: Overview of the best results about the performance of detecting personal tasks by training on usage data from standard tasks for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

then the *TaskPredictor 1* approach with 70.48%. Whereas the *SWISH* and the *TaskPredictor 1* had their best run with the J48 learner, the best result was achieved by the Dyonipos approach with the Naïve Bayes algorithm. In comparison with the best UICO feature combination, the *Top* $k = 2$ and the combination of all 50 features outperformed the best Dyonipos feature combination, the *SWISH* approach and the *TaskPredictor 1* approach by 1.9%, 4.8% and 6.66% respectively. The *Top* $k = 2$ UICO feature combination only needed 25 attributes whereas the best Dyonipos approach and the *SWISH* approach required 1000 and 497 attributes respectively.

Set	R_G	f	l	g	a	p	r
Dyonipos	1	$\mathcal{CW}$	NB	1000	75.24	0.92	0.77
	2	$\mathcal{ACW}$	NB	1000	74.29	0.92	0.77
	3	$\mathcal{AW}$	NB	175	73.33	0.92	0.75
	4	$\mathcal{W}$	NB	175	72.38	0.91	0.74
	7	$\mathcal{AC}$	J48	2000	61.90	0.86	0.63
	8	$\mathcal{C}$	NB	1308	60.00	0.85	0.59
	9	$\mathcal{A}$	J48	21	40.00	0.71	0.39
SWISH	5		J48	497	72.38	0.91	0.71
TaskPredictor 1	6		J48	25	70.48	0.90	0.71

Table 7.10: Overview of the best results about the performance of detecting personal tasks by training on usage data from standard tasks for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Influence of the "Specific Goal": The task detection performance was also evaluated based on *standard task* and *personal tasks* separately. This was done in order to study the influence of the *specific goal* in the automatic task detection. The results of the evaluation for the standard tasks unveiled a task detection performance of 88.41% accuracy with 175 attributes with the J48 decision tree algorithm (p=0.96,r=0.86) with the *Top* $k = 5$ best performing single features combination. The best two single performing features were the *acc. obj. name* feature (l=J48, a=87.80%, g=25, p=0.96, r=0.88) and the *window title* feature (l=KNN-5, a=81.44, g=25, p=0.94, r=0.84). The best feature category was the *Application Cat.* with an accuracy of 87.73% (l=J48, a=87.73, g=100, p=0.96, r=0.84).

For the *personal tasks* an accuracy of 86.00% was achieved by the *Top* $k = 5$ feature combination (l=NB, g=300, p=0.95, r=0.86). The best three single performing features were the *acc. obj. name* feature (l=J48, a=76.00%, g=25, p=0.92, r=0.77), *window title* (l=NB, a=71.18%, g=100, p=0.90, r=0.74) feature and the *acc. obj. value* (l=KNN-5, a=68.45%, g=50, p=0.88, r=0.66) feature. The best feature category in terms of accuracy was the *application feature category* with an accuracy of 83.09% (l=KNN-5, a=83.09%, g=75, p=0.94, r=0.81).

The difference of the accuracy values of detecting standard and personal tasks, i.e., tasks with a single predefined specific goal and without one, was only 2.41% in terms of accuracy. The highest accuracy values were achieved with the J48 decision tree learner on a rather small number of attributes in both evaluations. For standard and personal tasks the highest accuracy values were reached with 175 and 75 attributes respectively.

Concluding Remarks: The UICO, the Dyonipos, *SWISH* and *TaskPredictor 1* approaches had a similar performance in terms of accuracy. However, the UICO approach outperformed all the other existing approaches. The accuracy of detecting the task model of personal task when training on standard tasks was about 77% which could be an indication that it was possible to automatically detect new personal task instances when training the classifier on a predefined set of standard tasks. This indication has to be investigated in further experimental as well as real world settings, in another domains with other users and other tasks. The results of comparing the task detection performance of standard and personal tasks, i.e., tasks with a single predefined specific goal and without one, suggested that the influence of the specific goal was limited. The task detection performance for standard tasks was only 2.41% better than for personal tasks in terms of accuracy.

7.4.5 Research Question: Is there a difference in automatically detecting tasks on a laboratory computer or on a personal workstation?

This sections investigated the influence of the environment, i.e., computer desktop, for automatic task detection and the discriminative features for tasks. For this the usage data observed during the task executions were split into tasks from the laboratory computer and into ones from the personal workstations. First, the task detection performance and the most significant features without *Task 5 "Organization of a project meeting"* was evaluated for both settings. Secondly, this *Task 5* was included again for studying the task detection performance and the most discriminative features for personal workstation tasks.

7.4.5.1 Laboratory Computer Workstation Tasks without Task 5

This section evaluates the task detection performance of detecting the task model of the task instances on the dataset resulted from the task executions on the laboratory workstation. The distribution of the task instances in respect to the task models is shown in Table 7.11. An overview of all results about the performance of detecting the laboratory tasks (*Task 1-4*) is given in Table 7.12. Stratified 10-fold cross-validation was applied on the training instances. A training instance was built for each task instance.

Set	Task 1	Task 2	Task 3	Task 4	Sum
Laboratory Workstation	30	26	26	24	**106**

Table 7.11: This table shows the distribution of training/test instances of the different task classes ranging from Task 1 to Task 4 for stratified 10-fold cross-validation. Training and test instances were constructed based on the usage data recorded on the laboratory workstation.

Feature Categories: The best feature category was the *application category* which correctly identified 83.91% of the tasks (l=J48, g=750, p=0.94, r=0.84) and hence also achieved the second best result among all tested combinations. About 2.5% behind in terms of accuracy was the combination of all 50 features of all categories (l=NB, a=81.36%, g=5000, p=0.92, r=0.82). The third place in the category ranking went to the *resource category* with an 59.73% accuracy (l=KNN-1, g=50, p=0.79, r=0.59) which was about 24% worse than the best performing feature category. The high precision of 0.94 of the *application category* outperformed the *resource category* with 0.15. The range of the recall values was between 0.84 for the best one and 0.59 for the third best one. The worst performing feature categories were the *content category* and *action category* which achieved about 50% accuracy with the J48 learner. The range of the number of attributes for training the classification algorithms spanned from 50 to 5000 attributes for the best runs of the classifiers. The highest accuracy among the feature categories was achieved with 750 attributes.

Single Features: The performance of each single feature was evaluated separately and showed that the *acc. obj. name* feature was with an accuracy of 83.36% the best performing single feature (l=KNN-1, g=50, p=0.93, r=0.84). The feature *acc. obj. value* obtained a about 8% lower accuracy with 74.55% (l=J48, g=50, p=0.90, r=0.76) on the same number of attributes (g=50). The *window title* feature reached the third place of the single performing features with the J48 learner on 293 attribtues with an accuracy of 72.36% (p=0.88, r=0.73). Simply counting the number of UICO datatype relations, which was done by the *datatype properties* feature resulted in an accuracy of 58.91% (l=J48, g=221, p=0.80, r=0.58) and hence was about 13% worse than *window title* feature. The accuracy values of the single performing features from rank 7 downwards were bellow 50%. Interesting to note here is that the range of the numbers of attributes for the best runs of the single performing features was between 50 and 291 which goes along with the findings of [Shen *et al.*, 2006] and [Lokaiczyk *et al.*, 2007].

Top k Features: The task detection performance for the different k best performing single feature combinations resulted between 80.64% and 83.91% accuracy. The *Top* k = 4 features obtained the best result with the Naïve Bayes classifier on 125 attributes (a=83.91%, p=0.94, r=0.85). This combination also achieved the highest accuracy in comparison to all evaluated single features and feature categories. The precision and recall values of the *Top k* features were close together between p=0.91 and p=0.94 and r=0.80 and r=0.85 respectively. The Naïve Bayes and the J48 classifier were the best performing learning algorithms for the *Top k* feature combinations. The range of the numbers of attributes for the best runs of the *Top k* performing features was between 25 and 500 except for the Top k = 20 combination which had its best run with 2500 attributes.

Comparison with existing approaches: The highest accuracy value of 75.18% among the existing approaches was achieved by the Dyoniops feature combination $\mathcal{ACW}$ with the Naïve Bayes learner on 250 attributes. The *SWISH* and the *TaskPredictor 1* approaches obtained 71.91% with the J48 learner and 70.82% with the KNN-10 learner respectively. In comparison with

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	Application Cat.	J48	750	83.91	0.94	0.84	2
	2	All Categories	J48	5000	81.36	0.92	0.82	10
	3	Resource Cat.	KNN-1	50	59.73	0.79	0.59	17
	4	Ontology Str. Cat.	J48	359	57.82	0.79	0.58	20
	5	Switching Seq. Cat.	J48	50	52.91	0.75	0.51	21
	6	Content Cat.	J48	1614	50.73	0.71	0.49	23
	7	Action Cat.	J48	1751	50.18	0.74	0.50	25
Single Feat.	1	acc. obj. name	KNN-1	50	83.36	0.93	0.84	4
	2	acc. obj. value	J48	50	74.55	0.90	0.76	15
	3	window title	J48	293	72.36	0.88	0.73	16
	4	datatype properties	J48	221	58.91	0.80	0.58	18
	5	used res. metadata	KNN-10	75	58.36	0.79	0.58	19
	6	content of EB	NB	175	52.27	0.74	0.50	22
	7	mean EB duration	SVM-$C2^{-2}$	1	50.27	0.74	0.51	24
	8	applications interact.	J48	5	49.45	0.71	0.48	26
	9	concept instances	KNN-10	50	49.09	0.73	0.50	27
	10	used resources	NB	10	48.18	0.72	0.47	28
	11	content in focus	J48	50	48.09	0.71	0.48	29
	12	acc. obj. role	SVM-$C2^{10}$	3	47.73	0.72	0.48	30
	13	acc. obj. role des.	J48	10	47.55	0.72	0.49	31
	14	res. types interact.	J48	25	47.09	0.73	0.49	32
	15	objecttype properties	J48	57	46.82	0.71	0.46	33
Top k Feat.	1	Top $k = 4$	NB	125	83.91	0.94	0.85	1
	2	Top $k = 5$	J48	125	83.55	0.93	0.84	3
	3	Top $k = 2$	J48	25	83.18	0.93	0.83	5
	4	Top $k = 20$	J48	2500	82.82	0.93	0.84	6
	5	Top $k = 3$	NB	125	82.18	0.92	0.82	7
	6	Top $k = 15$	J48	200	81.82	0.92	0.82	8
	7	Top $k = 9$	J48	500	81.36	0.92	0.82	9
	8	Top $k = 8$	J48	150	81.36	0.92	0.80	11
	9	Top $k = 6$	J48	125	81.09	0.93	0.82	12
	10	Top $k = 10$	J48	250	80.82	0.91	0.81	13
	11	Top $k = 7$	J48	250	80.64	0.92	0.80	14

Table 7.12: Overview of the best results about the performance of detecting laboratory computer tasks (Task 1-4) by stratified 10-fold cross-validation for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

the best UICO feature combination, the *Top* $k = 4$, all the existing approaches were outperformed. More specifically, the *Top* $k = 4$ topped the best Dyonipos feature combination, the *SWISH* and the *TaskPredictor 1* approaches by 8.18%, 12% and 13.09%. A detailed overview of the feature and classifier performances of the existing approaches is given in Table 7.13.

Set	R_G	f	l	g	a	p	r
Dyonipos	1	$\mathcal{ACW}$	NB	250	76.18	0.90	0.78
	2	$\mathcal{CW}$	NB	200	75.45	0.90	0.75
	4	$\mathcal{AW}$	NB	367	71.73	0.88	0.71
	5	$\mathcal{W}$	KNN-5	25	71.64	0.88	0.73
	7	$\mathcal{AC}$	KNN-1	25	54.64	0.77	0.54
	8	$\mathcal{C}$	NB	200	53.00	0.75	0.52
	9	$\mathcal{A}$	J48	16	40.64	0.65	0.39
SWISH	3		J48	413	71.91	0.87	0.71
TaskPredictor 1	6		KNN-10	75	70.82	0.87	0.71

Table 7.13: Overview of the best results about the performance of detecting laboratory computer tasks (Task 1-4) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Concluding Remarks: All the *Top k* combinations outperformed the feature categories and the single features except the *application feature category* (a=83.91%, R_G=2) and the *acc. obj. name* (a=83.36%, R_G=4) single feature. The highest accuracy was achieved by the *Top* $k = 4$ feature combination with the Naïve Bayes algorithm on 125 attributes with 83.91% accuracy. The UICO feature combination was between 8.18% and 13.09% better than the already existing approaches.

7.4.5.2 Personal Workstation Tasks without Task 5

This section describes the results achieved by training the learning algorithms on personal workstation tasks. For training, the tasks *Task 1* to *Task 4* were used. *Task 5* was not considered for this experiment to assure better comparability to the laboratory task experiment results in Section 7.4.5.1. The task distribution in respect to the task models is shown in Table 7.14 An overview of all results about the performance of detecting the workstation computer tasks (*Task 1-4*) is given in Table 7.15. Stratified 10-fold cross-validation was applied on the training instances. A training instance was built for each task instance of a personal workstation task.

Feature Categories: The combination of all 50 features achieved 91.29% accuracy (l=NB, g=500, p=0.97, r=0.92) which was the highest accuracy value of all studied feature categories. On the second rank was the *application category* with 88.56% accuracy which was only 2.73% worse (l=NB, g=300, p=0.96, r=0.89). The *content category* obtained an accuracy of 84.52%

Set	Task 1	Task 2	Task 3	Task 4	Sum
Dataset CV	25	19	25	28	**97**

Table 7.14: This table shows the distribution of the stratified 10-fold cross-validation training/test instances for the different task classes ranging from Task 1 to Task 4 recorded on the personal workstations.

(l=NB, g=50, p=0.93, r=0.85) which was 6.77% worse than the combination of all feature categories. The precision values of the top three categories were all above 0.93 which was rather high and differed not more than 0.04. The range of the number of attributes for the best classifier runs spanned from 50 to 500 attributes. The highest accuracy among the feature categories were achieved with 500 attributes.

Single Features: The *window title* feature performed best among the single features with an accuracy of 87.78% (l=J48, g=3, p=0.95, r=0.88). The second best accuracy was obtained the *acc. obj. name* feature with 86.67% (l=J48, g=75, p=0.95, r=0.86) which was about 1% worse than the best one. The feature *content in focus* obtained 3% lower accuracy than the best one with 83.78% (l=KNN-10, p=0.93, r=0.84) based on only 10 attributes. Close behind was the *content of EB* feature with 82.78% accuracy based on 300 attributes (l=KNN-35, p=0.92, r=0.82). The top 4 and the top 9 best single performing features were above 80% and 70% accuracy respectively. The range of the numbers of the selected attributes for the 10 best performing single features was between 3 and 750 whereas the highest accuracy and precision was achieved with 3 attributes by the *window title* feature.

Top k Features: The task detection performance for the different *Top k* best performing single feature combinations resulted between 87.44% and 91.78% accuracy. The best UICO *Top k* feature combinations were the *Top k* = 9 (p=0.97,r=0.92) and *Top k* = 15 (p=0.97,r=0.91) with an accuracy of 91.78% with the Naïve Bayes on 300 and 175 attributes respectively. These two feature combinations also achieved the highest accuracy values of all UICO feature combinations. The best 10 *Top k* feature combinations outperformed all the other single features and feature categories with the Naïve Bayes learner on a range of 3 to 1000 attributes. The only exception was the *Top k* = 5 combination which had its best run with 3000 attributes.

Comparison with existing approaches: The *TaskPredictor 1* approach achieved the highest accuracy with 88.78% (l=J48, g=250, p=0.96, r=0.89). Very close behind was the Dyonipos $\mathcal{ACW}$ feature combination with 88.44% accuracy (l=NB, g=200, p=0.96, r=0.88). With a 4.22% lower accuracy than the *TaskPredictor 1* was the *SWISH* approach (l=J48, a=84.56%, g=125, p=0.96, r=0.89). The number of attributes for the best runs of the existing approach were between 25 and 250 attributes. The best UICO feature combinations, the *Top k* = 9 and *Top k* = 15 outperformed all the existing approaches with an accuracy of 91.78% on 300 and 175 attributes respectively by 3% to 7.22%. A detailed comparison of the feature and classifier performance evaluation of the existing approaches is given in Table 7.16.

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	NB	500	91.29	0.97	0.92	4
	2	Application Cat.	NB	300	88.56	0.96	0.89	8
	3	Content Cat.	NB	50	84.22	0.93	0.85	16
	4	Ontology Str. Cat.	J48	359	70.00	0.86	0.70	19
	5	Resource Cat.	KNN-5	125	78.67	0.91	0.79	20
	6	Action Cat.	NB	100	66.22	0.85	0.67	26
	7	Switching Seq. Cat.	J48	250	60.11	0.80	0.60	29
Single Feat.	1	window title	J48	3	87.78	0.95	0.88	9
	2	acc. obj. name	J48	75	86.67	0.95	0.86	15
	3	content in focus	KNN-10	10	83.78	0.93	0.84	17
	4	content of EB	KNN-35	300	82.78	0.92	0.82	18
	5	resource content	KNN-1	250	76.67	0.90	0.76	21
	6	used res. content	KNN-5	175	75.44	0.88	0.74	22
	7	acc. obj. value	J48	750	73.89	0.89	0.74	23
	8	datatype properties	J48	200	73.33	0.87	0.72	24
	9	used res. metadata	SVM-$C2^{10}$	5	73.11	0.88	0.72	25
	10	acc. obj. role des.	KNN-10	10	65.78	0.84	0.63	27
	11	acc. obj. role	J48	37	64.78	0.83	0.64	28
	12	applications interact.	KNN-10	5	56.00	0.78	0.57	30
	13	res. types interact.	J48	25	56.89	0.79	0.56	31
	14	concept instances	KNN-5	5	55.78	0.77	0.52	32
	15	objecttype properties	J48	57	54.78	0.77	0.55	33
Top k Feat.	1	Top $k = 9$	NB	300	91.78	0.97	0.92	1
	2	Top $k = 15$	NB	175	91.78	0.97	0.91	2
	3	Top $k = 10$	NB	200	91.67	0.97	0.91	3
	4	Top $k = 20$	NB	300	90.78	0.97	0.91	5
	5	Top $k = 8$	NB	750	89.78	0.96	0.90	6
	6	Top $k = 7$	NB	1000	89.78	0.96	0.90	7
	7	Top $k = 3$	NB	1000	87.67	0.95	0.88	10
	8	Top $k = 5$	NB	3000	87.56	0.95	0.88	11
	9	Top $k = 4$	NB	200	87.56	0.95	0.88	12
	10	Top $k = 2$	J48	3	87.56	0.95	0.87	13
	11	Top $k = 6$	NB	200	87.44	0.95	0.87	14

Table 7.15: Overview of the best results about the performance of detecting personal workstation tasks (Tasks 1-4) by stratified 10-fold cross-validation for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

Set	R_G	f	l	g	a	p	r
	2	$\mathcal{ACW}$	NB	200	88.44	0.96	0.88
	3	$\mathcal{CW}$	NB	125	87.67	0.95	0.88
	4	$\mathcal{AW}$	SVM-$C2^{10}$	25	86.78	0.95	0.87
Dyonipos	5	$\mathcal{W}$	SVM-$C2^{0}$	25	86.44	0.94	0.87
	6	$\mathcal{AC}$	NB	25	85.78	0.94	0.86
	8	$\mathcal{C}$	NB	150	83.78	0.93	0.84
	9	$\mathcal{A}$	J48	19	50.11	0.74	0.50
SWISH	7		J48	125	84.56	0.94	0.85
TaskPredictor 1	1		J48	250	88.78	0.96	0.89

Table 7.16: Overview of the best results about the performance of detecting laboratory computer tasks (Task 1-4) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Concluding Remarks: This classification problem of detecting the task model (Task 1-4) of task instances observed on the personal workstations was solved with an accuracy of 91.78% with the Naïve Bayes classifier on this dataset. The best UICO feature combination outperformed the existing approaches by 3% to 7.22% in terms of accuracy.

7.4.5.3 Personal Workstation Tasks with Task 5

This section describes the results achieved by training the learning algorithms on the complete set of personal workstation task available (*Task 1-5*). This evaluation is different in comparison to Section 7.4.5.2 in which *Task 5* was excluded. Having five tasks resulted in having five classes for the classification task, i.e., a five class classification problem. The dataset is shown in Table 7.17. An overview of all results about the performance of detecting the workstation computer tasks (Task 1-5) is given in Table 7.18. The results were achieved by applying 10-fold cross-validation on the training instances. A training instance was built for each task instance of a personal workstation task.

Set	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Sum**
Dataset CV	25	19	25	28	15	**112**

Table 7.17: This table shows the distribution of the stratified 10-fold cross-validation training/test instances for the different task classes ranging from Task 1 to Task 5 recorded on the personal workstations.

Feature Categories: The combination of all 50 features achieved 91.06% accuracy (l=NB, g=250, p=0.97, r=0.92) which was the highest accuracy value of all studied feature categories.

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	NB	250	91.06	0.97	0.92	4
	2	Application Cat.	J48	75	85.53	0.95	0.86	14
	3	Content Cat.	NB	750	82.35	0.94	0.81	15
	4	Resource Cat.	KNN-1	175	76.14	0.92	0.75	19
	5	Action Cat.	J48	2062	60.00	0.85	0.62	24
	6	Ontology Str. Cat.	J48	300	69.77	0.89	0.70	25
	7	Switching Seq. Cat.	SVM-$C2^5$	500	46.67	0.74	0.41	35
Single Feat.	1	window title	J48	25	86.67	0.96	0.85	13
	2	content of EB	NB	75	82.12	0.94	0.79	16
	3	content in focus	J48	125	80.61	0.94	0.80	17
	4	acc. obj. name	J48	25	79.55	0.93	0.78	18
	5	used res. metadata	KNN-5	175	75.68	0.92	0.76	20
	6	resource content	KNN-35	75	73.18	0.90	0.72	21
	7	used res. content	KNN-10	100	72.50	0.90	0.71	22
	8	acc. obj. value	J48	1323	70.68	0.89	0.69	23
	9	datatype properties	J48	221	69.62	0.89	0.70	26
	10	acc. obj. role des.	J48	64	60.98	0.84	0.59	27
	11	acc. obj. role	J48	37	60.53	0.85	0.61	28
	12	concept instances	J48	81	55.45	0.82	0.56	29
	13	res. types interact.	J48	33	52.73	0.80	0.53	30
	14	applications interact.	KNN-5	59	51.89	0.79	0.48	31
	15	objecttype properties	J48	57	51.14	0.77	0.50	32
Top k Feat.	1	Top $k = 8$	NB	1000	91.21	0.97	0.91	1
	2	Top $k = 9$	NB	300	91.14	0.98	0.93	2
	3	Top $k = 20$	NB	500	91.06	0.97	0.92	3
	4	Top $k = 15$	NB	2000	90.91	0.97	0.92	5
	5	Top $k = 7$	NB	250	90.30	0.97	0.90	6
	6	Top $k = 10$	NB	3500	90.15	0.97	0.91	7
	7	Top $k = 6$	NB	500	90.08	0.97	0.92	8
	8	Top $k = 5$	NB	300	89.47	0.97	0.91	9
	9	Top $k = 4$	NB	2500	87.73	0.96	0.88	10
	10	Top $k = 2$	NB	150	87.50	0.96	0.89	11
	11	Top $k = 3$	NB	1500	86.74	0.96	0.86	12

Table 7.18: Overview of the best results about the performance of detecting personal workstation tasks (Tasks 1-5) by stratified 10-fold cross-validation for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

On the second rank was the *application category* with 88.56% accuracy which was almost 6% worse (l=J48, g=75, p=0.95, r=0.86). The *content category* obtained an accuracy of 82.35% (l=NB, g=750, p=0.94, r=0.81) which was almost 9% worse than the combination of all features from all categories. The precision values of the top four categories were all above 0.92 which was rather high and differed not more than 0.05. The range of the number of attributes of the best classifier runs spanned from 75 to 2062 attributes but the highest accuracy values were achieved between 75 and 750 attributes. The precision values were quite high (0.89 to 0.97) except for the *switching sequence category* which just achieved a precision of 0.74 and 46.67% accuracy.

Single Features: The best performing single feature was the *window title* feature with an accuracy of 86.67% (l=J48, g=25, p=0.96, r=0.85). On the second place was the *content of EB* feature which was approximately 4.5% worse in terms of accuracy than the best single performing feature (l=NB, g=75, p=0.94, r=0.79). The *content in focus* feature obtained an accuracy of 80.61% (l=J48, g=125, p=0.94, r=0.80) which led to the third rank. The accuracy values for the top four single performing features were close together and differed only approximately 6% raning from 79.55% to 86.67%. The range of the numbers of the selected attributes for the best top 20 best performing single features was between 10 and 221 except for the *acc. obj. value* and the *used resources* which performed best with 1323 and 582 attributes respectively. Based on only 25 attributes the *window title* feature achieved the highest accuracy of 86.67% among the single performing features.

Top k Features: The best task detection performance for different k best performing single feature combinations achieved the *Top* $k = 8$ best single features combination with 91.21% with the Naïve Bayes learner on 1000 attributes. This was also the highest accuracy among all UICO features and feature categories. The second best among the *Top k* was the one with $k = 9$ with the same learner but only with 300 attributes and an accuracy of 91.15%. This one was only 0.07% worse in terms of accuracy. The range of accuracy values resulted from the *Top k* feature combination was between 86.74% and 91.15% accuracy with Naïve Bayes learner. The number of attributes used in the best runs of the classifiers spanned from 150 to 3500 attributes.

Comparison with existing approaches: The Dyonipos feature combination $\mathcal{ACW}$ obtained the highest accuracy with 89.55% on 200 attributes with the Naïve Bayes classifier. This combination was 6.6% better than the *SWISH* and 2.05% better than the *TaskPredictor 1* approach in terms of accuracy. All the existing approaches were outperformed by the best seven *Top k* with $k = \{6, 7, 8, 9, 10, 15, 20\}$ UICO single feature combinations. The best Dyonipos approach, the *SWISH* and the *TaskPredictor 1* approach were worse by 1.66%, 8.26% and 3.71% accuracy. A detailed comparison of the feature and classifier performance evaluation of the existing approaches is given in Table 7.19.

Concluding Remarks: The task models of the task instances from personal workstations can be detected with an accuracy of 91.21% with the Naïve Bayes learner. The existing approaches were outperformed by 1.66% to 8.26% by the best UICO feature combination which was the *Top* $k = 9$.

Set	R_G	f	l	g	a	p	r
	1	$\mathcal{ACW}$	NB	200	89.55	0.97	0.90
	2	$\mathcal{CW}$	NB	200	88.41	0.97	0.89
	4	$\mathcal{W}$	NB	25	85.76	0.96	0.87
Dyonipos	5	$\mathcal{AW}$	NB	125	85.76	0.96	0.86
	7	$\mathcal{C}$	NB	150	82.35	0.94	0.80
	8	$\mathcal{AC}$	NB	500	82.27	0.94	0.82
	9	$\mathcal{A}$	J48	21	53.33	0.81	0.54
SWISH	6		J48	50	82.95	0.95	0.84
TaskPredictor 1	3		J48	250	87.50	0.96	0.87

Table 7.19: Overview of the best results about the performance of detecting personal workstation tasks (Tasks 1-5) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

7.4.6 Research Question: Can the type of task be automatically detected when distinguishing routine and knowledge-intensive tasks?

The goal of the following evaluations was to answer the question *"Can the type of task be automatically detected when distinguishing routine and knowledge-intensive tasks?"*. The dataset on which this question was investigated contained 218 tasks from 14 users. Among these 218 tasks there were 151 *routine tasks* (Task 1-3) and 67 *knowledge-intensive tasks* (Task 4 and 5). The distribution of the task instances in respect to the task type (routine or knowledge-intensive) is shown in Table 7.20.

Type	**Classes**	**Class Instances**	**Sum**
	Task 1	55	
Routine Tasks	Task 2	45	151
	Task 3	51	
Knowledge Intensive Tasks	Task 4	52	67
	Task 5	15	
Dataset CV			**218**

Table 7.20: This table shows the distribution of the stratified 10-fold cross-validation training/test instances for the different task classes routine tasks (Task 1,2,3) and knowledge-intensive tasks (Task 4,5) which were recorded on the laboratory computer as well as on the personal workstations from 14 users.

An overview of all results about the performance of detecting if a task instance is a routine task or a knowledge-intensive task by applying stratified 10-fold cross-validation is given in Table 7.21.

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	Application Cat.	SVM-$C2^{-5}$	3	94.07	0.92	0.92	3
	2	All Categories	J48	300	92.73	0.91	0.91	14
	3	Resource Cat.	SVM-$C2^{-2}$	10	91.77	0.90	0.90	18
	4	Action Cat.	NB	500	88.61	0.84	0.84	19
	5	Ontology Str. Cat.	J48	3	87.62	0.84	0.84	24
	6	Switching Seq. Cat.	NB	175	86.23	0.81	0.81	25
	7	Content Cat.	KNN-35	200	82.60	0.73	0.73	34
Single Feat.	1	window title	J48	25	93.64	0.91	0.91	6
	2	acc. obj. name	J48	100	93.61	0.92	0.92	7
	3	acc. obj. value	SVM-$C2^{-2}$	25	92.68	0.90	0.90	15
	4	used res. metadata	SVM-$C2^{5}$	10	91.80	0.90	0.90	17
	5	datatype properties	KNN-35	100	88.55	0.85	0.85	20
	6	applications interact.	J48	25	88.55	0.85	0.85	21
	7	acc. obj. role des.	J48	5	88.05	0.83	0.83	22
	8	concept instances	SVM-$C2^{1}$	5	88.03	0.86	0.86	23
	9	res. types interact.	NB	36	85.84	0.80	0.80	26
	10	used resources	NB	300	85.78	0.80	0.80	27
	11	objecttype properties	J48	57	85.76	0.82	0.82	28
	12	acc. obj. role	SVM-$C2^{0}$	3	85.30	0.82	0.82	29
	13	EB res. interact.	SVM-$C2^{2}$	25	84.85	0.79	0.79	30
	14	used res. interact.	KNN-5	125	83.96	0.77	0.77	31
	15	res. interact.	KNN-5	50	83.55	0.75	0.75	32
Top k Feat.	1	Top $k = 3$	NB	500	94.94	0.94	0.94	1
	2	Top $k = 4$	SVM-$C2^{0}$	10	94.55	0.93	0.93	2
	3	Top $k = 2$	NB	100	94.05	0.93	0.93	4
	4	Top $k = 7$	J48	2500	93.64	0.92	0.92	5
	5	Top $k = 10$	J48	300	93.59	0.92	0.92	8
	6	Top $k = 20$	J48	150	93.16	0.92	0.92	9
	7	Top $k = 5$	NB	125	93.14	0.93	0.93	10
	8	Top $k = 9$	J48	175	93.14	0.92	0.92	11
	9	Top $k = 15$	J48	3500	93.07	0.92	0.92	12
	10	Top $k = 6$	J48	750	92.73	0.92	0.92	13
	11	Top $k = 8$	J48	3500	92.66	0.91	0.91	16

Table 7.21: Overview of the best results about the performance of detecting routine (Task 1-3) and knowledge-intensive tasks (Task 4 and 5) for each feature category, for all feature categories combined, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

Feature Categories: The *application category* performed best with 94.07% accuracy on only 3 attributes with the linear SVM algorithm (p=0.92, r=0.92) which resulted in the third rank of the global ranking ($R_G = 3$). Only 1.34% less accurate was the combination of all 50 features with the J48 learner on 300 attributes (a=92.73%, p=0.91, r=0.91). The *resource category* achieved with 91.77% accuracy with the linear SVM algorithm on 10 attributes the third rank (a=92.73%, p=0.91, r=0.92). The number of attributes for the best runs of the classifiers for the feature categories ranged from 3 to 500 attributes.

Single Features: The best performing single feature was the *window title* feature with an accuracy of 93.64% with the J48 algorithm. The performance in terms of accuracy was only 0.43% less than the best performing feature category, the *application category.* The *acc. obj. name* feature obtained 93.61% accuracy with the same algorithm as the *window title* feature but required additional 75 attributes. With 92.68% accuracy the *acc. obj. value* feature achieved 92.68% accuracy with the linear SVM learner on 25 attributes. The accuracy of the best 15 single performing features ranged from 83.55% to 93.64% whereas the top 4 ones were above 91%.

Top k Features: The *Top k* feature combinations resulted in accuracy values between 92.66% to 94.94%. The best among the *Top k* feature combinations was the *Top* k = 3 with an accuracy of 94.94% with the Naïve Bayes learner on 500 attributes (p=0.94, r=0.94) which outperformed all the other UICO single features and feature categories. The second and third best ones were the *Top* k = 4 and *Top* k = 2 which obtained 94.55% (l=SVM, g=10, p=0.93, r=0.93) and 94.05% (l=NB, g=100, p=0.93, r=0.93) accuracy. The number of attributes for the best runs was between 10 and 3500 attributes.

Set	R_G	f	l	g	a	p	r
	1	$\mathcal{AW}$	SVM-$C2^{-5}$	125	94.55	0.93	0.93
	2	$\mathcal{W}$	SVM-$C2^{-5}$	100	94.50	0.93	0.93
	3	$\mathcal{CW}$	SVM-$C2^{-5}$	300	94.05	0.92	0.92
Dyonipos	6	$\mathcal{ACW}$	SVM-$C2^{8}$	300	93.61	0.91	0.91
	7	$\mathcal{AC}$	NB	5	82.51	0.72	0.72
	8	$\mathcal{C}$	NB	125	79.85	0.68	0.68
	9	$\mathcal{A}$	SVM-$C2^{8}$	24	78.98	0.67	0.67
	5	*SWISH*	NB	150	94.03	0.92	0.92
	4	*TaskPredictor 1*	NB	150	94.03	0.93	0.93

Table 7.22: Overview of the best results about the performance of detecting routine (Task 1-3) and knowledge-intensive tasks (Task 4 and 5) for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Comparison with existing approaches: The best Dyonipos feature combination $\mathcal{AW}$, the *SWISH* and the *TaskPredictor 1* approach performed almost equally well with an accuracy of 94.55% (l=SVM, g=125,p=0.93,r=0.93), 94.03% (l=NB, g=150,p=0.92,r=0.92) and 94.03% (l=NB, g=150,p=0.93,r=0.93) respectively. However, the best UICO feature combination, the *Top* $k = 3$ with an accuracy of 94.94%, outperformed all the existing approaches. The difference in terms of accuracy was very low ranging from 0.39% to 0.91%. The number of attributes of the best runs of the classifiers for the existing approaches were between 5 and 300 attributes.

Concluding Remarks: The classification task for identifying routine (*Task 1-3*) and knowledge-intensive tasks (*Task 4* and *Task 5*) based on this dataset was solved with an accuracy of 94.94% by the UICO feature combination *Top* $k = 3$. The performance of the existing approaches were very close to the accuracy of the best UICO feature combination.

7.4.7 Research Question: Can the task model of a task instance be automatically detected based on task instances from only one expert user?

The goal of the following evaluation was to answer the question *"Can the task model of a task instance be automatically detected based on task instances from only one expert user?"*. The dataset on which this question was investigated contained 271 tasks from 14 users whereas 68 task instances came from the expert user and 203 task instances from 13 other users. These 271 tasks were almost equally distributed among the five task models (*Task 1-5*) for the expert user as well as the user group as visualized in Table 7.23.

Class	Single Expert	User Group	Sum
Task 1	15	51	66
Task 2	15	41	56
Task 3	18	47	65
Task 4	11	48	59
Task 5	9	16	25
Dataset (Train/Test/Sum)	**68**	**203**	271

Table 7.23: This table shows the distribution of the training instances (single expert user) and the test instances (multiple users) for the task models (Task 1-5) recorded on the laboratory computer and on the personal workstations.

An overview of all results about the task detection performance of detecting is given in Table 7.24. These results were achieved by training on the task instance of the single expert user and testing on the task instances of the user group. For each task instance a training instance was built.

Feature Categories: The best feature category was the *application category* with an accuracy of 57.64% (l=NB, g=1000, p=0.85, r=0.60). The performance of this feature category obtained the 11th rank in the overall ranking ($R_G = 11$). Only 0.50% behind in terms of accuracy was the combination of all 50 features which correctly identified 57.14% of the classes with the maximum

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	Application Cat.	NB	1000	57.64	0.85	0.60	11
	2	All Categories	J48	9348	57.14	0.81	0.50	12
	3	Resource Cat.	NB	500	54.68	0.81	0.52	15
	4	Content Cat.	KNN-35	100	52.71	0.80	0.48	18
	5	Ontology Str. Cat.	KNN-35	300	44.83	0.74	0.41	25
	6	Switching Seq. Cat.	J48	50	41.87	0.74	0.42	27
	7	Action Cat.	KNN-35	75	39.90	0.70	0.36	32
Single Feat.	1	window title	NB	75	68.97	0.88	0.63	2
	2	acc. obj. value	NB	75	61.58	0.86	0.62	4
	3	used res. metadata	NB	3000	54.19	0.82	0.54	17
	4	datatype properties	KNN-35	75	50.74	0.78	0.46	19
	5	content in focus	J48	150	49.26	0.77	0.45	20
	6	acc. obj. name	NB	270	48.28	0.79	0.50	21
	7	content of EB	NB	125	47.78	0.77	0.46	22
	8	objecttype properties	KNN-35	25	45.81	0.75	0.42	23
	9	applications interact.	J48	25	44.83	0.75	0.44	24
	10	acc. obj. role	KNN-10	3	42.36	0.71	0.37	26
	11	res. types interact.	KNN-35	10	41.38	0.74	0.44	28
	12	EB duration	J48	5	41.38	0.71	0.37	29
	13	app. switch seq.	NB	25	40.39	0.73	0.41	30
	14	E type switch seq.	J48	19	39.90	0.70	0.37	31
	15	used res. content	J48	50	37.93	0.67	0.33	33
Top k Feat.	1	Top $k = 2$	NB	100	72.91	0.91	0.71	1
	2	Top $k = 3$	NB	500	65.52	0.88	0.65	3
	3	Top $k = 10$	KNN-35	300	60.59	0.84	0.55	5
	4	Top $k = 9$	KNN-35	300	59.61	0.84	0.55	6
	5	Top $k = 7$	KNN-35	300	59.61	0.84	0.55	7
	6	Top $k = 8$	KNN-35	300	59.11	0.83	0.54	8
	7	Top $k = 6$	J48	300	59.11	0.83	0.52	9
	8	Top $k = 5$	J48	300	59.11	0.83	0.52	10
	9	Top $k = 20$	KNN-35	750	55.17	0.82	0.53	13
	10	Top $k = 4$	NB	500	55.17	0.82	0.52	14
	11	Top $k = 15$	KNN-35	1500	54.19	0.82	0.54	16

Table 7.24: Overview of the best results about the performance of detecting user tasks (Tasks 1-5) by training on expert tasks for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

of available attributes (l=J48, g=9348, p=0.81, r=0.50). The third place in the category ranking went to the *resource category* with an accuracy of 54.68% (l=NB, g=500, p=0.81, r=0.52), which was 2.96% worse than the best performing feature category. The accuracy values of the feature categories ranged from 39.90% (*action category*) to 57.64% (*application category*). The range of the number of attributes for training the classification algorithms spanned from 50 to 9348 attributes. The highest accuracy among the feature categories were achieved with 1000 attributes.

Single Features: The performance of each single feature was evaluated separately and showed that the *window title* feature was with an accuracy of 68.97% and with only 75 attributes (l=NB, p=0.88, r=0.63) the best performing single feature as well as the second best one in the global ranking (R_G). The *acc. obj. value* feature reached with the same number of attributes and the same learner but with an 7.39% less accuracy the second highest accuracy value of the single performing features (l=NB, g=75, p=0.86, r=0.62). The *used res. metadata* feature obtained an accuracy of 54.19% (l=NB, g=3000, p=0.82, r=0.54) and hence achieved the third place of the single performing features with a 14.78% less accuracy than the best one.

The accuracy values of the best 15 single performing features spanned from 37.93% to 68.97%. The range of the numbers of attributes for the best classifier runs of the best 15 single performing features was between 3 and 270 attributes except for the *used res. metadata* which had its best run with 3000 attributes. The best two performing single features outperformed all the feature categories in terms of accuracy. The *window title* feature was 11.33% more accurate in detecting the classes than the best feature category.

Top k Features: The classification performance for the combination of the *Top k* single performing features ranged from 54.19% to 72.91% accuracy. The *Top* k = 2 features obtained the best result with the NB classifier on 100 attributes (a=72.91%, p=0.91, r=0.71). This combination also achieved the highest accuracy in comparison to all evaluated single features and feature categories. The second best *Top k* feature combination was the *Top* k = 3 with a 7.39% less accuracy value (l=NB, a=65.52%, g=500, p=0.88, r=0.65) than the best one. With an accuracy of 60.59% the *Top* k = 10 feature combination obtained the third place in this category. This accuracy value was 12.32% lower then the best *Top k* feature combination.

The range of the numbers of attributes for the best runs of the *Top k* performing feature combinations were between 100 and 1500 attributes. The best eight *Top k* feature combinations outperformed all the feature categories in terms of accuracy.

Comparison with existing approaches: A comparison with existing approaches showed that the best overall tested combination of the UICO features, the *Top* k = 2 with 72.91% accuracy, achieved a higher accuracy than all the existing approaches. The *SWISH* (l=NB, a=65.52%, g=75, p=0.88, r=0.64), the *TaskPredictor 1* (l=NB, a=68.97%, g=100, p=0.89, r=0.69) and the best Dyonipos feature combination $\mathcal{ACW}$ (l=NB, a=69.46%, g=250, p=0.89, r=0.64) were outperformed by 7.39%, 3.94% and 3.45% accuracy respectively. The existing approaches' accuracy values ranged from 32.51% to 69.46% whereas the best Dyonipos, *SWISH* and *TaskPredictor 1* approach only differed about 3%. The number of attributes of the best classifier runs of the algorithms were between 12 and 250 attributes. A detailed comparison of

Set	R_G	f	l	g	a	p	r
Dyonipos	1	$\mathcal{ACW}$	NB	250	69.46	0.89	0.64
	3	$\mathcal{CW}$	NB	250	67.98	0.88	0.61
	5	$\mathcal{W}$	SVM-$C = 2^{10}$	75	65.52	0.86	0.56
	6	$\mathcal{AW}$	NB	50	65.02	0.88	0.67
	7	$\mathcal{AC}$	NB	125	57.64	0.82	0.52
	8	$\mathcal{C}$	NB	100	48.28	0.77	0.45
	9	$\mathcal{A}$	KNN-35	12	32.51	0.66	0.34
SWISH	4		NB	75	65.52	0.88	0.64
TaskPredictor 1	2		NB	100	68.97	0.89	0.69

Table 7.25: Overview of the best results about the performance of detecting user tasks (Tasks 1-5) by training on expert tasks for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

the feature and classifier performance evaluation of the existing approaches is given in Table 7.64.

Extended Evaluation: In order to explore the capabilities of the UICO features beyond the chosen evaluation methodology, all feature combinations consisting of 2-6 features of the 10 best performing single features was performed. The best single features can be observed in Table 7.28. This extended evaluation resulted in $\sum_{k=2}^{6} \frac{10!}{k!*(10-k!)} = 837$ feature combinations. The results show that an accuracy of 75.37% was reached with multiple feature combinations as shown in Table 7.26. The accuracy of the standard evaluation methodology was only increased by 2.34%.

FC	*L*	*l*	*g*	*a*	*p*	*r*
f1+f2+f7	3	NB	500	75.37	0.92	0.73
f1+f2+f5	3	NB	500	74.38	0.92	0.73
f1+f2+f5+f7+f9	5	NB	500	74.38	0.92	0.73
f1+f2+f5+f7	4	NB	500	74.38	0.91	0.72
f1+f2	2	NB	100	72.91	0.91	0.71
f1+f2+f5+f9	4	NB	500	72.41	0.91	0.72
f2+f7	2	NB	200	72.41	0.91	0.70

Table 7.26: Overview of the best results about the performance of detecting users' tasks (Tasks 1-5) by training on tasks from one expert via the evaluation of all feature combinations (FC) consisting of 2-6 features of the best 10 performing single features ($f1 - f10$). The number of features involved in the combination (L), learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Concluding Remarks: The evaluations of the task classification performance of detecting user tasks by training on expert tasks when focusing on five task models (*Tasks 1-5*) showed that an accuracy of about 73% can be reached when utilizing the UICO's *Top* $k = 2$ best single performing feature combination on this dataset. This feature combination was specific to the UICO approach and outperformed the feature and feature combinations of the existing approaches. The extended evaluation methodology showed that the accuracy was only increased by 2.34% to 75.37% in comparison with the standard evaluation methodology. This can be seen as an indication that although calculating multiple combinations of the best single performing this only lead to a small accuracy increase. In case of this research question the minor accuracy increase strengthen the view that one expert is not enough for classifier training in order to reach a high accuracy.

7.4.8 Research Question: Can the task model of a task instance of a single user be automatically detected based on task instances from multiple expert users?

The goal of this evaluation was to answer the question *"Can the task model of a task instance of a single user be automatically detected based on task instances from multiple expert users?"*. The datasets on which this question was investigated contained 271 tasks from 14 users whereas 203 task instances came from the expert user group and 68 task instances from one single user. These 271 tasks were almost equally distributed among the five task models (*Task 1-5*) for the expert user group as well as for the single user as visualized in in Table 7.27. This evaluation was exactly the opposite to the one described in Section 7.4.7 in the sense that the tasks used for training and testing were swapped.

An overview of all results about the task detection performance is given in Table 7.28. These results were achieved by training on the task instances of the expert group and testing on the task instances of the single user. For each task instance a training instance was built.

Class	Expert User Group	Single User	Sum
Task 1	51	15	66
Task 2	41	15	56
Task 3	47	18	65
Task 4	48	11	59
Task 5	16	9	25
Dataset (Train/Test/Sum)	**203**	**68**	271

Table 7.27: This table shows the distribution of the training instances (expert user group) and the test instances (one single user) for the task models (Task 1-5) recorded on the laboratory computer and on the personal workstations.

Feature Categories: The feature category which achieved the highest accuracy values was the combination of all 50 features of all feature categories with the J48 algorithm and 92.65%

accuracy (g=250, p=0.98, r=0.91). This feature combination resulted in the global rank $R_G = 6$. A 2.94% less accuracy was obtained by the *application category* with 89.71% accuracy and the NB algorithm on 500 attributes (p=0.97, r=0.90). The *ontology structure category* achieved 85.29% accuracy (l=KNN-35, g=75, p=0.96, r=0.85) and hence rank 3 in the feature category ranking.

The accuracy values resulting based on the feature categories were between 54.41% and 92.65%. In comparison to the performance of the single features and the best *Top k* feature combinations the best feature category the combination of all features of all categories, had the same accuracy values. The range of the number of attributes for the best classifier runs was between 75 and 3206 attributes whereas the best feature category used 250 attributes.

Single Features: The best performing single feature was the *window title* feature with an accuracy of 92.65% (l=NB, g=75, p=0.98, r=0.92). Only 1.47% less accurate in detecting the classes was the *datatype properties* feature with 57 attributes (l=NB, a=91.18%, p=0.98, r=0.91). The third rank of the best performing single features went to the *concept instances* feature which achieved an accuracy of 83.82% with 150 attributes (l=KNN-35, p=0.95, r=0.83).

The range of the numbers of attributes for the best runs of the best 15 single performing features was between 3 and 1500 attributes. The accuracy values for the best 15 single features ranged from 67.65% to 92.65%. In comparison with the *Top k* feature combinations and feature categories only the best single feature, the *window title* feature, performed equally well.

Top *k* Features: The *Top k* best single feature combinations achieved accuracy values ranging from 88.24% to 92.65% and hence only differed among each other by 4.41%. The highest accuracy values resulted from the *Top k* with $k = \{3, 6, 7, 20\}$ with the Naïve Bayes and the J48 learner with 50 to 250 attributes. This high accuracy of 92.65% was also obtained by the best single performing feature, the *window title*, and the best feature category (*All Categories*).

The range of the number of attributes for the best runs of the algorithms was between 50 and 250 attributes except for the *Top* $k = 4$ which had its best run with 1000 attributes and the KNN-35 algorithm.

Comparison with existing approaches: The evaluation of the existing approaches showed that the best two Dyonipos approaches, $\mathcal{CW}$ and $\mathcal{ACW}$ as well as the *TaskPredictor 1* approach achieved 97.06% accuracy with the Naïve Bayes algorithm. The *SWISH* approach achieved 89.71% accuracy also with the Naïve Bayes learner. The Dyonipos $\mathcal{CW}$ and $\mathcal{ACW}$ as well as the *TaskPredictor 1* approach outperformed the best UICO feature combination by 4.41% accuracy.

In all the best performing existing and UICO approaches the *window title* was one of the involved features. In the *Top k* best performing single feature construction strategy only the first k were considered in the set of features. This was the reason why there was no combination of the two features *window title* and *content of EB* or *window title* and *content in focus*.

A detailed overview of the feature and classifier performances of the existing approaches is given in Table 7.29.

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	J48	250	92.65	0.98	0.91	6
	2	Application Cat.	NB	500	89.71	0.97	0.90	10
	3	Ontology Str. Cat.	KNN-35	75	85.29	0.96	0.85	16
	4	Content Cat.	KNN-35	75	82.35	0.95	0.79	22
	5	Action Cat.	J48	300	76.47	0.92	0.75	27
	6	Resource Cat.	NB	3500	72.06	0.91	0.71	29
	7	Switching Seq. Cat.	J48	3206	54.41	0.83	0.58	36
Single Feat.	1	window title	NB	75	92.65	0.98	0.92	2
	2	objecttype properties	NB	57	91.18	0.98	0.91	8
	3	datatype properties	KNN-35	150	83.82	0.95	0.83	17
	4	content in focus	KNN-35	125	82.35	0.95	0.80	18
	5	used res. content	NB	1000	82.35	0.95	0.79	19
	6	content of EB	SVM-$C = 2^{10}$	1000	82.35	0.95	0.79	20
	7	resource content	NB	1500	82.35	0.95	0.79	21
	8	acc. obj. name	SVM-$C = 2^{10}$	10	80.88	0.94	0.81	23
	9	user input	NB	500	80.88	0.94	0.81	24
	10	acc. obj. value	KNN-35	25	77.94	0.93	0.78	25
	11	acc. obj. role des.	J48	3	77.94	0.93	0.71	26
	12	EB duration	SVM-$C = 2^{-1}$	3	73.53	0.91	0.68	28
	13	acc. obj. role	NB	3	70.59	0.90	0.63	30
	14	res. types interact.	J48	25	69.12	0.90	0.70	31
	15	concept instances	KNN-35	50	67.65	0.89	0.70	32
Top k Feat.	1	Top $k = 3$	NB	150	92.65	0.98	0.92	1
	2	Top $k = 7$	J48	50	92.65	0.98	0.91	3
	3	Top $k = 6$	J48	50	92.65	0.98	0.91	4
	4	Top $k = 20$	J48	250	92.65	0.98	0.91	5
	5	Top $k = 2$	NB	100	91.18	0.98	0.91	7
	6	Top $k = 15$	NB	250	91.18	0.98	0.89	9
	7	Top $k = 9$	NB	150	89.71	0.97	0.87	11
	8	Top $k = 8$	NB	150	89.71	0.97	0.87	12
	9	Top $k = 10$	NB	200	89.71	0.97	0.87	13
	10	Top $k = 5$	KNN-35	200	88.24	0.97	0.86	14
	11	Top $k = 4$	KNN-35	1000	88.24	0.97	0.86	15

Table 7.28: Overview of the best results about the performance of detecting a single user's tasks (Tasks 1-5) by training on tasks from multiple experts for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

Set	R_G	f	l	g	a	p	r
	2	$\mathcal{CW}$	NB	750	97.06	0.99	0.97
	3	$\mathcal{ACW}$	NB	1000	97.06	0.99	0.97
	4	$\mathcal{W}$	NB	25	92.65	0.98	0.93
Dyonipos	6	$\mathcal{AW}$	J48	10	89.71	0.97	0.88
	7	$\mathcal{AC}$	J48	500	86.76	0.96	0.85
	8	$\mathcal{C}$	NB	25	82.35	0.95	0.80
	9	$\mathcal{A}$	J48	22	57.35	0.84	0.59
SWISH	5		NB	75	89.71	0.97	0.90
TaskPredictor 1	1		NB	200	97.06	0.99	0.97

Table 7.29: Overview of the best results about the performance of detecting a single user's tasks (Tasks 1-5) by training on tasks from multiple experts for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

FC	L	l	g	a	p	r
f2+f6+f8	3	SVM-$C = 2^{10}$	25	98.53	1.00	0.98
f2+f6+f8+f9	4	SVM-$C = 2^{10}$	25	98.53	1.00	0.98
f2+f3+f4+f6+f8	5	J48	100	98.53	1.00	0.98
f2+f4+f5+f7+f8	5	J48	75	98.53	1.00	0.98
f2+f3+f4+f5+f7+f8	6	J48	100	98.53	1.00	0.98
f2+f3+f4+f6+f8+f9	6	J48	100	98.53	1.00	0.98
f2+f4+f5+f7+f8+f9	6	J48	75	98.53	1.00	0.98

Table 7.30: Overview of the best results about the performance of detecting single user's tasks (Tasks 1-5) by training on tasks from multiple experts via the evaluation of all feature combinations (FC) consisting of 2-6 features of the best 10 performing single features ($f1 - f10$). The number of features involved in the combination (L), learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Extended Evaluation: The UICO approach was outperformed by some combinations of features of the existing approaches because of the chosen evaluation methodology. The evaluation methodology for measuring the performance of the best k single performing features did not take into account all possible feature combinations such that the best performing feature combinations were not evaluated. Since it is not feasible to calculate all possible combinations with computers available today some combinations of features were not evaluated. For a more detailed discussion about finding the best feature combination it is referred to Section 7.7. However, in order to explore the capabilities of the UICO features beyond the chosen evaluation methodology, all feature combinations consisting of 2-6 features of the 10 best performing single features was performed. The best single features can be observed in Table 7.28. This extended evaluation resulted in $\sum_{k=2}^{6} \frac{10!}{k!*(10-k!)} = 837$ feature combinations. The results show that an accuracy of 98.53% was reached with multiple feature combinations as shown in Table 7.30. The accuracy of the standard evaluation methodology was increased by 5.88% and hence all the existing approaches were outperformed by UICO feature combinations. Interesting to note here is that the *window title* feature was not part of the best performing combinations.

Concluding Remarks: The evaluations of the task classification performance of detecting a single user's tasks (Tasks 1-5) by training on tasks from multiple experts (*Tasks 1-5*) showed that an accuracy of 97.06% was reached by the Dyonipos $\mathcal{CW}$ and $\mathcal{ACW}$ as well as the *TaskPredictor 1* approach on this dataset. A 4.41% lower accuracy was reached by utilizing the best UICO feature combinations based on the standard evaluation methodology. Through an extended evaluation of all feature combinations consisting of 2-6 features of the 10 best performing single features an accuracy of 98.53% was reached. Hence the UICO features outperformed all the existing features and feature combinations of the existing approaches.

7.4.9 Finding the Best Features/Feature Categories

Feature Dominance Matrix: Table 7.31 displays the dominance matrix for the features and feature combinations as described in Section 7.3.5.1. It showed that the *application category*, the *window title* feature and the combination of all features (*All Categories*) outperformed the other features and feature combinations most often. As expected the *window title* feature performed really well. Surprising was that the four accessibility object features *acc. obj. name*, *acc. obj. value*, *acc. obj. role des.*, *acc. obj. role* turned out to have such a good performance. The *acc. obj. name* feature was only 26 times less dominant to other features than the well-known *window title* feature. A good performance as well showed the *datatype properties* feature and the *ontology structure* category which achieved the 8th and the 10th rank in the feature dominance matrix respectively.

Paired T-Tests: The statistical significance tests $T-Test^{f}$ and $T-Test_{rt}^{f}$ as described in Section 7.3.5.1 are summarized in Table 7.32 and Table 7.33 respectively. The $T-Test^{f}$ showed that the *application category* and the combination of all features (*All Categories*) performed statistically significantly better than all other features and feature combinations on a $p < 0.05$ significance level. By comparing the results for the $T-Test^{f}$ and $T-Test_{rt}^{f}$ one can observe

	Application Cat.	window title	All Categories	acc. obj. name	acc. obj. value	Resource Cat.	Content Cat.	datatype properties	used res. metadata	Ontology Str. Cat.	content of EB	content in focus	Action Cat.	acc. obj. role des.	resource content	acc. obj. role	Ψ
Application Cat.	-	6	5	9	8	9	9	9	9	9	9	9	9	9	9	9	495
window title	3	-	4	6	8	9	9	9	9	9	9	9	9	9	9	9	489
All Categories	4	5	-	7	8	8	7	8	8	9	9	8	9	9	9	9	486
acc. obj. name	0	3	2	-	8	8	5	7	8	8	7	5	9	9	8	9	463
acc. obj. value	1	1	1	1	-	6	4	7	7	7	4	4	9	9	6	9	440
Resource Cat.	0	0	1	1	3	-	5	6	8	7	5	5	8	8	8	9	439
Content Cat.	0	0	2	4	5	4	-	5	6	5	8	8	8	8	9	8	437
datatype properties	0	0	1	2	2	3	4	-	5	5	5	5	8	9	7	9	430
used res. metadata	0	0	1	1	2	1	3	4	-	6	4	4	8	8	7	8	419
Ontology Str. Cat.	0	0	0	1	2	2	4	4	3	-	4	4	8	8	7	9	418
content of EB	0	0	0	2	5	4	1	4	5	5	-	4	8	8	9	8	416
content in focus	0	0	1	4	5	4	1	4	5	5	5	-	7	8	9	8	415
Action Cat.	0	0	0	0	0	1	1	1	1	1	1	2	-	7	5	7	380
acc. obj. role des.	0	0	0	0	0	1	1	0	1	1	1	1	2	-	3	5	362
resource content	0	0	0	1	3	1	0	2	2	2	0	0	4	6	-	5	361
acc. obj. role	0	0	0	0	0	0	1	0	1	0	1	1	2	4	4	-	361

Table 7.31: Feature dominance matrix for the top 15 features and feature combinations. (The last two features performed equally well, hence the matrix has 16 rows and columns.)

that (i) the first 13 features and feature combinations appear in both tables and (ii) that the order in which they were ranked (Ψ) was only slightly different. A comparison of the t-test tables with the feature dominance matrix in Table 7.31 highlights that the best 15 ranked features appear in all three tables. These results strength the argument that these were the best performing features for this dataset, i.e., the ones with the highest discriminative power.

	All Categories	Application Cat.	window title	acc. obj. name	Content Cat.	Resource Cat.	acc. obj. value	content of EB	datatype properties	Ontology Str. Cat.	content in focus	used res. metadata	Action Cat.	acc. obj. role	used res. content	Ψ
All Categories	-	∼	∼	>	>	≫	>	≫	≫	≫	▷	≫	≫	≫	≫	54
Application Cat.	∼	-	∼	▷	▷	≫	≫	≫	≫	≫	≫	≫	≫	≫	≫	54
window title	∼	∼	-	∼	≫	≫	≫	≫	≫	≫	≫	≫	≫	≫	≫	53
acc. obj. name	<	◁	∼	-	∼	>	∼	>	>	▷	∼	>	≫	≫	▷	50
Content Cat.	<	◁	≪	∼	-	∼	∼	∼	∼	∼	≫	∼	>	▷	≫	46
Resource Cat.	≪	≪	≪	<	∼	-	∼	∼	∼	∼	∼	>	▷	≫	>	46
acc. obj. value	<	≪	≪	∼	∼	∼	-	∼	∼	∼	∼	∼	▷	≫	>	45
content of EB	≪	≪	≪	<	∼	∼	∼	-	∼	∼	∼	∼	>	▷	≫	45
datatype properties	≪	≪	≪	<	∼	∼	∼	∼	-	∼	∼	∼	≫	≫	>	45
Ontology Str. Cat.	≪	≪	≪	◁	∼	∼	∼	∼	∼	-	∼	∼	≫	≫	∼	43
content in focus	◁	≪	≪	∼	≪	∼	∼	∼	∼	∼	-	∼	∼	>	▷	43
used res. metadata	≪	≪	≪	<	∼	<	∼	∼	∼	∼	∼	-	∼	≫	∼	41
Action Cat.	≪	≪	≪	≪	<	◁	◁	<	≪	≪	∼	∼	-	>	∼	40
acc. obj. role	≪	≪	≪	≪	◁	≪	≪	◁	≪	≪	<	≪	<	-	∼	38
used res. content	≪	≪	≪	◁	≪	<	<	≪	<	∼	◁	∼	∼	∼	-	37

Table 7.32: Feature significance matrix for the top 15 features without rank transformation ($T - Test^f$)

7.4.10 Finding the Best Learning Algorithms

Classifier Dominance Matrix: Table 7.34 visualizes the results from the classifier dominance matrix computations as described in Section 7.3.5.2. The order of the best performing classifiers is $NB \succ_{230} J48 \succ_{377} SVM\text{-}lin \succ_{97} KNN\text{-}5 \succ_{42} KNN\text{-}1 \succ_{13} KNN\text{-}35 \succ_{41} KNN\text{-}10$, whereas $c_1 \succ_d c_2$ indicates that classifier c_1 performed d times better than c_2 based on the Ψ.

	Application Cat.	All Categories	window title	acc. obj. name	Content Cat.	datatype properties	Ontology Str. Cat.	content of EB	Resource Cat.	acc. obj. value	content in focus	used res. metadata	Action Cat.	acc. obj. role	acc. obj. role des.	Ψ
Application Cat.	-	~	~	>	≫	≫	≫	≫	≫	≫	≫	≫	≫	≫	≫	54
All Categories	~	-	~	~	>	≫	≫	≫	≫	≫	▷	≫	≫	≫	≫	53
window title	~	~	-	~	≫	≫	≫	≫	≫	▷	≫	≫	≫	≫	≫	53
acc. obj. name	<	~	~	-	~	>	>	~	▷	▷	~	▷	≫	≫	≫	50
Content Cat.	≪	<	≪	~	-	~	~	~	~	~	▷	~	>	>	>	45
datatype properties	≪	≪	≪	<	~	-	~	~	~	~	~	~	≫	≫	≫	45
Ontology Str. Cat.	≪	≪	≪	<	~	~	-	~	~	~	~	~	▷	≫	≫	44
content of EB	≪	≪	≪	~	~	~	~	-	~	~	~	~	~	>	>	44
Resource Cat.	≪	≪	≪	◁	~	~	~	~	-	~	~	>	▷	≫	≫	44
acc. obj. value	≪	≪	◁	◁	~	~	~	~	~	-	~	~	▷	≫	▷	43
content in focus	≪	◁	≪	~	◁	~	~	~	~	~	-	~	~	>	~	42
used res. metadata	≪	≪	≪	◁	~	~	~	~	<	~	~	-	~	>	>	42
Action Cat.	≪	≪	≪	≪	<	≪	◁	~	◁	◁	~	~	-	~	~	39
acc. obj. role	≪	≪	≪	≪	<	≪	≪	<	≪	≪	<	<	~	-	~	39
acc. obj. role des.	≪	≪	≪	≪	<	≪	≪	<	≪	◁	~	<	~	~	-	37

Table 7.33: Feature significance matrix for the top 15 features with rank transformation $(T - Test^{f}_{rt})$

	NB	J48	SVM-lin	KNN-5	KNN-1	KNN-35	KNN-10	Ψ
NB	-	291	360	328	331	336	342	1988
J48	251	-	347	288	292	291	289	1758
SVM-lin	207	211	-	241	242	237	243	1381
KNN-5	208	221	322	-	173	176	184	1284
KNN-1	206	216	321	156	-	164	179	1242
KNN-35	200	222	325	152	166	-	164	1229
KNN-10	195	222	321	136	151	163	-	1188

Table 7.34: Dominance matrix for the classifiers $l \in$ *{J48, KNN-1, KNN-10, KNN-35, KNN-5, NB, SVM-lin}*

Paired T-Tests: Table 7.35 summarizes the statistical significance tests for classifiers as described in Section 7.4.10. Since the results of "with" and "without" rank transformation were the same Table 7.35 shows both. Using each column of this table the following complete partial order of the classifiers was obtained for the different significance levels: {*J48, NB*} $\gg$ {*KNN-1, KNN-5, KNN-10, KNN-35*}$\gg${*SVM-lin*}, whereas the classifiers with insignificant performance differences were grouped into one set.

Paired t-tests performed based on the micro f-measures without and with rank transformation resulted in the same partial order.

The results suggested the conclusion that the *Naïve Bayes* and the *J48 decision tree* learners performed better on this dataset as the *k-Nearest Neighbor* (KNN) algorithms and the *linear Support Vector Machines* (SVM-lin).

	J48	NB	KNN-1	KNN-10	KNN-35	KNN-5	SVM-lin	Ψ
J48	-	~	≫	≫	≫	≫	≫	5
NB	~	-	≫	≫	≫	≫	≫	5
KNN-1	≪	≪	-	~	~	~	≫	1
KNN-10	≪	≪	~	-	~	~	≫	1
KNN-35	≪	≪	~	~	-	~	≫	1
KNN-5	≪	≪	~	~	~	-	≫	1
SVM-lin	≪	≪	≪	≪	≪	≪	-	0

Table 7.35: Significance matrix for the classifiers $l \in$ {J48, KNN-1, KNN-10, KNN-35, KNN-5, NB, SVM-lin} with and without rank transformation (both have equal entries)

7.4.11 Concluding Remarks

In Table 7.36 the results of the task detection laboratory experiment 1 described in this section are summarized. Table 7.36 shows which research questions were investigated as well as the task detection performance results of the UICO approach in comparison with the Dyonipos, *SWISH* and *TaskPredictor 1* approaches. One can observe that the UICO approach outperformed all the other approaches on all research questions investigated.

Evaluations	**UICO**	**Dyonipos**	***SWISH***	**Task P. 1**
Task Models (5 TM)	**88.55%**	83.51%	79.35%	79.42%
Pers. based on Lab. Workst. (4 TM)	**94.85%**	88.66%	81.44%	80.41%
Laboratory Computer (4 TM)	**83.91%**	76.18%	71.91%	71.91%
Personal Computer (4 TM)	**91.78%**	88.44%	84.56%	88.78%
Personal Computer (5 TM)	**91.21%**	89.55%	82.95%	87.50%
Routine vs. Knowledge-Int. T. (5 TM)	**94.94%**	94.55%	94.03%	94.03%
Pers. based on Std. Tasks (5 TM)	**77.14%**	75.24%	72.38%	72.38%
Standard Tasks (5 TM)	**88.41%**	83.18%	77.05%	76.97%
Personal Tasks (5 TM)	**86.00%**	72.36%	73.45%	70.27%
One Expert and User Group (5 TM)	**75.37%**[1]	69.46%	65.52%	68.97%
Expert Group and One User (5 TM)	**98.53%**[2]	97.06%	89.71%	97.06%

Table 7.36: This table displays an overview of the results of the laboratory experiment 1. The highest achieved accuracy value for the evaluation of each investigated research question is marked in bold.

[1] Achieved with an extended feature combination evaluation as described in Section 7.4.7. The standard evaluation methodology described in Section 7.3.3 only reached 72.91% because not all feature combinations were evaluated.

[2] Achieved with an extended feature combination evaluation as described in Section 7.4.8. The standard evaluation methodology described in Section 7.3.3 only reached 92.65% because not all feature combinations were evaluated.

The answers to the research questions were elaborated for the influence of the environment in Section 7.4.11.1, for personal and standard task in Section 7.4.11.2, for routine and knowledge-intensive tasks in Section 7.4.11.3, and for expert task training in Section 7.4.11.4. Concluding remarks and open questions round off this section.

7.4.11.1 Influence of the Computer Environment

The task instances recorded on the laboratory computer and the personal workstations during this experiment were classified to the four task models (Task 1-4) with an accuracy of 83.91% and 91.78% respectively. The difference in terms of accuracy was 7.87% which was unexpected because the laboratory computer seemed to be a more controlled environment. The limitations of the laboratory computer environment were shared among the experiment's participants. It was expected that these limited conditions would result into very similar task executions and hence would increase the task detection performance. However, this was not the case. The better detectability of personal workstation tasks suggested that the performance of classification algorithms was positively influenced by the variety of task executions and by the users' freedom in performing the tasks. The generalizability of these findings have to be further investigated with other tasks as well as other users and in other domains.

A training on laboratory task usage data and testing on personal workstation task executions resulted in an accuracy of 94.85%. The accuracy increased by 3.07% and 10.94% in comparison to only detecting the task models of personal workstation or laboratory task executions respectively. This could be an indication that the computer environment did not have that much influence in how users performed these four tasks and hence limited influence in the task detection performance.

What is further interesting next to the high accuracy values is that these results could be an indication that it would be possible to "train offline". Train offline here means to gather a group of users in a lab for executing tasks of a certain domain and recording these task execution for training the classifier. Packaging this trained classifier within a product and role it out to the whole domain or company for automatic task detection. This would then bring all the benefits of task-based work and learn support to the users. The arguments that (i) these task were very specific to the domain and that (ii) only four tasks were studied are supported at this point.

The influence of adding one further task model and further task instances to the personal workstation environment was also studied. The result was an accuracy of 91.21% which was only 0.57% less accurate than detecting four task models. For getting a clearer picture about the task detection performance in case of a continous adding of new tasks models requires long-term task observation experiments. For generalizing the achieved results further, experiments are suggested. For a discussion about the generalizability of the results the reader is referred to Section 8.1.3.

7.4.11.2 Detecting Personal Tasks based on Standard Tasks

Standard tasks are task that have a *specific goal* which was predefined by the experimenter. Personal tasks are tasks for which the experiment's participant can freely choose the specific goal to accomplish. The results showed that by training on tasks with a shared specific goal the task models of the task instances of personal task instances could be detected with an accuracy of

77.14%. This could indicate that the *specific goal* of a task has an influence in task detection performance when comparing this result to the detection of only standard tasks (88.41%) and only personal tasks (86.00%).

7.4.11.3 Routine vs. Knowledge-Intensive Tasks

The routine task instances (*Task 1-3*) and the knowledge-intensive task instances (*Task 4* and *Task 5*) were correctly identified as routine and knowledge intensive tasks with an accuracy of 94.94%. From the machine learning perspective this was a two-class or binary classification problem which was solved with a very high accuracy on this dataset.

7.4.11.4 Expert(s) and User(s) Tasks

The results of the evaluation in Section 7.4.7 about the performance of detecting user tasks by training on expert tasks when focusing on five task models (*Tasks 1-5*) showed that an accuracy of 72.91% can be reached when utilizing the UICO's *Top* $k = 2$ best single performing feature combination on this dataset. This feature combination was specific to the UICO approach and outperformed the feature and feature combinations proposed by the existing approaches.

Through an extended evaluation of all feature combinations consisting of 2-6 features of the 10 best performing single features an accuracy increase of 2.34% to 75.37% was achieved. This can be seen as an indication that (i) although calculating multiple combinations of the best single performing features only leads to a small accuracy increase and (ii) that one expert is not enough for classifier training in order to reach a high accuracy.

In Section 7.4.8 the reversed question was studied, namely the task classification performance of detecting a single user's tasks by training on tasks from multiple experts. It resulted in an accuracy of 97.06% obtained by the Dyonipos $\mathcal{CW}$ and $\mathcal{ACW}$ as well as the *TaskPredictor 1* approach. A 4.41% lower accuracy was reached by utilizing the best UICO feature combinations based on the standard evaluation methodology. Through an extended evaluation of all feature combinations consisting of 2-6 features of the 10 best performing single features an accuracy of 98.53% reached which outperformed all the existing features and feature combinations of the existing approaches.

7.4.11.5 Remarks about the Experiment

The small number of task models investigated in this domain had multiple reasons. The first one was that the same participants had to perform the tasks twice, once on the laboratory computer and once on their personal workstations. Since the experiment should not last longer than 90 min for each setting, the number of tasks were limited. From the participant's point of view it is an exhausting task to perform multiple task in a short period of time.

The tasks to be executed were chosen together with the participants beforehand during preliminary workshops such that the tasks reflect their actual work. Since the domain was a research company the routine tasks were about standard administration tasks and the knowledge-intensive tasks about journey planning and the organization of project meetings.

The execution of the experiment and the data collection went well. However, the time required for saving the observed usage data to the file system was very long. Sometimes it took over an hour because of the mass of usage data observed. At the point in time when the usage data collection took place the optimization mechanisms about separating triples into multiple named graphs were not yet implemented in the prototype. Although the application freezed and could not be closed during the saving time this did not cause any problems for the participants since they left their computer running in the background after returning to their daily work.

The laboratory computer was not the fastest machine such that a few participants mentioned that they were not able to be as fast and productive as usual in executing the tasks. These issues might have introduced a bias in the usage data collection process.

7.5 Laboratory Experiment 2 - Computer Science Students

The experiment described in this section investigated the domain of *"computer science students of Graz University of Technology in Austria"*. Preceding informal interviews showed that computer science students performed routine as well as knowledge-intensive tasks during their university time. This experiment was conducted to get further insights into the capabilities of automatic task detection for computer science tasks in a controlled setting. Four laboratory computers were setup and prepared for the experiment (see Section 7.5.1.1). 10 computer science students participated in this experiment. These students were considered as experts of the investigated domain. They allowed the observation of their user interaction context during their task executions and made it freely available for the evaluations described in this section.

Following questions were investigated:

- Can the task model of the task instances be automatically detected?
- Can the task models of the task instances from personal task executions be detected based on predefined standard task executions for training the classifier?
- Can the task model of the task instances be automatically detected when evaluating routine and knowledge-intensive tasks separately?
- Can the type of task be automatically detected when distinguishing routine and knowledge-intensive tasks?
- Which context features are most discriminative for the studied tasks?
- Which learning algorithm performs best in terms of automatic task detection on the collected dataset?

7.5.1 Experiment Design

The comparison was *within subjects* and the manipulations were achieved by (i) the type of task (standard or personal) and (ii) the task to be executed (7 different tasks). The experiment was designed in three phases. Phase 1 was the phase before the subjects executed the tasks on the laboratory computers. Phase 2 was the user interaction context observation phase. This phase was followed by Phase 3, which included the evaluation of the observed user interaction context about the task executions. A description of the steps in the phases of the experiment as well as the obtained results is given in the following sections.

Manipulation 1: Standard and Personal Tasks
Standard tasks are tasks that have a specific goal, like described in Section 7.4.1. By having multiple users executing a task with the same specific goal, similar task instances were expected. Tasks performed on behalf of the experiment's participants themselves are referred to as *personal tasks*. The order to start with a standard or a personal task of a task model was randomized.

Manipulation 2: Tasks
The second manipulation resulted from varying the tasks. Seven tasks chosen by four domain experts were studied. The tasks had different characteristics, like for example, complexity,

estimated execution time, number of involved resources, granularity and so on. Before starting the experiment the subjects were asked to read through the task descriptions and to confirm that they understood the tasks. The order in which the subjects executed the tasks was randomized. Furthermore it was randomized on which behalf they started to perform the tasks: (i) on behalf of themselves and (ii) on behalf of an artificial student (*persona*). The seven task models are listed bellow. A detailed description of them including the example tasks is given in Section 8.5.

1. *Routine Tasks*

Task 1: Register for an examination
Task 2: Finding course dates
Task 3: Reserve a book in the university's library
Task 4: Course registration

2. *Knowledge-intensive Tasks*

Task 5: Algorithm programming
Task 6: Prepare a scientific talk
Task 7: Plan a study trip

<u>Example of a task model:</u>

[Task 1] Register for an examination

Task Model (Description):
The registration for an exam is a task in which the student has to sign in to the TUGonline system and to register himself to a particular examination. Registration is required for doing and passing an exam at the university.

Task Instance (Task Standard):
Suppose you are a student named Georg Kompacher studying *Software Development and Business Management* at Graz University of Technology in Austria. You want to register for a specific examination named *"Verifikation und Testen"* taking place on the 15th May 2009. To register you have to use the TUGonline system. Use the Microsoft Internet Explorer and visit the web site `http://online.tugraz.at` and sign in with the following account: user name: `<wttestaccountuser>` and password: `<wttestpassword>`. Search for the examination for *"Verifikation und Testen"* and register for that exam.

Task Instance (Task Personal):
Open the browser of your choice. Search for an examination at the Graz University of Technology you want to participate and register for that exam using the TUGonline system.

7.5.1.1 Phase 1 - Before the Experiment

The selection of the appropriate tasks for the investigated domain and for the asked research questions were a key part of this phase. There had to be a almost balanced ratio between routine and knowledge-intensive tasks. At the beginning the computer science students were asked what kind of tasks they do during their university's curriculum. From the experiment's participants list of tasks, four computer science students selected four routine and four knowledge-intensive tasks during several workshops. In the task selection process the task execution times were measured by three students. The durations of the task executions of these students were averaged to get an estimation of how long it would take the participants to complete the tasks. Since a usage data recording session for a participant should not take longer than 90 minutes the longest knowledge-intensive task was deselected because it was the longest one.

The second key part of this phase was to prepare the experiment's computers. All the used laboratory computers were notebooks and should have the same software as well as the appropriate software for a computer science student installed. The operating systems were Microsoft Windows XP and Vista which were the mainly used operating systems of the experiment's participants. The list of the software products was compiled based on the experiences of the four higher semester computer science students and included all the software that they have used in their university courses.

Following software products were installed:

- Windows XP or Vista
- Microsoft Office 2003 or 2007
- Internet Explorer 7 or 8
- Integrated development environments (IDEs): Eclipse 3.x, MS Visual Studio 2008, NetBeans 6.x
- Diverse editors: Emacs, JEdit++, Notepad++, Vim, Microsoft Notepad
- Compiler: C, C++, C#, Java, Python, Perl, Ruby
- KnowSe Prototype and the context sensors for the installed applications

7.5.1.2 Phase 2 - User Context Observation

The user interaction context observation phase of the experiment took about two weeks with a maximum of two sessions per day. Per session no more than four subjects were asked to execute their tasks in parallel which allowed ongoing servicing during the experiment if required. There was a five minute introduction about how to use the user interaction context observation prototype *KnowSe* (see Chapter 6) but no specific training period as in the first laboratory experiment (see Section 7.4). However, the participants seemed to become very fast familiar with the handling of the prototype. Only very few questions were raised about the usage of the prototype. The prototype used in this experiment is described in Section 6.3.

7.5.1.3 Phase 3 - Task Detection

The evaluation and the analysis of the data played the main part of phase 3 of the experiment. The well known machine learning toolkit *Weka* [Witten & Frank, 2005] in combination with Weka integration of the libSVM [Chang & Lin, 2001] were utilized to study appropriate parameters and algorithms for attribute selection and automatic task detection performance. The training instances for the learning algorithms were built based the user interaction context ontology (UICO) as described in Section 5.

7.5.2 Research Question: Can the task model of the task instances be automatically detected?

The goal of this evaluation was to answer the question *"Can the task model of the task instances be automatically detected?"*. The dataset on which this question was investigated contained 134 tasks from 10 users. The distribution of the task instances in respect to the classes is shown in Table 7.37. An overview of all results about the performance of detecting the tasks (*Task 1-7*) is given in Table 7.38. A training instance was built for each task instance independently. For the evaluation of the task classification the evaluation method stratified 10-fold cross-validation was used (see Section 7.3.3).

Classes	**Class Instances**
Task 1	19
Task 2	20
Task 3	20
Task 4	17
Task 5	20
Task 6	19
Task 7	19
Dataset CV	**134**

Table 7.37: This table shows the distribution of the training/test instances for the different task classes ranging from Task 1 to Task 7 which were recorded on the laboratory computers.

Feature Categories: The feature category which achieved the highest accuracy value was the combination of all 50 features of all categories with the Naïve Bayes algorithm (a=94.84%, g=2500, p=0.99, r=0.95). This combination was also the best one among all single performing features and the *Top k* features. Very close behind with the same algorithm was the *application category* with approximately 1.5% lower accuracy (l=NB, a=93.30%, g=500, p=0.99, r=0.94). The *resource category* obtained an accuracy of 87.91% (l=J48, a=87.91%, g=500, p=0.97, r=0.87) which was only sufficient for the 12th global rank. The number of attributes of the best classifier runs were between 75 and 500 attributes except for the combination of all 50 features which required 2500 attributes for achieving the best overall accuracy.

Single Features: The best performing single feature was the *used res. metadata* feature (l=J48, a=90.33%, g=300, p=0.98, r=0.91). The *acc. obj. name* feature was the one with the second highest accuracy (l=NB, a=88.79%, g=50, p=0.98, r=0.89) and only about 2.5% worse than the best one. The third rank of the best performing single features went to the *window title* feature with an accuracy of 88.13% on 75 attributes with the J48 learner (p=0.98, r=0.89). The range of the numbers of attributes for the best classifier runs of the single performing features was between 5 and 300 attributes except for the features *E&EB res. switch seq.* and *E*

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	NB	2500	94.84	0.99	0.95	1
	2	Application Cat.	NB	500	93.30	0.99	0.94	7
	3	Resource Cat.	J48	200	87.91	0.97	0.87	17
	4	Ontology Str. Cat.	J48	359	81.59	0.96	0.82	19
	5	Action Cat.	NB	250	80.49	0.96	0.81	20
	6	Switching Seq. Cat.	NB	100	73.85	0.94	0.73	26
	7	Content Cat.	J48	75	50.88	0.85	0.49	36
Single Feat.	1	used res. metadata	J48	300	90.33	0.98	0.91	12
	2	acc. obj. name	NB	50	88.79	0.98	0.89	14
	3	window title	J48	75	88.13	0.98	0.89	15
	4	datatype properties	J48	221	82.14	0.96	0.83	18
	5	res. types interact.	J48	27	79.07	0.95	0.79	21
	6	acc. obj. value	KNN-5	75	78.35	0.95	0.79	22
	7	res. interact.	NB	50	76.15	0.95	0.75	23
	8	used res. interact.	NB	50	75.49	0.94	0.76	24
	9	used resources	NB	125	74.73	0.94	0.74	25
	10	applications interact.	J48	65	72.31	0.93	0.71	27
	11	concept instances	NB	81	71.76	0.94	0.71	28
	12	EB res. interact.	NB	175	69.56	0.92	0.69	29
	13	objecttype properties	J48	57	61.81	0.90	0.61	30
	14	E&EB res. switch seq.	J48	500	58.19	0.88	0.59	31
	15	E level res. switch seq.	J48	1238	57.64	0.88	0.57	32
Top k Feat.	1	Top $k = 6$	J48	100	94.07	0.99	0.94	2
	2	Top $k = 15$	NB	4000	94.01	0.99	0.94	3
	3	Top $k = 5$	NB	175	93.35	0.99	0.94	4
	4	Top $k = 20$	NB	3000	93.35	0.99	0.94	5
	5	Top $k = 9$	NB	750	93.30	0.99	0.94	6
	6	Top $k = 10$	NB	750	93.24	0.99	0.94	8
	7	Top $k = 8$	NB	150	92.58	0.99	0.93	9
	8	Top $k = 7$	NB	500	92.58	0.99	0.93	10
	9	Top $k = 4$	NB	250	91.87	0.98	0.91	11
	10	Top $k = 3$	NB	750	89.62	0.98	0.90	13
	11	Top $k = 2$	KNN-5	750	88.13	0.98	0.88	16

Table 7.38: Overview of the best results about the performance of detecting tasks (Tasks 1-7) by stratified 10-fold cross-validation for each feature category, for all feature categories combined, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

level res. switch seq. that achieved their highest accuracy values with 500 and 1238 attributes respectively. In comparison with the *Top k* feature combinations just the two best single features outperformed the two worst *Top k* feature combinations.

Top k Features: The *Top k* feature combinations achieved accuracy values ranging from 88.13% to 94.07%. The highest and the second highest accuracy resulted from the *Top* $k = 6$ features (l=J48, a=94.07%, g=100, p=0.99, r=0.94) and the *Top* $k = 15$ features (l=NB, a=94.01%, g=4000, p=0.99, r=0.94). The *Top* $k = 15$ had its best run on 4000 attributes whereas the *Top* $k = 6$ feature only required 100 attributes. Based on 175 attributes the *Top* $k = 5$ features achieved the third highest accuracy value with 93.35% and the Naïve Bayes algorithm (p=0.99, r=0.94). The best eight *Top k* features had a hight precision of 0.99 and a recall between 0.93 and 0.94. The range of the numbers of attributes for the best classifier runs of the *Top k* performing features was between 100 and 750 except for the *Top* $k = 15$ and the *Top* $k = 20$ features which had their best classifier runs with 4000 and 3000 attributes respectively.

Set	R_G	f	l	g	a	p	r
	1	$\mathcal{CW}$	NB	125	95.49	0.99	0.96
	2	$\mathcal{ACW}$	NB	125	94.73	0.99	0.95
	3	$\mathcal{W}$	NB	50	94.01	0.99	0.94
Dyonipos	4	$\mathcal{AW}$	SVM-$C2^{-2}$	50	93.35	0.99	0.94
	7	$\mathcal{AC}$	J48	25	67.25	0.92	0.66
	8	$\mathcal{A}$	J48	24	55.82	0.88	0.56
	9	$\mathcal{C}$	SVM-$C2^{5}$	125	45.71	0.83	0.45
SWISH	6		J48	50	88.90	0.98	0.89
TaskPredictor 1	5		NB	300	93.24	0.99	0.92

Table 7.39: Overview of the best results about the performance of detecting tasks (Tasks 1-7) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Comparison with existing approaches: A comparison with existing approaches showed that the best overall tested combinations of the UICO features were the combination of all 50 features. It outperformed the *SWISH* and the *TaskPredictor 1* approach by 5.94% and 1.6% accuracy respectively. Only the $\mathcal{CW}$ combination of the Dyonipos approach had a 0,65% higher accuracy than the best UICO feature combination. The top four performed features of the Dyonipos approach achieved over 93% accuracy, 0.99 micro precision and over 0.94 micro recall. All approaches had an accuracy value between 88.9% and 95.49%. Part of the best feature combinations was always the *window title* feature. The number of attributes of the *SWISH* , the *TaskPredictor 1* and the best Dyonipos feature combination achieved the highest accuracy values within 50 and 300 attributes whereas the best UICO feature combination required 2500 attributes. The second best UICO feature combination, the *Top* $k = 6$, laid with 100 attributes in the range of the compared approaches. A detailed comparison of the feature and classifier performance evaluation of the existing approaches is given in Table 7.39.

Concluding Remarks: Most of the *Top k* features outperformed the feature categories and the single features. The best overall performances were achieved by the combination of all 50 features and the *Top* $k = 6$ with 94.84% and 94.07% respectively. Only the $\mathcal{CW}$ combination of the Dyonipos approach had a 0,65% higher accuracy. This was not significantly higher and might have originated from the random selection process in the 10-fold cross-evaluation for the classifiers. Another reason could have been the UICO feature evaluation method, because the task detection performance of the combination of *content of EB* feature and *window title* were not evaluated. For a further discussion about the UICO feature evaluation method the reader is referred to Section 7.7. All in all the UICO and the three existing approaches performed very well on this dataset with an accuracy value between 88.9% and 95.49% percent. These results showed that these seven tasks could be detected with a very high accuracy based on this dataset.

7.5.3 Research Question: Can the task models of the task instances from personal task executions be detected based on predefined standard task executions for training the classifier?

The goal of this evaluation was to answer the question *"Can the task models of the task instances from personal task executions be detected based on predefined standard task executions for training the classifier?"* The dataset on which this question was investigated contained 134 tasks from 10 users. Among these 134 tasks there were 68 *standard tasks* and 66 *personal tasks.* Standard tasks are tasks that have a *specific goal*, as described in Section 7.4.1. Personal tasks were tasks in which the subjects performed the task on behalf of themselves. The distribution of the task instances in respect to the classes (*Task 1-7*) as well as in respect to the standard and personal tasks is shown in Table 7.40.

Classes	**Standard Tasks**	**Personal Tasks**	**Sum**
Task 1	10	9	19
Task 2	10	10	20
Task 3	10	10	20
Task 4	8	9	17
Task 5	10	10	20
Task 6	10	9	19
Task 7	10	9	19
Dataset (Train/Test)	**68**	**66**	134

Table 7.40: This table shows the distribution of the training instances (standard tasks) and test instances (personal tasks) for the different task classes ranging from Task 1 to Task 7.

An overview of all results about the performance of detecting the tasks (*Task 1-7*) is given in Table 7.41. A training instance was built for each standard task and a test instance for each personal task independently. For the evaluation of the task classification the evaluation method train and test set (see Section 7.3.3) was used whereas the standard task instances constituted the training set and the personal task instances the test set.

Feature Categories: The feature category which achieved the highest accuracy value was the combination of all 50 features of all categories (l=NB, a=92.42%, g=7500, p=0.99, r=0.92). This was also the best among all UICO feature combinations and the *Top k* features with a global rank R_G=1. Close behind with the same algorithm, the Naïve Bayes, was the *application category* with a 3.03% lower accuracy (l=NB, a=93.30%, g=250, p=0.98, r=0.90) with only 250 attributes instead of 7500 attributes. The *resource category* obtained an accuracy of 81.82% (l=NB, a=81.82%, g=75, p=0.96, r=0.81) which was only sufficient for the global rank R_G=16. The number of attributes of the best runs were between 75 and 500 attributes except for the combination of all 50 features which required 7500 attributes for achieving the best overall accuracy. All feature categories performed best with the Naïve Bayes classifier except the *ontology structure category* which achieved its best classifier run with the KNN-35 classifier.

Single Features: The best performing single feature was the *acc. obj. name* feature (l=NB, a=86.36%, g=50, p=0.97, r=0.86). Only 4.54% less accurate was the *window title* feature with the second highest accuracy (l=NB, a=81.82%, g=125, p=0.96, r=0.82). The *used res. metadata* feature achieved the third rank of the best performing single features with an accuracy of 78.79% on 200 attributes with the KNN-35 learner (p=0.96, r=0.82). The range of the numbers of attributes for the best classifier runs of the best 15 single performing features was between 3 and 250 attributes. The only exception were the feature *acc. obj. value* and *E level res. switch seq.* that had its best classifier run with 750 attributes. In comparison with the *Top k* feature combinations all *Top k* feature combinations outperformed the best single feature except the worst two *Top k* feature combinations which were *Top* $k = 4$ and *Top* $k = 3$.

Top k Features: The *Top k* feature combinations achieved accuracy values ranging from 81.82% to 90.91%. The highest accuracy resulted from the *Top k* with $k = \{2, 6, 7, 8, 9, 10\}$ features (l=NB, a=90.91%, p=0.98, r=0.91). All these *Top k* feature combinations had their best classifier runs with 300 attributes except the *Top* $k = 2$ which only required 125 attributes. The range of the numbers of attributes for the best classifier runs of the *Top k* performing features was between 125 and 300 attributes. All the *Top k* feature combinations achieved their highest accuracy with the Naïve Bayes classifier.

Comparison with existing approaches: A comparison with existing approaches showed that the best overall tested combinations of the UICO features, the combination of all 50 features (l=NB, a=92.42%, g=7500, p=0.99, r=0.92), outperformed all the existing approaches. The *SWISH* , the *TaskPredictor 1* and the best Dyonipos feature combination $\mathcal{AW}$ were outperformed by 13.63%, 4.54% and 1.51% respectively. The *SWISH* approach had its best classifier run with 423 attributes with the Naïve Bayes algorithm (l=NB, a=78.79%,p=0.96,

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	NB	7500	92.42	0.99	0.92	1
	2	Application Cat.	NB	250	89.39	0.98	0.90	8
	3	Resource Cat.	NB	75	81.82	0.96	0.81	16
	4	Action Cat.	NB	100	77.27	0.95	0.77	19
	5	Ontology Str. Cat.	KNN-35	75	65.15	0.92	0.65	22
	6	Switching Seq. Cat.	NB	50	62.12	0.91	0.62	24
	7	Content Cat.	NB	500	48.48	0.85	0.49	34
Single Feat.	1	acc. obj. name	NB	50	86.36	0.97	0.86	12
	2	window title	NB	125	81.82	0.96	0.82	15
	3	used res. metadata	KNN-35	200	78.79	0.96	0.78	17
	4	acc. obj. value	NB	750	77.27	0.95	0.77	18
	5	datatype properties	J48	50	72.73	0.94	0.73	20
	6	res. types interact.	KNN-35	10	69.70	0.93	0.69	21
	7	concept instances	J48	25	65.15	0.92	0.65	23
	8	used res. interact.	NB	25	59.09	0.90	0.59	25
	9	res. interact.	NB	25	59.09	0.90	0.59	26
	10	applications interact.	NB	25	57.58	0.89	0.57	27
	11	objecttype properties	NB	25	54.55	0.88	0.53	28
	12	acc. obj. role des.	NB	54	53.03	0.87	0.53	29
	13	EB res. interact.	NB	250	51.52	0.87	0.52	30
	14	acc. obj. role	J48	10	51.52	0.86	0.51	31
	15	E type switch seq.	SVM-$C2^{10}$	3	50.00	0.86	0.50	32
Top k Feat.	1	Top $k = 9$	NB	300	90.91	0.98	0.91	2
	2	Top $k = 8$	NB	300	90.91	0.98	0.91	3
	3	Top $k = 7$	NB	300	90.91	0.98	0.91	4
	4	Top $k = 6$	NB	300	90.91	0.98	0.91	5
	5	Top $k = 2$	NB	125	90.91	0.98	0.91	6
	6	Top $k = 10$	NB	300	90.91	0.98	0.91	7
	7	Top $k = 5$	NB	300	89.39	0.98	0.89	9
	8	Top $k = 20$	NB	250	89.39	0.98	0.89	10
	9	Top $k = 15$	NB	250	89.39	0.98	0.89	11
	10	Top $k = 4$	NB	250	83.33	0.97	0.83	13
	11	Top $k = 3$	NB	200	81.82	0.96	0.82	14

Table 7.41: Overview of the best results about the performance of detecting personal tasks by training on standard tasks for each feature category, for all feature categories combined, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

r=0.78). The *TaskPredictor 1* approach also performed best with the Naïve Bayes algorithm but required more than double the number of attributes than the *SWISH* and the Dyonipos approaches. Among the existing approaches the Dyonipos one with the $\mathcal{AW}$ resulted in the highest accuracy (l=NB, a=90.91%,p=0.98, r=0.91). All the existing approaches had their best classifier runs with the Naïve Bayes algorithm. A detailed comparison of the feature and classifier performance evaluation of the existing approaches is given in Table 7.42.

Set	R_G	f	l	g	a	p	r
	1	$\mathcal{AW}$	NB	300	90.91	0.98	0.91
	2	$\mathcal{CW}$	NB	75	89.39	0.98	0.89
	3	$\mathcal{W}$	NB	50	89.39	0.98	0.89
Dyonipos	4	$\mathcal{ACW}$	NB	75	89.39	0.98	0.89
	7	$\mathcal{AC}$	NB	200	59.09	0.90	0.60
	8	$\mathcal{A}$	NB	17	50.00	0.85	0.49
	9	$\mathcal{C}$	NB	250	40.91	0.81	0.41
SWISH	6		NB	423	78.79	0.96	0.78
TaskPredictor 1	5		NB	1000	87.88	0.98	0.88

Table 7.42: Overview of the best results about the performance of detecting personal tasks by training on standard tasks for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are given.

Influence of the "Specific Goal": The task detection performance was also evaluated based on *standard task* and *personal tasks* separately. This was done in order to study the influence of the *specific goal* in the automatic task detection performance.

The results of the evaluation for the *standard tasks* unveiled a task detection performance of 98.57% accuracy on 300 attributes with the Naïve Bayes algorithm (p=1.00, r=0.99) with the *Top* k = 5 best performing single features combination. The best five single performing features were the *window title* feature (l=J48, a=97.14%, g=50, p=0.99, r=0.97), the *used res. metadata* feature (l=KNN-10, a=92.62%, g=250, p=0.99, r=0.93), the *acc. obj. name* feature (l=NB, a=91.43%, g=50, p=0.98, r=0.91), the *res. interact.* feature (l=NB, a=89.05%, g=25, p=0.97, r=0.89), and the *acc. obj. value* feature (l=KNN-1, a=88.33%, g=75, p=0.97, r=0.88). The best feature category was the combination of all features of all categories with an accuracy of 95.71% (l=NB, g=300, p=0.99, r=0.96).

The task detection of the *personal tasks* resulted in an accuracy of 94.05% with the *Top* k = 4 best performing single features combination (l=J48, g=125, p=0.99, r=0.94). The best three single performing features were the *acc. obj. name* feature (l=NB, a=90.95%, g=50, p=0.98, r=0.91), the *window title* feature (l=KNN-10, a=86.67%, g=10, p=0.97, r=0.87), the *datatype properties* feature (l=J48, a=86.67%, g=75, p=0.97, r=0.87) and the *used res. metadata* feature (l=J48, a=86.43%, g=75, p=0.97, r=0.86). The best feature category was the combination of all features of all categories with an accuracy of 93.81% (l=J48, g=200, p=0.99, r=0.94).

The difference of the accuracy of detecting standard and personal tasks, i.e., task with a single predefined specific goal and without one, was 4,52% in terms of accuracy. The highest accuracy values were achieved in both evaluations with the J48 decision tree learner on a rather small number of attributes. For standard and personal tasks the highest accuracy values were reached with 300 and 125 attributes respectively.

Concluding Remarks: The majority of the *Top k* features outperformed the feature categories and the single features. The best overall performances were achieved by the combination of all 50 features and the *Top k* with $k = \{2, 6, 7, 8, 9, 10\}$ with 92.42% and 90.91% respectively. The *SWISH* , the *TaskPredictor 1* and the best Dyonipos feature combination $\mathcal{AW}$ were outperformed by 13.63%, 4.54% and 1.51% respectively.

These results showed that the task model of a personal task execution can be detected with an accuracy of over 90% when training on standard task executions on the gathered dataset. In other words, it seemed to be possible to have a standard set of tasks executed once by users for the classifier training purposes in order to allow an accurate classification of personal tasks to the task models.

Regarding the influence of the *specific goal* in the automatic task detection performance, the obtained results suggested that there was a rather small influence. The accuracy for standard tasks was only 4.52% higher than the one for personal tasks.

7.5.4 Research Question: Can the task model of the task instances be automatically detected when evaluating routine and knowledge-intensive tasks separately?

The goal of the following evaluations was to answer the question *"Can the task model of the task instances be automatically detected when evaluating routine and knowledge-intensive tasks separately?"*. The datasets on which this question was investigated contained 134 tasks from 10 users. Among these 134 tasks there were 76 *routine tasks* (*Task 1-4*) and 58 *knowledge-intensive tasks* (*Task 5-7*). The distribution of the task instances in respect to the classes (*Task 1-7*) as well as in respect to the task type (routine and knowledge-intensive) is shown in Table 7.43. For both, the routine task and the knowledge-intensive task dataset the task classification performance and the discriminative power of features were evaluated.

An overview of all results about the performance of detecting routine tasks and knowledge-intensive tasks is given in Table 7.44. These results were achieved by applying stratified 10-fold cross-validation on the routine and the knowledge-intensive tasks dataset.

7.5.4.1 Routine Tasks

The evaluation of the task detection performance for routine tasks was a 4-class classification problem because four task models (*Task 1-4*) were studied.

Feature Categories: The best feature category was the one with the combination of all 50 features which correctly identified 93.57% of the routine task classes (l=NB, g=750, p=0.98, r=0.94) and also achieved the third best result among all tested UICO feature combinations. Only 1.25% behind in terms of accuracy was *resource category* (l=KNN-35, a=92.32%, g=175,

Type	**Classes**	**Class Instances**	**Sum**
Routine Tasks	Task 1	19	76
	Task 2	20	
	Task 3	20	
	Task 4	17	
Knowledge Intensive Tasks	Task 5	20	58
	Task 6	19	
	Task 7	19	
Dataset CV			**134**

Table 7.43: This table shows the distribution of the training/test instances for stratified 10-fold cross-validation for the routine tasks Task 1 to Task 4 and the knowledge-intensive tasks Task 5 to Task 7 which were recorded on laboratory computers from 10 users.

p=0.97, r=0.92). The third place in the category ranking went to the *application category* with an accuracy of 92.14% (l=J48, g=25, p=0.97, r=0.92), which was only 1.43% worse than the best performing feature category. In the overall ranking (R_G) the second and third best feature categories only achieved the 15th and 16th place respectively. The range of the number of attributes for training the classification algorithms spanned from 25 to 650 attributes. The highest accuracy among the feature categories were achieved with 750 attributes.

Single Features: The performance of each single feature was evaluated separately and showed that the *window title* feature was with an accuracy of 93.39% on only 25 attributes the best performing single feature (l=KNN-5, p=0.93, r=0.84). The *used res. metadata* feature obtained only a 1.07% lower accuracy with 92.32% (l=NB, g=75, p=0.97, r=0.92). The *acc. obj. name* feature reached the third place of the best performing single features with an accuracy of 89.64% and the KNN-35 learner on 10 attribtues (p=0.96, r=0.89). The range of the numbers of attributes for the best classifier runs of the best 15 single performing features was between 3 and 75 except for the *EB res. interact.* (g=280) and the *E level res. switch seq.* (g=472) features.

Top k Features: The classification performance of the combination of the *Top k* best single performing features ranged from 92.32% to 94.64% accuracy. The *Top k = 8* features obtained the best result with the KNN-5 classifier on 100 attributes (a=94.64%, p=0.98, r=0.94). This combination also achieved the highest accuracy in comparison to all evaluated single features and feature categories. The precision and recall values of the Top k features were close together between p=0.97 and p=0.98 and r=0.92 and r=0.94 respectively. The range of the numbers of attributes for the best runs of the *Top k* performing features was between 75 and 750 attributes. All the *Top k* feature combinations outperformed all the other single performing features and feature categories except the *resource category* which achieved 93.57% accuracy.

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	NB	750	93.57	0.98	0.94	3
	2	Resource Cat.	KNN-35	175	92.32	0.97	0.92	15
	3	Application Cat.	J48	25	92.14	0.97	0.92	16
	4	Ontology Str. Cat.	J48	385	85.54	0.94	0.86	19
	5	Action Cat.	NB	25	80.54	0.92	0.80	21
	6	Switching Seq. Cat.	NB	150	72.68	0.88	0.72	28
	7	Content Cat.	KNN-35	347	54.46	0.76	0.54	33
Single Feat.	1	window title	KNN-5	25	93.39	0.98	0.94	8
	2	used res. metadata	NB	75	92.32	0.97	0.92	14
	3	acc. obj. name	KNN-35	10	89.64	0.96	0.89	17
	4	acc. obj. value	J48	25	86.79	0.95	0.88	18
	5	datatype properties	KNN-10	3	85.54	0.94	0.85	20
	6	concept instances	KNN-1	3	79.11	0.91	0.79	22
	7	res. interact.	KNN-10	25	78.04	0.90	0.78	23
	8	used resources	NB	10	76.07	0.89	0.76	24
	9	EB res. interact.	J48	280	75.18	0.90	0.75	25
	10	used res. interact.	J48	10	74.46	0.88	0.74	26
	11	res. types interact.	J48	13	74.29	0.88	0.72	27
	12	applications interact.	SVM-$C2^{-1}$	5	71.43	0.88	0.72	29
	13	objecttype properties	KNN-10	25	63.57	0.83	0.61	30
	14	E level res. switch seq.	J48	472	62.32	0.81	0.60	31
	15	E&EB res. switch seq.	NB	25	60.89	0.81	0.60	32
Top k Feat.	1	Top $k = 8$	KNN-5	100	94.64	0.98	0.94	1
	2	Top $k = 7$	KNN-1	500	93.75	0.98	0.94	2
	3	Top $k = 4$	NB	75	93.57	0.98	0.94	4
	4	Top $k = 20$	NB	75	93.57	0.98	0.94	5
	5	Top $k = 3$	NB	750	93.57	0.98	0.92	6
	6	Top $k = 6$	KNN-5	100	93.57	0.97	0.92	7
	7	Top $k = 15$	KNN-1	175	93.39	0.98	0.92	9
	8	Top $k = 2$	NB	75	93.21	0.98	0.94	10
	9	Top $k = 9$	KNN-5	150	92.50	0.97	0.92	11
	10	Top $k = 10$	J48	750	92.50	0.97	0.92	12
	11	Top $k = 5$	NB	75	92.32	0.97	0.92	13

Table 7.44: Overview of the best results about the performance of detecting routine tasks (Tasks 1-4) by stratified 10-fold cross-validation for each feature category, for all feature categories combined, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

Set	R_G	f	l	g	a	p	r
	3	$\mathcal{W}$	SVM-$C2^{10}$	25	93.39	0.98	0.94
	4	$\mathcal{AW}$	SVM-$C2^{-2}$	25	93.21	0.97	0.94
	5	$\mathcal{CW}$	KNN-35	10	93.21	0.97	0.92
Dyonipos	6	$\mathcal{ACW}$	SVM-$C2^{10}$	25	92.32	0.97	0.92
	7	$\mathcal{AC}$	KNN-5	149	55.89	0.77	0.55
	8	$\mathcal{C}$	SVM-$C2^{-5}$	50	54.46	0.76	0.51
	9	$\mathcal{A}$	KNN-1	6	33.39	0.57	0.31
SWISH	2		KNN-5	25	93.57	0.98	0.94
TaskPredictor 1	1		KNN-5	25	93.75	0.98	0.94

Table 7.45: Overview of the best results about the performance of detecting routine tasks (Tasks 1-4) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Comparison with existing approaches: The *SWISH* , the *TaskPredictor 1* and some feature combinations of the Dyonipos approach achieved over 93% accuracy on this dataset. In comparison with the UICO feature performances the existing approaches were only about 1% worse than the best UICO feature combination, the *Top k=8*. A detailed overview of the feature and classifier performances of the existing approaches is given in Table 7.45.

Concluding Remarks: The evaluations of the task detection performance of the routine tasks (*Task 1-4*) on this task dataset showed that an accuracy of approximately 95% was reached when utilizing UICO feature combinations and that the UICO approach slightly outperformed existing approaches.

7.5.4.2 Knowledge-Intensive Tasks

The evaluation of the task detection performance for knowledge-intensive tasks was a 3-class classification problem because three task models (*Task 1-3*) were studied.

Feature Categories: The combination of all 50 features (l=KNN-35, g=25), the *switching sequence category* (l=NB, g=25), *application category* (l=NB, g=150) and the *action category* (l=NB, g=10) achieved 100% accuracy in detecting the task models of the knowledge-intensive task instances. The range of utilized attributes was between 10 and 150 attributes. The worse feature categories performed between 86.67% and 96.67% accuracy.

Single Features: The best performing single feature was the *res. types interact.* feature with 100% accuracy (l=NB, g=3). The *window title* feature, the *app. switch seq.* feature, and the *applications interact.* feature achieved the second highest accuracy with only 1.77% behind the best performing single feature. The range of the number of attributes of the best 15 performing

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	KNN-35	25	100.00	1.00	1.00	1
	2	Switching Seq. Cat.	NB	25	100.00	1.00	1.00	13
	3	Application Cat.	NB	150	100.00	1.00	1.00	15
	4	Action Cat.	NB	10	100.00	1.00	1.00	16
	5	Resource Cat.	NB	300	96.67	0.98	0.97	20
	6	Ontology Str. Cat.	KNN-35	50	96.67	0.98	0.97	21
	7	Content Cat.	SVM-$C2^{-1}$	175	86.67	0.93	0.87	31
Single Feat.	1	res. types interact.	NB	3	100.00	1.00	1.00	14
	2	window title	NB	383	98.33	0.99	0.98	17
	3	app. switch seq.	NB	50	98.33	0.99	0.98	18
	4	applications interact.	NB	25	98.33	0.99	0.98	19
	5	application name	NB	10	96.67	0.98	0.97	22
	6	acc. obj. name	NB	200	96.67	0.98	0.97	23
	7	used res. metadata	NB	1000	95.00	0.97	0.95	24
	8	concept instances	NB	50	95.00	0.97	0.95	25
	9	used res. interact.	J48	50	95.00	0.97	0.95	26
	10	datatype properties	J48	224	92.67	0.96	0.93	27
	11	res. interact.	NB	300	91.67	0.95	0.92	28
	12	objecttype properties	KNN-1	10	89.67	0.94	0.90	29
	13	acc. obj. role des.	KNN-35	5	88.33	0.93	0.88	30
	14	E&EB res. switch seq.	SVM-$C2^{1}$	200	86.33	0.93	0.87	32
	15	used resources	NB	25	86.33	0.92	0.87	33
Top k Feat.	1	Top $k = 9$	KNN-35	50	100.00	1.00	1.00	2
	2	Top $k = 8$	KNN-35	50	100.00	1.00	1.00	3
	3	Top $k = 7$	KNN-35	75	100.00	1.00	1.00	4
	4	Top $k = 6$	KNN-35	25	100.00	1.00	1.00	5
	5	Top $k = 5$	KNN-35	25	100.00	1.00	1.00	6
	6	Top $k = 4$	KNN-35	10	100.00	1.00	1.00	7
	7	Top $k = 3$	KNN-35	10	100.00	1.00	1.00	8
	8	Top $k = 2$	KNN-35	10	100.00	1.00	1.00	9
	9	Top $k = 20$	KNN-35	25	100.00	1.00	1.00	10
	10	Top $k = 15$	KNN-35	25	100.00	1.00	1.00	11
	11	Top $k = 10$	KNN-35	25	100.00	1.00	1.00	12

Table 7.46: Overview of the best results about the performance of detecting knowledge-intensive tasks (Tasks 5-7) by stratified 10-fold cross-validation for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

single features was between 3 and 383 attributes except for the *used res. metadata* feature which had it's best classifier run with 1000 attributes.

Top k Features: All the *Top k* feature combinations achieved 100% accuracy with the smallest number of attributes with the KNN-35 learner. The range of the numbers of attributes for the best runs of the *Top k* performing features was between 10 and 75 attributes.

Comparison with existing approaches: The evaluation of the existing approaches showed that the Dyonipos and the *TaskPredictor 1* approaches also achieved 100% accuracy. Whereas the *TaskPredictor 1* approach only achieved 100% with the Naïve Bayes learner the feature combinations $\mathcal{AC}$, $\mathcal{CW}$ and $\mathcal{ACW}$ of the Dyonipos approach managed it with the KNN, NB and the SVM-lin learners. The *SWISH* approach performed worse with 96.67% accuracy and the J48 decision tree learner. A detailed overview of the feature and classifier performances of the existing approaches is given in Table 7.47.

Set	R_G	f	l	g	a	p	r
Dyonipos	2	$\mathcal{CW}$	SVM-$C2^5$	100	100.00	1.00	1.00
	3	$\mathcal{W}$	KNN-35	25	100.00	1.00	1.00
	4	$\mathcal{ACW}$	SVM-$C2^{10}$	125	100.00	1.00	1.00
	5	$\mathcal{AW}$	KNN-35	25	100.00	1.00	1.00
	6	$\mathcal{AC}$	SVM-$C2^{10}$	10	100.00	1.00	1.00
	7	$\mathcal{A}$	KNN-35	24	100.00	1.00	1.00
	9	$\mathcal{C}$	SVM-$C2^{10}$	75	79.33	0.88	0.80
SWISH	8		J48	175	96.67	0.98	0.97
TaskPredictor 1	1		NB	50	100.00	1.00	1.00

Table 7.47: Overview of the best results about the performance of detecting knowledge-intensive tasks (Tasks 5-7) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Concluding Remarks: The evaluations of the task detection performance of the knowledge-intensive tasks (*Task 5-7*) showed that an accuracy of 100% was reached on this task dataset UICO feature combinations as well as feature combinations of the Dyonipos and *TaskPredictor 1* approaches managed to identify all task instances correctly. The high accuracy values achieved for this three class classification problem seemed to origin from the nature of the classes. The classes corresponded to the knowledge-intensive tasks. Since these tasks were very different, the resulting classification problem was easy to solve for the classification models based on various feature configurations.

7.5.5 Research Question: Can the type of task be automatically detected when distinguishing routine and knowledge-intensive tasks?

The goal of the following evaluations was to answer the question *"Can the type of task be automatically detected when distinguishing routine and knowledge-intensive tasks?"*. The datasets on which this question was investigated contained 134 tasks from 10 users. Among these 134 tasks there were 76 *routine tasks* (*Task 1-4*) and 58 *knowledge-intensive tasks* (*Task 5-7*). The distribution of the task instances in respect to the task type (routine and knowledge-intensive tasks) is shown in Table 7.48.

An overview of all results about the performance of detecting if a task instance was a routine task or a knowledge-intensive task is given in Table 7.49. These results were achieved by applying stratified 10-fold cross-validation on the routine and the knowledge-intensive tasks dataset.

Classes	**Class Instances**
Routine Tasks (Task 1-4)	76
Knowledge-Intensive Tasks (Task 5-7)	58
Dataset CV	**134**

Table 7.48: This table shows the distribution of the stratified 10-fold cross-validation training/test instances for the different task classes routine tasks (Task 1-4) and knowledge-intensive tasks (Task 5-7) which were recorded on a laboratory computer

Feature Categories: The combination of all 50 features with 300 attributes and the *application category* with 75 attributes achieved 100% accuracy with the Naïve Bayes algorithm in detecting whether a task instance was a routine or a knowledge-intensive task. Very close behind were the *action category*, the *resource category*, the *ontology structure category* and the *switching sequence category* with an accuracy of 99.29%, 97.80%, 97.69% and 96.43% respectively. The worst feature category was the *content category* with 82.20% accuracy (l=KNN-10, g=75, p=0.81, r=0.81).

Single Features: The best 20 performing single features achieved an accuracy higher than 90%. The best one was the *window title* feature with 99.29% accuracy with 50 attributes (l=NB, p=0.99, r=0.99). Only 0.83% worse were the *used res. metadata* feature (l=KNN-35, g=200) and the *acc. obj. name* feature (l=NB, g=100) with 0.99 micro precision and 0.99 micro recall. None of the single performing features scored 100% accuracy.

Top k Features: All the *Top k* feature combinations obtained 100% accuracy whereas the number of utilized attributes ranged from 75 to 1000 attributes. The Naïve Bayes and the KNN-35 algorithms were the most accurate.

Comparison with existing approaches: The best four Dyonipos feature combinations $\mathcal{CW}$ (l=NB, g=50), $\mathcal{W}$ (l=KNN-35, g=25), $\mathcal{ACW}$ (l=KNN-35, g=50), $\mathcal{ACW}$ (g=NB,

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	NB	300	100.00	1.00	1.00	1
	2	Application Cat.	NB	75	100.00	1.00	1.00	12
	3	Action Cat.	NB	750	99.29	0.99	0.99	14
	4	Resource Cat.	KNN-1	150	97.80	0.98	0.98	18
	5	Ontology Str. Cat.	KNN-35	3	97.69	0.97	0.97	20
	6	Switching Seq. Cat.	NB	300	96.43	0.96	0.96	21
	7	Content Cat.	KNN-10	75	82.20	0.81	0.81	46
Single Feat.	1	window title	NB	50	99.29	0.99	0.99	13
	2	used res. metadata	KNN-35	200	98.46	0.98	0.98	16
	3	acc. obj. name	NB	100	98.46	0.98	0.98	17
	4	datatype properties	KNN-35	25	97.80	0.98	0.98	19
	5	res. interact.	NB	125	96.26	0.96	0.96	22
	6	res. types interact.	J48	3	94.89	0.94	0.94	23
	7	used res. interact.	KNN-10	25	94.84	0.94	0.94	24
	8	acc. obj. value	KNN-1	25	93.46	0.94	0.94	25
	9	applications interact.	J48	50	93.35	0.93	0.93	26
	10	concept instances	KNN-35	5	93.24	0.93	0.93	27
	11	E type switch seq.	NB	42	92.80	0.92	0.92	28
	12	application name	SVM-$C2^{-5}$	24	92.75	0.92	0.92	29
	13	app. switch seq.	SVM-$C2^{10}$	97	91.98	0.92	0.92	30
	14	used resources	NB	616	91.92	0.93	0.93	31
	15	EB duration	SVM-$C2^{5}$	9	91.21	0.90	0.90	32
Top k Feat.	1	Top $k = 9$	NB	200	100.00	1.00	1.00	2
	2	Top $k = 8$	KNN-35	75	100.00	1.00	1.00	3
	3	Top $k = 7$	KNN-35	75	100.00	1.00	1.00	4
	4	Top $k = 6$	KNN-35	75	100.00	1.00	1.00	5
	5	Top $k = 5$	KNN-35	75	100.00	1.00	1.00	6
	6	Top $k = 4$	KNN-35	75	100.00	1.00	1.00	7
	7	Top $k = 3$	NB	750	100.00	1.00	1.00	8
	8	Top $k = 20$	NB	250	100.00	1.00	1.00	9
	9	Top $k = 15$	NB	250	100.00	1.00	1.00	10
	10	Top $k = 10$	NB	250	100.00	1.00	1.00	11
	11	Top $k = 2$	NB	1000	99.23	0.99	0.99	15

Table 7.49: Overview of the best results about the performance of detecting routine or knowledge-intensive tasks for each feature category, for all feature categories combined, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

g=50) and $\mathcal{AW}$ (g=NB, g=50) performed with 100% accuracy. The *SWISH* and the *TaskPredictor 1* approach were close behind the highest accuracy values of the UICO and Dyonipos feature combinations with only 1.48% and 0.71%. The number of attributes required for the 100% accuracy of the Dyonipos feature combinations ranged between 25 and 50 attributes.

Set	R_G	f	l	g	a	p	r
	1	$\mathcal{CW}$	NB	50	100.00	1.00	1.00
	2	$\mathcal{W}$	KNN-35	25	100.00	1.00	1.00
	3	$\mathcal{ACW}$	KNN-35	50	100.00	1.00	1.00
Dyonipos	4	$\mathcal{AW}$	KNN-35	50	100.00	1.00	1.00
	7	$\mathcal{AC}$	J48	1137	91.87	0.91	0.91
	8	$\mathcal{A}$	J48	24	91.04	0.90	0.90
	9	$\mathcal{C}$	SVM-$C2^{-1}$	125	74.84	0.71	0.71
SWISH	6		NB	75	98.52	0.98	0.98
TaskPredictor 1	5		NB	300	99.29	0.99	0.99

Table 7.50: Overview of the best results about the performance of detecting routine or knowledge-intensive tasks for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Concluding Remarks: The evaluations showed that routine task instances were distinguished from knowledge-intensive task instances with an accuracy of 100% on this dataset. All the approaches performed really well on this two-class classification problem. The accuracy values of the best UICO and Dyonipos feature combinations were 100% whereas the ones of the *SWISH* and the *TaskPredictor 1* approaches were 98.52% and 99.29% respectively.

Why were accuracy values of 100% possible? The classification problem was binary with the classes (i) routine and (ii) knowledge-intensive tasks. The `window title` feature of the UICO approach and the `window title` feature of the Dyonipos approach showed an accuracy of 99.29% with 50 attributes and 100% with 25 attributes respectively. All the routine tasks included to log in the online university's information management system called *TUGonline*. None of the knowledge-intensive tasks required such a login step. The used 25 attributes of the Dyonipos `window title` feature are listed in Table 7.51. It was not surprising that the two attributes with the highest discriminative power include the term `‘‘Anmeldung’’`[2] and `‘‘TUGonline’’`. In Table 7.51 the terms that were specific to the TUGonline system are marked with a small "x". All attributes listed in Table 7.51 except the attribute containing "http" are also part of the 50 attributes of the `window title` feature of the UICO which resulted in an accuracy of 99.29%. It seemed that the tasks involving the *TUGonline* system were the reason for the high accuracy

[2] "Anmeldung" is a German term and translates to "Login" in English.

values for this binary classification problem.

Rank	TUGonline Term	Attribute	Rank	TUGonline Term	Attribute
1	x	Anmeldung	14	x	tugraz
2	x	TUGonline	15	x	Termine
3		Google	16		Berlin
4		Microsoft	17	x	Bibliothek
5		Präsentation	18	x	Detailansicht
6		PowerPoint	19		Internet
7	x	Lehrveranstaltung	20		Explorer
8		Suche	21		Einfache
9	x	Visitenkarte	22	x	Katalog
10	x	Gruppenauswahl	23		Java
11		java	24		files
12		Hotels	25	x	Ergebnisliste
13		http			

Table 7.51: Dyonipos best 25 discriminating window title attributes. A small "x" in the "TUGonline Term" columns indicates that this term is highly specific to the university's information management system. The ranks were computed by the information gain attribute selection algorithm.

7.5.6 Finding the Best Features/Feature Categories

Feature Dominance Matrix: Table 7.52 displays the results of the dominance matrix evaluations for the features and feature combinations as explained in Section 7.3.5.1. It shows that the combination of all 50 features (*All Categories*), *application category* and the *window title* feature outperformed the other features and feature combinations most often. Although the *acc. obj. name* feature, the *action category*, the *used res. metadata* feature and the *resource category* were very close behind the well-known *window title* feature with a difference of 7, 10, 11 and 12 respectively. Among the top 15 features and feature combinations were two times an accessibility object features. These were the *acc. obj. name* feature and *acc. obj. value* feature. Two features of the *ontology structure category*, the *datatype properties* feature and the *concept instances* feature, as well as the *ontology structure category* itself were one of the top 15. In the top 15 list all feature categories were present.

Paired T-Tests: The statistical significance tests $T - Test^f$ and $T - Test^f_{rt}$ as described in Section 7.3.5.1 are summarized in Table 7.53 and in Table 7.54 respectively. The $T - Test^f$ showed that the best seven ranked features and feature combinations were not statistically significantly better in comparison to each other on a $p < 0.05$ significance level. The only exception was the combination of all 50 features (*All Categories*) which was statistically significantly better than the *resource category* on a $p < 0.05$ significance level. After applying a rank transforming a higher

	All Categories	Application Cat.	window title	acc. obj. name	Action Cat.	used res. metadata	Resource Cat.	Ontology Str. Cat.	datatype properties	res. types interact.	Switching Seq. Cat.	acc. obj. value	res. interact.	concept instances	used res. interact.	Ψ
All Categories	-	4	5	5	4	5	5	5	5	4	4	5	5	5	5	276
Application Cat.	1	-	4	5	4	4	4	5	5	4	4	5	5	5	5	270
window title	0	1	-	3	4	4	5	5	5	4	4	5	5	5	5	263
acc. obj. name	0	0	2	-	3	3	4	4	5	4	4	5	5	5	5	256
Action Cat.	1	1	1	2	-	2	2	3	3	4	5	4	5	5	5	253
used res. metadata	0	1	1	2	3	-	3	4	5	4	4	5	5	4	4	252
Resource Cat.	0	1	0	1	3	2	-	4	4	4	4	5	5	5	5	250
Ontology Str. Cat.	0	0	0	1	2	1	1	-	2	3	4	3	5	4	5	238
datatype properties	0	0	0	0	2	0	1	3	-	4	4	3	5	4	4	237
res. types interact.	1	1	1	1	1	1	1	2	1	-	4	3	3	4	4	236
Switching Seq. Cat.	0	1	1	1	0	1	1	1	1	1	-	2	3	3	3	226
acc. obj. value	0	0	0	0	1	0	0	2	2	2	3	-	3	4	3	219
res. interact.	0	0	0	0	0	0	0	0	0	2	2	2	-	2	3	218
concept instances	0	0	0	0	0	1	0	1	1	1	2	1	3	-	2	216
used res. interact.	0	0	0	0	0	1	0	0	1	1	2	2	1	2	-	215

Table 7.52: Dominance matrix for the top 15 features and feature combinations.

number of statistically significant pairs were found. This does not necessarily mean that the first test was invalid. It means that the tests provided a complementary view that can be used in the decision which features and feature combination to use. Since all the feature and feature combinations are present among the top results of the $T - Test^f$ evaluation in Table 7.53 and of $T - Test^f_{rt}$ evaluation in Table 7.54 suggested that these were the best features and feature combinations with the highest discriminative power for this dataset.

	All Categories	Application Cat.	window title	Resource Cat.	used res. metadata	Action Cat.	Ontology Str. Cat.	acc. obj. name	datatype properties	Switching Seq. Cat.	res. types interact.	res. interact.	used res. interact.	concept instances	applications interact.	Ψ
All Categories	-	~	~	>	~	~	~	▷	>	~	>	>	>	>	>	50
Application Cat.	~	-	~	~	~	~	~	≫	>	~	>	>	>	>	>	49
window title	~	~	-	~	~	~	~	~	▷	~	~	>	>	>	>	45
Resource Cat.	<	~	~	-	~	~	~	~	>	~	~	>	>	>	~	44
used res. metadata	~	~	~	~	-	~	~	~	>	~	~	>	>	>	~	44
Action Cat.	~	~	~	~	~	-	~	~	~	~	~	~	>	>	>	43
Ontology Str. Cat.	~	~	~	~	~	~	-	~	~	~	~	▷	>	~	~	41
acc. obj. name	◁	≪	~	~	~	~	~	-	~	~	~	~	~	>	~	41
datatype properties	<	<	◁	<	<	~	~	~	-	~	~	~	~	~	~	39
Switching Seq. Cat.	~	~	~	~	~	~	~	~	~	-	~	~	~	~	>	38
res. types interact.	<	<	~	~	~	~	~	~	~	~	-	~	~	~	~	37
res. interact.	<	<	<	<	<	~	◁	~	~	~	~	-	~	~	~	37
used res. interact.	<	<	<	<	<	<	<	~	~	~	~	~	-	~	~	37
concept instances	<	<	<	<	<	<	~	<	~	~	~	~	~	-	~	36
applications interact.	<	<	<	~	~	<	~	~	~	<	~	~	~	~	-	36

Table 7.53: Feature significance matrix for the top 15 features and feature combinations without rank transformation ($T - Test^f$).

7.5.7 Finding the Best Learning Algorithms

Classifier Dominance Matrix: Table 7.55 visualizes the results from the classifier dominance matrix computations as described in Section 7.3.5.2. The order of the best performing classifiers was *NB*$\succ_{351}$*J48*$\succ_{100}$*KNN-10*$\succ_{51}$*KNN-35*$\succ_{3}$*KNN-5*$\succ_{14}$*KNN-1*$\succ_{5}$*SVM-lin*, whereas $c_1 \succ_d c_2$

	All Categories	Application Cat.	window title	Action Cat.	Resource Cat.	acc. obj. name	used res. metadata	Ontology Str. Cat.	Switching Seq. Cat.	datatype properties	res. types interact.	res. interact.	applications interact.	concept instances	used res. interact.	Ψ
All Categories	-	~	>	>	▷	▷	▷	▷	>	▷	>	▷	▷	▷	≫	55
Application Cat.	~	-	~	>	~	>	~	▷	>	>	>	▷	≫	>	≫	52
window title	<	~	-	~	>	~	~	≫	~	≫	~	≫	>	▷	▷	48
Action Cat.	<	<	~	-	~	~	~	~	>	~	~	≫	≫	≫	≫	46
Resource Cat.	◁	~	<	~	-	~	∼	>	~	>	~	≫	~	≫	▷	46
acc. obj. name	◁	<	~	~	~	-	~	>	~	>	~	≫	~	≫	▷	46
used res. metadata	◁	~	~	~	∼	~	-	~	~	≫	~	▷	~	>	>	44
Ontology Str. Cat.	◁	◁	≪	~	<	<	~	-	~	~	~	≫	~	>	>	43
Switching Seq. Cat.	<	<	~	<	~	~	~	~	-	~	~	~	≫	~	~	40
datatype properties	◁	<	≪	~	<	<	≪	~	~	-	~	≫	~	~	~	40
res. types interact.	<	<	~	~	~	~	~	~	~	~	-	~	≫	~	~	40
res. interact.	◁	◁	≪	≪	≪	≪	◁	≪	~	≪	~	-	~	~	~	39
applications interact.	◁	≪	<	≪	~	~	~	~	≪	~	≪	~	-	~	~	39
concept instances	◁	<	◁	≪	≪	≪	<	<	~	~	~	~	~	-	∼	38
used res. interact.	≪	≪	◁	≪	◁	◁	<	<	~	~	~	~	~	∼	-	37

Table 7.54: Feature significance matrix for the top 15 features and feature combinations with rank transformation ($T - Test^{f}_{rt}$).

indicates that classifier c_1 performed d times better than c_2 based on Ψ. The *Naïve Bayes* (NB) learner seemed to perform significantly better than the other learning algorithms. This intention was confirmed by the paired t-test results highlighted in Table 7.56 and in Table 7.57 for "without" and "with" rank transformation respectively.

	NB	J48	KNN-10	KNN-35	KNN-5	KNN-1	SVM-lin	Ψ
NB	-	193	206	213	215	215	208	1250
J48	97	-	144	154	155	151	198	899
KNN-10	88	152	-	115	126	129	189	799
KNN-35	77	141	106	-	124	110	190	748
KNN-5	78	139	103	106	-	123	196	745
KNN-1	76	143	102	114	107	-	189	731
SVM-lin	107	117	127	127	120	128	-	726

Table 7.55: Dominance matrix for the classifiers $l \in$ { J48, KNN-1, KNN-10, KNN-35, KNN-5, NB, SVM-lin}

Paired T-Tests: Two complete partial orders of classifiers were computed by using each column of Table 7.56 and Table 7.57 for the different significance levels. The classifiers that showed no significant performance differences were grouped together. The partial order of the results without rank transformation was {*NB*}≫{*J48, KNN-1, KNN-5, KNN-10,KNN-35*}≫{*SVM-lin*}. The one for the results with rank transformation was {*NB*}≫{*J48*}>{*KNN-1, KNN-5, KNN-10, KNN-35*}≫{*SVM-lin*}. The effect that the *J48* decision tree learner was statistically significantly better with a $p < 0.05$ significance level in the second case, might be the result of the smoothing of the differences of the accuracy values after a rank transformation. However, this result went hand in hand with the ranking obtained from the classifier dominance matrix in Table 7.55.

Paired t-tests performed based on the micro f-measures without and with rank transformation resulted in a similar partial order $\{NB\}\gg\{J48, KNN\text{-}1, KNN\text{-}5, KNN\text{-}10, KNN\text{-}35\}\gg\{SVM\text{-}lin\}$.

	NB	J48	KNN-10	KNN-1	KNN-35	KNN-5	SVM-lin	Ψ
NB	-	≫	≫	≫	≫	≫	≫	6
J48	≪	-	∼	>	>	∼	≫	3
KNN-10	≪	∼	-	>	>	∼	≫	3
KNN-1	≪	<	<	-	∼	∼	≫	1
KNN-35	≪	<	<	∼	-	∼	≫	1
KNN-5	≪	∼	∼	∼	∼	-	≫	1
SVM-lin	≪	≪	≪	≪	≪	≪	-	0

Table 7.56: Significance matrix for the classifiers $l \in$ {J48, KNN-1, KNN-10, KNN-35, KNN-5, NB, SVM-lin} without rank transformation.

	NB	J48	KNN-1	KNN-10	KNN-35	KNN-5	SVM-lin	Ψ
NB	-	≫	≫	≫	≫	≫	≫	6
J48	≪	-	>	>	>	>	≫	5
KNN-1	≪	<	-	∼	∼	∼	≫	1
KNN-10	≪	<	∼	-	∼	∼	≫	1
KNN-35	≪	<	∼	∼	-	∼	≫	1
KNN-5	≪	<	∼	∼	∼	-	≫	1
SVM-lin	≪	≪	≪	≪	≪	≪	-	0

Table 7.57: Significance matrix for the classifiers $l \in$ {J48, KNN-1, KNN-10, KNN-35, KNN-5, NB, SVM-lin} with rank transformation.

These results suggested the conclusion that the *Naïve Bayes* and the *J48 decision tree* learners perform better on this dataset as the *k-Nearest Neighbor* (KNN) algorithms and the *linear Support Vector Machines* (SVM).

7.5.8 Concluding Remarks

In Table 7.58 the results of laboratory experiment 2 which were described in this section, are summarized. It shows (i) which research questions were investigated and (ii) the results of automatic task detection of the UICO approach in comparison to the Dyonipos, *SWISH* and *TaskPredictor 1* approaches. One can observe in Table 7.58 that the UICO approach outperformed all the existing approaches on the majority of the investigated research questions. Further remarks about the answers to the research questions are elaborated for personal and standard task in Section 7.5.8.1 and for routine and knowledge-intensive tasks in Section 7.5.8.2. Remarks about the experiment itself are mentioned in Section 7.5.8.3.

Evaluations	**UICO**	**Dyonipos**	***SWISH***	**Task P. 1**
Task Models (7 TM)	94.84%	**95.49%**	88.90%	93.24%
Routine vs. Knowledge-Int. (7 TM)	**100.00%**	**100.00%**	98.52%	99.29%
Routine (4 TM)	**94.64%**	93.39%	93.57%	93.75%
Knowledge-Int. (3 TM)	**100.00%**	**100.00%**	**100.00%**	**100.00%**
Pers. based on Std. Task (7 TM)	**92.42%**	90.91%	78.79%	87.88%
Standard Tasks (7 TM)	**98.57%**	97.14%	95.71%	97.14%
Personal Tasks (7 TM)	**94.05%**	91.19%	87.86%	89.76%

Table 7.58: Overview of the results of the laboratory experiment 2. The task detection performances for the task models (TM) elaborated in this chapter are visualized in this table. The highest achieved accuracy values for each evaluation are marked in bold.

7.5.8.1 Detecting Personal Tasks based on Standard Tasks

The answer to the question if it is possible to detect personal task executions with a learning algorithm trained on standard task execution can be answered with a clear *yes* on this dataset. The results of this experiment were that the UICO approach detected 92.43% of the personal executed tasks. This was a very high accuracy which suggested that a training on standard task executions was sufficient for detecting personal tasks, i.e., the influence of the *specific goal* of a task was not that great as expected (see Section 7.5). The results also suggested that a set of standard tasks could be defined for execution in a training phase by users prior deployment. This would allow an automatic detection of real personal tasks in a productive environment without training the classifier during work.

7.5.8.2 Routine vs. Knowledge-Intensive Tasks

The results of the task detection laboratory experiment 2 made the impression that routine tasks and knowledge task can be distinguished very well. A 100% accuracy in detecting whether a task execution was a routine or a knowledge-intensive task by the UICO and the Dyonipos approach on this dataset underlines this impression. It can of course be argued that the tasks were too different such that an easy classification was possible. However, laboratory experiment 2 (see Section 7.4.6) showed also a high accuracy (94%) in solving this two-class classifiction problem. Nevertheless further experiments with other tasks and users of other domains are recommended to study this effect in more detail.

7.5.8.3 Remarks about the Experiment

The experiment was designed to reduce bias as much as possible. For this randomization of tasks as well as task types were introduced. Since the prototype utilized in the experiment required a not insignificantly large amount of memory, at some points the laboratory computer slowed down a little bit or the prototype had to be restarted. This might had introduced bias in the interaction behavior of the participants and hence in the task execution data collected. One participant had to execute a few tasks again because of a not prototype related software failure. The collection of the data from the other participants went well and was reliable. The participants had only a few questions about the prototype at the beginning of a test session regarding its functionality about creating, starting, suspending and completing a task.

7.5.8.4 Open Questions

The evaluations on the dataset of this laboratory experiment showed a 100% accuracy in detecting knowledge-intensive tasks. Due to the fact that only 3 knowledge-intensive tasks were studied in this experiment the open question if other ones are as well distinguishable among each other as the ones captured in this experiment remains open. This question was investigated in the third laboratory experiment in which a special focus was put on knowledge-intensive tasks (see Section 7.6).

7.6 Laboratory Experiment 3 - Computer Science Students

The experiment described here investigated knowledge-intensive tasks in the domain of *"computer science students at Graz University of Technology"*. The *CommonKADS* knowledge-intensive task classification [Schreiber *et al.*, 1999] was utilized to categorize tasks. Informal interviews showed that students performed knowledge-intensive tasks during their university time. This experiment was conducted to get insights into the capabilities of automatic task detection applied to usage data observed during executions of knowledge-intensive tasks. The tasks of the student's domain were studied in a controlled setting consisting of four laboratory computers. 18 computer science students participated in the experiment. These students were considered as experts of the studied domain. They allowed the observation of their user interaction context during their task executions and made it freely available for the evaluations described in this section.

Following questions were investigated:

- Can the task model of the task instances be automatically detected?
- Can the task models of analytic task instances be automatically detected?
- Can the task models of synthetic task instances be automatically detected?
- Can the analytic and the synthetic knowledge-intensive task models of the task instances be automatically detected?
- Which context features are most discriminative for the studied tasks?
- Which learning algorithm performs best in terms of automatic task detection on the collected dataset?

7.6.1 Experiment Design

The comparison was *within subjects* and the manipulations were achieved by (i) the *type of knowledge-intensive task* (analytic and synthetic) and (ii) the *task* to be executed (8 different tasks).

The experiment was designed in three phases. Phase 1 was the phase before the recording of the user interaction context. Phase 2 was the user interaction context observation phase. This phase was followed by Phase 3, which included the evaluation of the automatic task detection performance. Phase 1, 2 and 3 were the same as in the experiment described in Section 7.5 with one exception in Phase 2. In Phase 2 an additional 10 minutes *"get familiar with the computer system"* time slot was introduced in order to allow the participants to get familiar with the computer setup, the installed applications and resources available. The intention was to further reduce the chance of introducing a bias through an unfamiliar computer desktop environment by including a short training phase.

Manipulation 1: Type of tasks (analytic and synthetic)

The first manipulation was achieved by varying the type (class/categorization) of a knowledge-intensive task according to the *CommonKADS* task classification. *CommonKADS* distinguishes knowledge-intensive tasks into two groups of task types: *analytic tasks* and *synthetic tasks.*

> *"The distinguishing feature between the two groups is the "system" the task operates on. 'System' is an abstract term for the object to which the task is applied. For example, in technical diagnosis the system is the artifact or device being diagnosed; in elevator configuration it is the elevator to be designed. In analytic tasks the system preexists although it is typically not completely 'known'. All analytic tasks take as input some data about the system, and produce some characterization of the system as output. In contrast, for synthetic tasks the system does not yet exist: the purpose of the task is to construct a system description. The input of a synthetic task typically consists of requirements that the system to be constructed should satisfy [Schreiber* et al.*, 1999]."*

Four analytic and four synthetic task executions from each participant were recorded. The order in which the experiment's participants started performing a task of a specific type was randomized.

Manipulation 2: Tasks
The second manipulation resulted from varying the tasks themselves. Eight knowledge intensive tasks were studied. Four tasks from the sub-categories of analytic tasks and four tasks from the sub-categories of synthetic tasks had to be executed by the experiment's participants. The hierarchy of the studied task types is shown bellow within the task classification. A detailed description of the task models and example tasks is given in Section 8.6.

1. *Analytic Task*	2. *Synthetic Task*
Task A1: Classification	Task S1: Design
Task A2: Diagnose	Task S2: Assign
Task A3: Assess	Task S3: Plan
Task A4: Predict	Task S4: Schedule

In the experiment the two task types of the *CommonKADS* classification *monitoring* and *modeling* were excluded because of the following reasons: Monitoring would have required several assessment cycles which would not have been possible because of the experiment's time constraints. Furthermore monitoring is a kind of "long-term task" and hence should be studied separately and not in an approximately 120min laboratory experiment. Modeling involves the modeling of a physical phenomena. In the test domain of computer science students at Graz University of Technology, the modeling of a physical artifact is normally not a common task and hence was excluded.

Example of a task model:

[Task A1] Task Classify (max. 5 min.)

Task Model (Description):
In classification, an object needs to be characterized in terms of the class to which it belongs.

> *Task Instance:*
> The following list contains 10 well-known computer specific terms. Please classify them into the following categorization schema: *hardware* (with the sub-categories input device and non-input device) and *software* (with the sub-categories game, office application and operating system). Your classification schema shall be stored on the computer. Choose a program of your choice and save the result on the computer.
>
> Terms to categorize: Ahead Nero 9.0, Adobe Acrobat 9.0 Professional, Adobe Photoshop CS4, Call of Duty 5 - World at War, Intel Core 2 Duo E8400 (C0), 2x 3.0GHz, Logitech Classic Keyboard 200, Logitech MX 518 Optical Gaming Mouse Refresh, Microsoft Office 2007 Professional, Microsoft Windows Vista Business 32Bit, Seagate Barracuda 7200.11 1500GB, SATA II

Before the participants of the experiment started the execution of the tasks they were asked to read through the task descriptions and to confirm that they were clear to them. Since they were asked to execute the task on a laboratory computer, they were given a time to get familiar with the capabilities of the computer and the installed programs. The instruction given to the participants was:

> *"At the begin of the experiment you have 10 min. to get familiar with the configuration of the PC and the resources available. If you think you are ready earlier you can start with your first task."*

The participants were especially instructed to utilize the given computer as much as possible and to not use any other resources except the computer. The instructions given to the participants are shown bellow:

> *"Interact as much as possible with the PC and do everything you need for performing the task on the PC (e.g, note taking, drawings, calculations etc.). Do not take any notes on paper. Your interactions with the PC are important for our research. You are free to use any resources available on the PC (applications, tools, internet, documents, files, folders etc.). Thank you for your participation!"*

7.6.2 Research Question: Can the task model of a task instances be automatically detected?

The goal of this evaluation was to answer the question *"Can the task model of the task instances be automatically detected?"*. The dataset on which this question was investigated contained 132 tasks from 18 users. The distribution of the task instances in respect to the classes and the knowledge-intensive task type (analytic and synthetic task) is shown in Table 7.59. Analytic and synthetic tasks were not distinguished in this evaluation but are listed in Table 7.59 to provide the reader with a complete picture about the task distribution in the dataset. An overview of all results about the performance of detecting the tasks (*Tasks A1-A4* and *Tasks S1-S4*) is given in Table 7.6.2. The results were achieved by applying stratified 10-fold cross-validation on the training instances. A training instance was built for each task instance independently.

Type	**Task Class**	**Task Instances**	**Sum**
Analytic Tasks	Task A1	17	67
	Task A2	15	
	Task A3	19	
	Task A4	16	
Synthetic Tasks	Task S1	18	65
	Task S2	16	
	Task S3	16	
	Task S4	15	
Dataset CV			**132**

Table 7.59: This table shows the distribution of the training/test instances for the different analytic and synthetic task classes ranging from Task A1 to A4 and from Task S1 to S4 which were recorded on the laboratory computers.

Feature Categories: The feature category which achieved the highest accuracy value was the combination of all 50 features of all categories (l=J48, a=85.00%, g=10000, p=0.97, r=0.86). This category only achieved rank 4 in the global ranking. About 5% less accurate was with the same algorithm but with only 500 attributes the *application category* (l=J48, a=80.38%, p=0.97, r=0.81). The *resource category* obtained an accuracy of 62.86% with 1500 attributes (l=J48, p=0.92, r=0.64) which was 22.14% worse than the best feature category. The number of attributes of the best runs were between 75 and 10000 attributes.

Single Features: The best performing single feature with 80.27% accuracy was the *acc. obj. name* feature (l=J48, g=175, p=0.96, r=0.82). Far behind with a 16.7% less high accuracy value was the *window title* feature with the same algorithm (l=J48, a=63.57%, p=0.91, r=0.64). The *used res. metadata* feature achieved the third rank of the best performing single features with an accuracy of 61.43% on 2000 attributes with the same algorithm as the best two performing single features. The range of the numbers of attributes for the best classifier runs of the top 15 single

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	J48	10000	85.00	0.97	0.86	4
	2	Application Cat.	J48	500	80.38	0.96	0.81	12
	3	Resource Cat.	J48	1500	62.86	0.92	0.64	16
	4	Action Cat.	NB	250	60.71	0.90	0.60	18
	5	Content Cat.	J48	250	59.78	0.91	0.60	20
	6	Switching Seq. Cat.	NB	1173	54.51	0.88	0.52	22
	7	Ontology Str. Cat.	KNN-10	75	53.02	0.88	0.54	23
Single Feat.	1	acc. obj. name	J48	175	80.27	0.96	0.82	13
	2	window title	J48	250	63.57	0.91	0.64	15
	3	used res. metadata	J48	2000	61.43	0.91	0.60	17
	4	acc. obj. value	J48	100	60.55	0.92	0.62	19
	5	applications interact.	J48	50	54.67	0.89	0.58	21
	6	concept instances	KNN-1	10	52.09	0.86	0.51	24
	7	content in focus	J48	150	48.52	0.85	0.49	25
	8	content of EB	NB	712	47.91	0.86	0.51	26
	9	datatype properties	J48	221	47.69	0.85	0.48	27
	10	used resources	NB	150	46.21	0.83	0.45	28
	11	app. switch seq.	NB	182	46.04	0.84	0.48	29
	12	res. interact.	NB	150	45.60	0.83	0.42	30
	13	res. types interact.	KNN-10	5	45.49	0.85	0.46	31
	14	acc. obj. role	NB	37	45.38	0.84	0.46	32
	15	used res. interact.	J48	125	44.18	0.83	0.43	33
Top k Feat.	1	Top $k = 6$	J48	1500	86.43	0.98	0.86	1
	2	Top $k = 20$	J48	10000	85.66	0.97	0.84	2
	3	Top $k = 7$	J48	2000	85.49	0.98	0.86	3
	4	Top $k = 8$	J48	2000	84.89	0.97	0.85	5
	5	Top $k = 10$	J48	10000	84.89	0.97	0.84	6
	6	Top $k = 4$	J48	2000	84.78	0.97	0.86	7
	7	Top $k = 5$	J48	7500	84.07	0.97	0.87	8
	8	Top $k = 9$	J48	7500	83.41	0.97	0.86	9
	9	Top $k = 3$	J48	2500	82.64	0.97	0.82	10
	10	Top $k = 15$	J48	7500	82.58	0.97	0.83	11
	11	Top $k = 2$	J48	150	80.27	0.96	0.79	14

Table 7.60: Overview of the best results about the performance of detecting knowledge-intensive tasks (Tasks A1-A4 and S1-S4) by stratified 10-fold cross-validation for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

performing features was between 3 and 250 attributes except for the *used res. metadata* feature and the *content of EB* feature which had their best classifier runs with 2000 and 712 attributes respectively. In comparison with the *Top k* feature combinations all *Top k* feature combinations except the worst *Top k* feature combination which was the *Top* $k = 2$ outperformed all the single features.

Top ***k*** **Features:** The *Top k* single feature combinations achieved accuracy values ranging from 80.27% to 86.43%. The highest accuracy resulted from the *Top* $k = 6$ single feature combination (l=J48, a=86.43%, p=0.98, r=0.86) which obtained also the highest accuracy value among all feature categories and single features. All the *Top k* feature combinations had their best classifier runs with a high number of attributes ranging from 1500 to 10000 attributes. The only exception was the *Top* $k = 2$ which had its best classifier run with 150 attributes and the J48 learner. The *Top* $k = 2$ obtained an 6.26% worse accuracy than the best performing *Top k* feature combination. All the *Top k* feature combinations achieved their highest accuracy with the J48 classifier.

Set	R_G	f	l	g	a	p	r
Dyonipos	1	$\mathcal{CW}$	SVM-$C2^{-2}$	500	66.04	0.93	0.66
	2	$\mathcal{ACW}$	KNN-5	175	64.23	0.92	0.64
	4	$\mathcal{W}$	J48	337	59.78	0.91	0.61
	5	$\mathcal{AW}$	J48	250	58.57	0.90	0.59
	6	$\mathcal{AC}$	KNN-35	150	57.42	0.90	0.57
	8	$\mathcal{A}$	KNN-1	10	45.60	0.84	0.44
	9	$\mathcal{C}$	NB	750	43.85	0.83	0.45
SWISH	7		J48	414	56.76	0.91	0.61
TaskPredictor 1	3		J48	150	63.85	0.92	0.64

Table 7.61: Overview of the best results about the performance of detecting knowledge-intensive tasks (Tasks A1-A4 and S1-S4) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Comparison with existing approaches: A detailed comparison of the feature and classifier performance evaluation of the existing approaches is given in Table 7.61. This comparison shows that the best overall tested combinations of the UICO features, the *Top* $k = 6$ single feature combination (l=J48, a=86.43%, p=0.98, r=0.86), outperformed all the existing approaches. The *SWISH* , the *TaskPredictor 1* and the best Dyonipos feature combination $\mathcal{CW}$ were outperformed by 29.67%, 22.58% and 20.39% respectively. The *SWISH* approach had its best run at 414 attributes with the J48 algorithm (a=56.76%,p=0.91, r=0.61). The *TaskPredictor 1* approach also performed best with the J48 algorithm but required only 150 attributes. Among the existing approaches the Dyonipos one with the $\mathcal{CW}$ resulted in the highest accuracy (l=SVM,

a=66.04%,p=0.93, r=0.66).

All the existing approaches had their best runs with the J48 algorithm which was similar to all the best runs of the *Top k* single feature combinations. The range of attributes of the best runs of the existing approaches ranged from 10 to 500 attributes which was rather low in comparison with the best ones of the UICO *Top k* single feature combinations.

Concluding Remarks: All except one of the *Top k* single feature combinations outperformed the feature categories and the single features. The highest accuracy was achieved by the *Top* $k = 6$ single feature combination (l=J48, a=86.43%, p=0.98, r=0.86). This best UICO feature combination outperformed significantly the *SWISH* , the *TaskPredictor 1* and the best Dyonipos feature combination $\mathcal{CW}$ by 29.67%, 22.58% and 20.39% respectively. These results showed that the task model of knowledge-intensive tasks can be detected with an accuracy of 86.43% on the gathered dataset from a large scale experiment with 18 participants. In other words, it seemed to be possible to detect unstructured, knowledge-intensive and "free will how to achieve the requested goal" tasks even with a high accuracy. Although these results were promising, experiments with other tasks and users of the same domain and in other ones are necessary to further investigate the generalizability of these findings.

7.6.3 Research Question: Can the task models of analytic task instances be automatically detected?

The goal of the following evaluations was to answer the question *"Can the task models of analytic task instances be automatically detected?"*. The dataset on which this question was investigated contained 67 tasks from 18 users. These 67 tasks were almost equally distributed among the four analytic knowledge-intensive task types (*Task A1-A4*) as visualized in in Table 7.62.

An overview of all results about the task detection performance is given in Table 7.63. These results were achieved by applying stratified 10-fold cross-validation. For each task instance a training instance was built.

Set	Task A1	Task A2	Task A3	Task A4	Sum
Dataset CV	17	15	19	16	**67**

Table 7.62: This table shows the distribution of the training/test instances for stratified 10-fold cross-validation for the different analytic task classes ranging from Task A1 to Task A4 which were recorded on the laboratory computers.

Feature Categories: The feature category which achieved the highest accuracy value of 95.71% was the *resource category* with the J48 learner (g=75, p=0.98, r=0.96). A 1.42% less accuracy was obtained by the *application category* with the same algorithm (l=J48, a=94.29%, g=200, p=0.98, r=0.95). The combination of all 50 features of all categories resulted in a 94.05% accuracy with 750 attributes and the KNN-35 algorithm (p=0.98, r=0.94). The range of the number of attributes of the best classifier runs was between 75 and 385 attributes except for the combination of all 50 features which had its best one with 750 attributes. The accuracy values resulting from the feature categories' evaluations were between 76.67% and 95.71%. In comparison with the performance of the single features only the *resource category* could match the accuracy of the best single feature.

Single Features: The best performing single feature was the *used res. metadata* feature (l=J48, a=95.71%, g=125, p=0.98, r=0.95). The *acc. obj. value* feature was the one with the second highest accuracy (l=J48, a=92.38%, g=25, p=0.97, r=0.92) and only 3.33% worse than the best single feature's accuracy. The third rank of the best performing single features went to the *acc. obj. name* feature with an accuracy of 91.43% on 75 attributes and the Naïve Bayes learner (p=0.92, r=0.92). The range of the numbers of attributes for the best classifier runs of the single performing features was between 3 and 214 attributes. The accuracy values for the best 15 single features ranged from 76.43% to 95.71%. In comparison with the *Top k* feature combinations only the best single feature outperformed the worst three *Top k* feature combinations.

Top *k* Features: The *Top k* best single feature combinations achieved accuracy values ranging from 94.05% to 97.14%. The highest accuracy value resulted from the *Top* $k = 7$ (l=J48, a=97.14%, g=125, p=0.99, r=0.98) and the *Top* $k = 15$ single feature combinations (l=J48, a=97.14%, g=175, p=0.99, r=0.98). These two also outperformed all the other feature categories and single features in terms of accuracy. The *Top* $k = 8$ had its best run on 175 attributes with an accuracy of 96.90% which was an only 0.24% worse accuracy than the best two ones. The range of the numbers of attributes for the best runs of the *Top k* best performing single features was between 75 and 200 except for the *Top* $k = 5$ which had its best classifier run with 1000 attributes (l=J48, a=95.71%, p=0.98, r=0.96). The *Top k* with $k = \{2, 3, 5, 6, 9\}$ performed equally well in terms of accuracy with 95.71%. All the *Top k* achieved their highest accuracy with the J48 learner.

Comparison with existing approaches: A comparison with existing approaches showed that the best overall tested combinations of the UICO features which were the *Top* $k = \{7, 15\}$ best performing single feature combinations achieved a slightly higher accuracy than all the existing approaches. The *SWISH* (l=J48, a=92.86%, g=300, p=0.97, r=0.94), the *TaskPredictor 1* (l=J48, a=95.48%, g=10, p=0.98, r=0.95) and the best Dyonipos feature combination $\mathcal{ACW}$ (l=KNN-35, a=96.90%, g=150, p=0.99, r=0.98) were outperformed by 4.28%, 1.66% and 0.24% accuracy respectively. The existing approaches' accuracy values ranged from 72.38% to 96.90%. The number of attributes of the best runs of the algorithms were between 5 and 500 attributes. A detailed comparison of the feature and classifier performances is given in Table 7.64.

Concluding Remarks: The evaluations of the task classification performance of analytic knowledge-intensive tasks (*Task A1-A4*) on this task dataset showed that an accuracy of about 97% can be reached when utilizing the UICO's *Top* $k = \{7, 15\}$ best performing single feature combinations. The accuracy values achieved by the existing approaches were only 0.24% to 4.28% worse in comparison with the best UICO feature combinations.

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	Resource Cat.	J48	75	95.71	0.98	0.96	9
	2	Application Cat.	J48	200	94.29	0.98	0.95	13
	3	All Categories	KNN-35	750	94.05	0.98	0.94	14
	4	Action Cat.	NB	125	92.38	0.97	0.92	17
	5	Switching Seq. Cat.	NB	100	90.00	0.96	0.91	20
	6	Ontology Str. Cat.	J48	385	88.33	0.96	0.90	21
	7	Content Cat.	KNN-5	250	76.67	0.90	0.75	32
Single Feat.	1	used res. metadata	J48	125	95.71	0.98	0.95	10
	2	acc. obj. value	J48	25	92.38	0.97	0.92	16
	3	acc. obj. name	NB	75	91.43	0.97	0.92	18
	4	window title	J48	214	91.19	0.97	0.92	19
	5	concept instances	J48	104	88.33	0.95	0.88	22
	6	datatype properties	J48	175	85.48	0.94	0.88	23
	7	res. types interact.	KNN-35	10	85.24	0.94	0.85	24
	8	applications interact.	J48	50	82.38	0.93	0.82	25
	9	acc. obj. role	KNN-5	10	81.67	0.93	0.84	26
	10	res. interact.	NB	25	80.71	0.91	0.79	27
	11	used res. interact.	KNN-5	50	79.29	0.92	0.79	28
	12	used resources	NB	75	78.81	0.91	0.76	29
	13	app. switch seq.	J48	3	77.14	0.90	0.78	30
	14	E&EB res. switch seq.	KNN-35	25	76.67	0.90	0.76	31
	15	EB res. interact.	KNN-35	50	76.43	0.90	0.75	33
Top k Feat.	1	Top $k = 7$	J48	125	97.14	0.99	0.98	1
	2	Top $k = 15$	J48	175	97.14	0.99	0.98	2
	3	Top $k = 8$	J48	175	96.90	0.99	0.98	3
	4	Top $k = 9$	J48	125	95.71	0.98	0.96	4
	5	Top $k = 6$	J48	200	95.71	0.98	0.96	5
	6	Top $k = 5$	J48	1000	95.71	0.98	0.96	6
	7	Top $k = 3$	J48	75	95.71	0.98	0.96	7
	8	Top $k = 2$	J48	100	95.71	0.98	0.96	8
	9	Top $k = 10$	J48	175	95.48	0.98	0.95	11
	10	Top $k = 4$	J48	100	94.29	0.98	0.95	12
	11	Top $k = 20$	J48	150	94.05	0.98	0.94	15

Table 7.63: Overview of the best results about the performance of detecting the sub-types of analytic tasks (Tasks A1-A4) by stratified 10-fold cross-validation for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

Set	R_G	f	l	g	a	p	r
Dyonipos	1	$\mathcal{ACW}$	KNN-35	150	96.90	0.99	0.98
	2	$\mathcal{CW}$	KNN-35	175	95.71	0.98	0.95
	5	$\mathcal{AW}$	SVM-$C2^{10}$	10	92.86	0.97	0.94
	6	$\mathcal{W}$	SVM-$C2^{10}$	10	92.62	0.97	0.94
	7	$\mathcal{AC}$	J48	500	79.52	0.92	0.81
	8	$\mathcal{A}$	NB	5	75.00	0.90	0.78
	9	$\mathcal{C}$	KNN-1	125	72.38	0.87	0.74
SWISH	4		J48	300	92.86	0.97	0.94
TaskPredictor 1	3		J48	10	95.48	0.98	0.95

Table 7.64: Overview of the best results about the performance of detecting the sub-types of analytic tasks (Tasks 1-4) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

7.6.4 Research Question: Can the task models of synthetic task instances be automatically detected?

The goal of the following evaluations was to answer the question *"Can the task models of synthetic task instances be automatically detected?"*. The dataset on which this question was investigated contained 65 tasks from 18 users. These 65 tasks were almost equally distributed among the four synthetic knowledge-intensive task types (*Task S1-S4*) as visualized in in Table 7.65. An overview of all results achieved by applying stratified 10-fold cross-validation on dataset is given in Table 7.66. For each task instance a training instance was built.

Set	**Task S1**	**Task S2**	**Task S3**	**Task S4**	**Sum**
Dataset CV	18	16	16	15	**65**

Table 7.65: This table shows the distribution of the training/test instances for the different task classes ranging from Task 1 to Task 7 which were recorded on the laboratory computers.

Feature Categories: The feature category which achieved the highest accuracy values was the combination of all 50 features with 93.23% and the J48 algorithm (g=150, p=0.92, r=0.82). This feature combination resulted in the global rank $R_G = 5$. Only 0.24% less accurate was the *application category* with 91.76% with the same algorithm on only 5 attributes (p=0.92, r=0.79). The *content category* accomplished 63.57% accuracy (l=KNN-35, g=50, p=0.82, r=0.65) and hence was about 18.57% worse than the best feature category.

The accuracy values of the feature categories were between 54.29% and 82.14%. The number of attributes of the best classifier runs of the algorithms of the feature category ranged from 5 to

175 except for the *resource category* which had its best run with 2000 attributes. In comparison with the performance of the single features the best two feature categories, the combination of all features of all categories as well as the *application category*, outperformed all single features in terms of accuracy.

Single Features: The best performing single feature was the *acc. obj. name* feature with an accuracy of 81.67% on only 3 attributes (l=KNN-5, p=0.92, r=0.82). The *datatype properties* feature was 23.10% worse in terms of accuracy (l=KNN-5, a=58.57%, g=25, p=0.92, r=0.82). The third best single performing feature was the *content in focus* feature with 57.62% accuracy (l=J48, g=50, p=0.78, r=0.55). The range of the numbers of attributes for the best classifier runs of the best 15 single performing features was between 1 and 300 attributes with accuracy values between 46.90% and 81.67%. Only the best single performing feature, the *acc. obj. name* feature, outperformed the worst two *Top k* feature combinations.

Top *k* Features: The *Top k* single feature combinations achieved accuracy values ranging from 80.24% to 85.24% and hence only differ among each other by 5%. The highest accuracy value resulted from the *Top* $k = 20$ (l=J48, a=85.24%, g=7500, p=0.93, r=0.82) and was hence also the best among all feature categories and single features in terms of accuracy. The *Top* $k = 15$ feature combination obtained the global rank $R_G = 2$ with 83.57% accuracy on 5000 attributes with the same algorithm, the same micro precision as well as the same micro recall values. The third rank went to the *Top* $k = 10$ feature combination with an accuracy of 83.33% (l=J48, g=6807, p=0.94, r=0.84).

The best four *Top k* feature combinations, *Top* $k = \{9, 10, 15, 20\}$, outperformed all the other feature categories and single features in terms of accuracy but required many more attributes. The range of the number of attributes for the best classifier runs of the algorithms was generally high for this dataset and ranged between 1997 and 6807 attributes. The only exceptions to this high number of required attributes were the *Top* $k = 2$ and the *Top* $k = 3$ which had their best runs with 10 and 5 attributes respectively. The decrease of accuracy when focusing on a low number of attributes in comparison to the best *Top k* one was between 3.34% and 5%.

Comparison with existing approaches: The best five Dyonipos feature combinations achieved a higher accuracy value than the *SWISH* and *TaskPredictor 1* approaches. The Dyonipos $\mathcal{ACW}$ obtained the highest accuracy with 73.81% (l=NB, g=50, p=0.88, r=0.72). The *SWISH* and the *TaskPredictor 1* approach accomplished 55.48% accuracy. The range of the number of attributes of the best runs of the algorithms of the existing approaches was between 10 and 250 which was a lot less than the best UICO feature combination which required 7500 attributes. In comparison with the best UICO feature combination, the *Top* $k = 20$ (l=J48, a=85.24%, g=7500, p=0.93, r=0.82), the best Dyonipos feature combination $\mathcal{ACW}$, the *SWISH* approach and the *TaskPredictor 1* approach were outperformed by 11.43%, 29.76% and 29.76% respectively.

Concluding Remarks: These evaluation results showed that synthetic task instances could be distinguished from each other with an accuracy of 85.24% and that the UICO's

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	J48	150	82.14	0.92	0.82	5
	2	Application Cat.	J48	5	81.90	0.92	0.79	10
	3	Content Cat.	KNN-35	50	63.57	0.82	0.65	15
	4	Action Cat.	NB	175	56.19	0.76	0.58	20
	5	Resource Cat.	J48	2000	55.71	0.74	0.51	21
	6	Switching Seq. Cat.	NB	100	54.76	0.76	0.58	23
	7	Ontology Str. Cat.	NB	25	54.29	0.74	0.54	25
Single Feat.	1	acc. obj. name	KNN-5	3	81.67	0.92	0.82	12
	2	datatype properties	KNN-5	25	58.57	0.78	0.55	16
	3	content in focus	J48	50	57.62	0.78	0.58	17
	4	window title	J48	153	57.62	0.74	0.58	18
	5	used res. metadata	J48	300	57.14	0.77	0.55	19
	6	applications interact.	NB	64	55.24	0.75	0.55	22
	7	acc. obj. value	J48	75	54.52	0.78	0.56	24
	8	res. interact.	NB	168	52.62	0.75	0.52	26
	9	res. types interact.	NB	29	52.14	0.76	0.51	27
	10	user input	KNN-1	5	51.19	0.74	0.52	28
	11	app. switch seq.	SVM-$C2^{10}$	50	48.10	0.71	0.50	29
	12	application name	KNN-35	10	47.38	0.70	0.48	30
	13	content of EB	NB	5	47.38	0.68	0.44	31
	14	task duration	SVM-$C2^{-1}$	1	47.14	0.69	0.49	32
	15	mean time between EBs	SVM-$C2^{-5}$	1	46.90	0.69	0.48	33
Top k Feat.	1	Top $k = 20$	J48	7500	85.24	0.93	0.82	1
	2	Top $k = 15$	J48	5000	83.57	0.93	0.82	2
	3	Top $k = 10$	J48	6807	83.33	0.94	0.84	3
	4	Top $k = 9$	J48	5000	83.10	0.93	0.84	4
	5	Top $k = 8$	J48	5000	81.90	0.93	0.81	6
	6	Top $k = 6$	J48	5364	81.90	0.92	0.82	7
	7	Top $k = 3$	KNN-10	5	81.90	0.92	0.82	8
	8	Top $k = 7$	J48	6030	81.90	0.92	0.81	9
	9	Top $k = 4$	J48	1997	81.67	0.93	0.85	11
	10	Top $k = 5$	J48	5000	81.43	0.92	0.81	13
	11	Top $k = 2$	J48	10	80.24	0.91	0.79	14

Table 7.66: Overview of the best results about the performance of detecting the sub-types of synthetic tasks (Tasks S1-S4) by stratified 10-fold cross-validation for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

Set	R_G	f	l	g	a	p	r
	1	$\mathcal{ACW}$	NB	50	73.81	0.88	0.72
	2	$\mathcal{CW}$	NB	25	73.10	0.87	0.71
	3	$\mathcal{AC}$	NB	25	64.76	0.83	0.65
Dyonipos	4	$\mathcal{W}$	NB	10	60.00	0.77	0.54
	5	$\mathcal{AW}$	NB	10	59.76	0.79	0.56
	8	$\mathcal{C}$	NB	250	48.57	0.71	0.51
	9	$\mathcal{A}$	KNN-1	10	47.14	0.70	0.48
SWISH	6		J48	25	55.48	0.77	0.52
TaskPredictor 1	7		J48	175	55.48	0.74	0.56

Table 7.67: Overview of the best results about the performance of detecting the sub-types of synthetic tasks (Tasks S1-S4) by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

approach achieved a higher accuracy than all the existing approaches on this dataset. More specifically, all UICO's *Top k* feature combinations performed better than all the existing approaches. The best one, the *Top* $k = 20$, significantly outperformed the best Dyonipos feature combination $\mathcal{ACW}$, the *SWISH* approach and the *TaskPredictor 1* approach 11.43%, 29.76% and 29.76% respectively.

7.6.5 Research Question: Can the analytic and the synthetic knowledge-intensive task models of the task instances be automatically detected?

The goal of this evaluation was to answer the question *"Can the analytic and the synthetic knowledge-intensive task models of the task instances be automatically detected?"*. The dataset on which the question was investigated contained 132 tasks from 18 users. The distribution of the task instances in respect to the classes (analytic and synthetic) is shown in Table 7.68. An overview of the UICO task detection results is given in Table 7.69. The results were achieved by applying 10-fold cross-validation on the training instances. A training instance was built for each task instance independently.

Set	**Analytic Tasks**	**Synthetic Tasks**	**Sum**
Dataset CV	67	65	**132**

Table 7.68: This table shows the distribution of the training/test instances for the knowledge-intensive task models analytic and synthetic tasks for stratified 10-fold cross-validation which were recorded on laboratory computers.

Feature Categories: The highest accuracy of 93.23% was achieved by the combination of all 50 features with the J48 algorithm (g=3500, p=0.93, r=0.93). This feature combination resulted in the global rank $R_G = 4$. A 1.48% lower accuracy values was obtained by the *application category* with 91.76% and the KNN-5 algorithm on only 125 attributes (g=125, p=0.92, r=0.92). The *resource category* achieved 87.75% accuracy (l=J48, g=2000, p=0.88, r=0.88) and hence got the third rank in the feature category ranking. The accuracy values resulting from the feature categories' evaluations were between 80.16% and 93.24%. In comparison with the performance of the single features the best two feature categories, the combination of all features of all categories as well as the *application category*, outperformed all single features in terms of accuracy.

Single Features: The best performing single feature was the *acc. obj. name* feature with an accuracy of 91.10% (g=25, p=0.91, r=0.91). The *window title* feature was the one with the second highest accuracy with 86.48% (l=J48, g=279, p=0.87, r=0.87) and hence 4.62% worse than the best single feature's accuracy. The third rank of the best performing single features went to the *used res. metadata* feature which achieved an accuracy of 85.66% on 1813 attributes (l=J48, p=0.86, r=0.86). The range of the numbers of attributes for the best classifier runs of the single performing features was between 3 and 279 attributes. The only exception was the *used res. metadata* feature which had its best classifier run with 1813 attributes. The accuracy values for the best 15 single features ranged from 75.77% to 91.10%. In comparison with the *Top k* feature combinations non of the single features achieved a higher accuracy.

Top k Features: The *Top k* best single feature combinations achieved accuracy values ranging from 92.53% to 94.73% and hence only differ among each other by 2.2%. The highest accuracy value resulted from the *Top* $k = 6$ (l=J48, a=94.73%, g=2000, p=0.95, r=0.95). Only 0.66% worse in terms of accuracy was the *Top* $k = 3$ with 94.07% accuracy (l=J48, g=3248, p=0.94, r=0.94) with the same algorithm but with 1248 attributes more than the *Top* $k = 6$. With 1000 attributes and the J48 algorithm the *Top* $k = 20$ obtained 93.90% accuracy and hence the third best ranking in this category. The range of the number of attributes for the best classifier runs of the algorithms was between 25 and 3000 attributes. The best three *Top k* best single feature performance combinations, *Top* $k = \{3, 6, 20\}$, outperformed all the other feature categories and single features in terms of accuracy. All the *Top k* had their best classifier runs with the J48 decision tree learner.

Comparison with existing approaches: The best Dyonipos feature combination $\mathcal{CW}$, the *SWISH* and the *TaskPredictor 1* approach achieved 90.11%, 85.00% and 85.00% respectively. In comparison with the best UICO feature combination, the *Top* $k = 6$ with 94.73% outperformed the best Dyonipos $\mathcal{CW}$ feature combination by 4.72% as well as the *SWISH* and *TaskPredictor 1* approaches by 9.73%. The number of attributes of the *SWISH* , the *TaskPredictor 1* and the Dyonipos feature combinations for the runs with the highest accuracy were between 5 and 2392 attributes whereas the best UICO feature combination required 2000 attributes. A detailed comparison of the feature and classifier performance evaluation of the existing approaches is given in Table 7.70.

Set	R_S	f	l	g	a	p	r	R_G
Feat. Cat.	1	All Categories	J48	3500	93.24	0.93	0.93	4
	2	Application Cat.	KNN-5	125	91.76	0.92	0.92	13
	3	Resource Cat.	J48	2000	87.75	0.88	0.88	15
	4	Action Cat.	NB	75	81.15	0.81	0.81	19
	5	Ontology Str. Cat.	KNN-10	150	81.10	0.81	0.81	20
	6	Switching Seq. Cat.	NB	75	80.38	0.81	0.81	21
	7	Content Cat.	KNN-5	250	80.16	0.80	0.80	23
Single Feat.	1	acc. obj. name	KNN-5	25	91.10	0.91	0.91	14
	2	window title	J48	279	86.48	0.87	0.87	16
	3	used res. metadata	J48	1813	85.66	0.86	0.86	17
	4	acc. obj. value	KNN-5	150	85.66	0.86	0.86	18
	5	datatype properties	SVM-$C2^0$	3	80.27	0.80	0.80	22
	6	applications interact.	SVM-$C2^{-1}$	5	79.67	0.79	0.79	24
	7	app. switch seq.	SVM-$C2^0$	3	79.56	0.80	0.80	25
	8	resource content	SVM-$C2^{10}$	125	78.90	0.79	0.79	26
	9	application name	J48	26	78.90	0.79	0.79	27
	10	res. interact.	KNN-5	50	78.68	0.79	0.79	28
	11	concept instances	J48	50	78.19	0.78	0.78	29
	12	res. types interact.	SVM-$C2^0$	5	76.65	0.76	0.76	30
	13	objecttype properties	J48	57	76.59	0.77	0.77	31
	14	used resources	NB	251	76.43	0.77	0.77	32
	15	E level res. switch seq.	KNN-35	100	75.77	0.76	0.76	33
Top k Feat.	1	Top $k = 6$	J48	2000	94.73	0.95	0.95	1
	2	Top $k = 3$	J48	3248	94.07	0.94	0.94	2
	3	Top $k = 20$	J48	1000	93.90	0.94	0.94	3
	4	Top $k = 8$	J48	500	93.24	0.93	0.93	5
	5	Top $k = 4$	J48	3000	93.24	0.93	0.93	6
	6	Top $k = 15$	J48	3000	93.24	0.93	0.93	7
	7	Top $k = 7$	J48	500	93.19	0.93	0.93	8
	8	Top $k = 10$	J48	500	93.13	0.93	0.93	9
	9	Top $k = 9$	J48	2000	93.08	0.93	0.93	10
	10	Top $k = 5$	J48	750	92.53	0.93	0.93	11
	11	Top $k = 2$	KNN-35	25	92.53	0.93	0.93	12

Table 7.69: Overview of the best results about the performance of classifying task instances to analytic and synthetic tasks by stratified 10-fold cross-validation for each feature category, each single feature as well as the k top performing single features. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), the ranking in the corresponding section (R_S) and across sections (R_G) are also given.

Set	R_G	f	l	g	a	p	r
Dyonipos	1	$\mathcal{CW}$	SVM-$C2^1$	1000	90.11	0.90	0.90
	2	$\mathcal{ACW}$	SVM-$C2^5$	750	89.51	0.90	0.90
	3	$\mathcal{AC}$	J48	2392	85.82	0.86	0.86
	6	$\mathcal{AW}$	NB	50	82.64	0.82	0.82
	7	$\mathcal{W}$	J48	125	81.81	0.82	0.82
	8	$\mathcal{A}$	SVM-$C2^{-1}$	10	80.16	0.80	0.80
	9	$\mathcal{C}$	NB	826	77.25	0.77	0.77
SWISH	5		J48	414	85.00	0.85	0.85
TaskPredictor 1	4		KNN-5	5	85.00	0.85	0.85

Table 7.70: Overview of the best results about the performance of classifying task instances to analytic and synthetic tasks by stratified 10-fold cross-validation for Dyonipos combinations, Swish and TaskPredictor. The learning algorithm (l), the number of attributes (g), the micro precision (p), the micro recall (r), and the ranking (R_G) are also given.

Concluding Remarks: The evaluations of the task classification performance of classifying tasks into analytic and synthetic knowledge-intensive tasks showed that an accuracy of approximately 95% was reached when utilizing the UICO's *Top* $k = 6$ best performing single feature combination on this dataset. All the existing approaches were outperformed between 4.72% and 9.73% accuracy by a combination of six features specific to the UICO approach.

7.6.6 Finding the Best Features/Feature Categories

Feature Dominance Matrix: Table 7.71 displays the dominance matrix for the features and feature combinations as explained in Section 7.3.5.1. It shows that the combination of all 50 features (*All Categories*), *application category* and the *acc. obj. name* feature outperformed the other features and feature combinations most often. As expected the accessibility object's features as well as the *window title* feature performed really well. Surprisingly in comparison with the previous two experiments described in Section 7.4.9 and in Section 7.5.6 in which the *window title* feature achieved both times the second rank, this time this feature only obtained the seventh rank. All the feature categories were present in the top 15 rankings of the feature dominance matrix.

	All Categories	Application Cat.	acc. obj. name	Resource Cat.	used res. metadata	window title	Action Cat.	acc. obj. value	Switching Seq. Cat.	datatype properties	Ontology Str. Cat.	Content Cat.	applications interact.	concept instances	res. interact.	Ψ
All Categories	-	3	4	3	3	4	4	4	4	4	4	4	4	4	4	221
Application Cat.	1	-	4	3	3	4	4	4	4	4	4	4	4	4	4	219
acc. obj. name	0	0	-	3	3	4	3	3	4	4	4	4	4	4	4	212
Resource Cat.	1	1	1	-	3	2	3	4	4	3	4	3	4	4	4	208
used res. metadata	1	1	1	1	-	1	4	3	4	3	4	3	4	4	4	205
window title	0	0	0	2	3	-	3	3	4	3	4	3	4	4	4	204
Action Cat.	0	0	1	1	0	1	-	2	4	3	4	3	4	4	4	198
acc. obj. value	0	0	1	0	1	1	2	-	3	3	4	3	3	4	4	196
Switching Seq. Cat.	0	0	0	0	0	0	0	1	-	3	3	2	2	4	4	186
datatype properties	0	0	0	1	1	1	1	1	1	-	1	2	3	2	4	184
Ontology Str. Cat.	0	0	0	0	0	0	0	0	1	3	-	2	2	4	4	183
Content Cat.	0	0	0	1	1	1	1	1	2	2	2	-	3	3	3	182
applications interact.	0	0	0	0	0	0	0	1	2	1	2	1	-	3	4	180
concept instances	0	0	0	0	0	0	0	0	0	2	0	1	1	-	2	162
res. interact.	0	0	0	0	0	0	0	0	0	0	0	1	0	2	-	161

Table 7.71: Dominance matrix for feature categories and single features.

Paired T-Tests: The results of the statistical significance tests $T - Test^f$ and $T - Test^f_{rt}$ as described in Section 7.3.5.1 are highlighted in Table 7.72 and in Table 7.73 respectively. The $T - Test^f$ results showed that the top 8 ranked feature and feature combinations did not significantly outperform each other ($p \geq 0.05$ in the paired t-tests). The only exception to this was the *resource category* which was statistically significantly better on a $p < 0.05$ significance level than the *acc. obj. value* feature. When looking at the results of the $T - Test^f_{rt}$ evaluation in Table 7.72 one can observe that (i) the number of statistical significant pairs have increased, (ii) that the top 3 ranked features and feature combinations were the same and (iii) that 14 features appear in both tables on slightly different ranks. The feature dominance matrix, the paired t-test with and without rank transformation visualizes that 14 out of the 15 top features and feature combinations appear in all tables. These results suggested that these were the best performing, i.e., the most discriminative, features and feature combinations for this dataset.

7.6.7 Finding the Best Learning Algorithms

Classifier Dominance Matrix: Table 7.74 summarizes the results from the classifier dominance matrix computations as described in Section 7.3.5.2. The order of the best performing

	Application Cat.	All Categories	acc. obj. name	Resource Cat.	window title	used res. metadata	acc. obj. value	Action Cat.	Ontology Str. Cat.	applications interact.	Switching Seq. Cat.	datatype properties	Content Cat.	res. interact.	app. switch seq.	Ψ
Application Cat.	-	~	~	~	~	~	~	~	>	>	~	>	≫	>	>	47
All Categories	~	-	~	~	~	~	~	~	~	>	~	>	▷	>	>	46
acc. obj. name	~	~	-	~	~	~	~	~	~	>	~	~	≫	>	>	45
Resource Cat.	~	~	~	-	~	~	>	~	>	~	>	~	~	>	>	44
window title	~	~	~	~	-	~	~	~	~	>	~	~	~	>	>	44
used res. metadata	~	~	~	~	~	-	~	~	>	~	>	~	~	>	>	43
acc. obj. value	~	~	~	<	~	~	-	~	~	~	~	~	~	~	>	40
Action Cat.	~	~	~	~	~	~	~	-	~	~	~	~	~	~	~	37
Ontology Str. Cat.	<	~	~	<	~	<	~	~	-	~	~	~	~	~	>	37
applications interact.	<	<	<	~	<	~	~	~	~	-	~	~	~	~	~	35
Switching Seq. Cat.	~	~	~	<	~	<	~	~	~	~	-	~	~	~	~	34
datatype properties	<	<	~	~	~	~	~	~	~	~	~	-	~	>	~	32
Content Cat.	≪	◁	≪	~	~	~	~	~	~	~	~	~	-	~	~	31
res. interact.	<	<	<	<	<	<	~	~	~	~	~	<	~	-	~	29
app. switch seq.	<	<	<	<	<	<	<	~	<	~	~	~	~	~	-	25

Table 7.72: Significance matrix for the top 15 features and feature combinations without rank transformation.

	All Categories	Application Cat.	acc. obj. name	Ontology Str. Cat.	Switching Seq. Cat.	Content Cat.	used res. metadata	applications interact.	Resource Cat.	window title	datatype properties	acc. obj. value	app. switch seq.	Action Cat.	content in focus	Ψ
All Categories	-	~	>	>	>	▷	~	>	~	≫	>	≫	▷	▷	▷	53
Application Cat.	~	-	>	≫	>	≫	~	▷	~	>	▷	~	≫	~	≫	51
acc. obj. name	<	<	-	≫	≫	≫	~	▷	~	~	>	~	≫	~	≫	49
Ontology Str. Cat.	<	≪	≪	-	~	∼	~	~	~	~	~	~	>	~	~	36
Switching Seq. Cat.	<	<	≪	~	-	∼	~	∼	~	~	~	~	>	~	~	36
Content Cat.	◁	≪	≪	∼	∼	-	~	~	~	~	∼	~	~	~	≫	35
used res. metadata	~	~	~	~	~	~	-	>	~	~	~	~	>	≫	~	35
applications interact.	<	◁	◁	~	∼	~	<	-	<	<	~	~	>	~	~	35
Resource Cat.	~	~	~	~	~	~	~	>	-	∼	~	≫	>	~	~	34
window title	≪	<	~	~	~	~	~	>	∼	-	~	~	>	~	~	34
datatype properties	<	◁	<	~	~	∼	~	~	~	~	-	~	~	~	~	33
acc. obj. value	≪	~	~	~	~	~	~	~	≪	~	~	-	>	~	~	32
app. switch seq.	◁	≪	≪	<	<	~	<	<	<	<	~	<	-	<	∼	32
Action Cat.	◁	~	~	~	~	~	≪	~	~	~	~	~	>	-	~	31
content in focus	◁	≪	≪	~	~	≪	~	~	~	~	~	~	∼	~	-	29

Table 7.73: Significance matrix for the top 15 features and feature combinations with rank transformation.

classifiers was *J48*$\succ_{220}$*KNN-5*$\succ_{23}$*KNN-35*$\succ_{1}$*KNN-1*$\succ_{13}$*NB*$\succ_{4}$*KNN-10*$\succ_{197}$*SVM-lin*, whereas $c_1 \succ_d c_2$ indicates that classifier c_1 performed d times better than c_2 based on Ψ. The *J48 decision tree* learner seemed to perform significantly better than the other learning algorithms.

Paired T-Tests: The intention from the classifier dominance matrix was confirmed by the paired t-test results illuminated in Table 7.75 for "without" and "with" rank transformation respectively. Since there was no difference in these results they are displayed together in one table. A complete partial order of classifiers can be computed by using each column of Table 7.56 for the different significance levels. The classifiers that showed no significant performance differences were grouped together. The partial order of the results was {*J48*}≫{*NB, KNN-1, KNN-5, KNN-10, KNN-35*}≫{*SVM-lin*} (and {*KNN-5*}>{*KNN-10*}).

	J48	KNN-5	KNN-35	KNN-1	NB	KNN-10	SVM-lin	Ψ
J48	-	165	171	166	155	167	185	1009
KNN-5	87	-	126	130	134	139	173	789
KNN-35	81	123	-	116	139	134	173	766
KNN-1	84	114	135	-	133	126	173	765
NB	100	119	116	122	-	125	170	752
KNN-10	86	113	119	125	132	-	173	748
SVM-lin	81	93	93	94	97	93	-	551

Table 7.74: Dominance matrix for the classifiers $l \in \{$ J48, KNN-1, KNN-10, KNN-35, KNN-5, NB, SVM-lin $\}$

Paired t-tests performed based on the micro f-measures without rank transformation resulted in the similar partial orders {*J48*}≫{*KNN-1, KNN-5, KNN-10*}≫{*SVM-lin*} and {*J48*}>{*NB, KNN-35*}≫{*SVM-lin*} With rank transformation the resulting partial order was {*J48*}≫{*NB, KNN-1, KNN-5, KNN-10, KNN-35*}≫{*SVM-lin*}.

These results went hand in hand with the ranking obtained from the classifier dominance matrix in Table 7.74 and suggested the conclusion that the *J48 decision tree* learner performed better on this dataset as the *Naïve Bayes* (NB), the *k-Nearest Neighbor* (KNN) and the *linear Support Vector Machines* (SVM-lin) algorithms.

	J48	KNN-5	KNN-1	KNN-10	KNN-35	NB	SVM-lin	Ψ
J48	-	≫	≫	≫	≫	≫	≫	6
KNN-5	≪	-	~	>	~	~	≫	2
KNN-1	≪	~	-	~	~	~	≫	1
KNN-10	≪	<	~	-	~	~	≫	1
KNN-35	≪	~	~	~	-	~	≫	1
NB	≪	~	~	~	~	-	≫	1
SVM-lin	≪	≪	≪	≪	≪	≪	-	0

Table 7.75: Significance matrix for the classifiers $l \in \{$ J48, KNN-1, KNN-10, KNN-35, KNN-5, NB, SVM-lin $\}$ (with and without rank transformation).

7.6.8 Concluding Remarks

In Table 7.76 the results of laboratory experiment 3, which was described in this section, are summarized. It shows which research questions were investigated as well as the task detection performance results of the UICO approach in comparison with the Dyonipos, *SWISH* and *TaskPredictor 1* approaches. One can observe in Table 7.76 that the UICO approach outperformed all the other approaches on all research questions investigated.

The answers to the research questions were elaborated for the detectability of the sub-types analytic and synthetic knowledge-intensive task types in Section 7.6.8.1 and for distinguishing analytic and synthetic task types in Section 7.6.8.2. Remarks about the experiment itself were mentioned in Section 7.4.11.5.

Evaluations	**UICO**	**Dyonipos**	***SWISH***	**Task P. 1**
Task Models (8 TM)	**86.43%**	66.04%	56.76%	63.85%
Analytic Tasks (4 TM)	**97.14%**	96.90%	92.86%	95.48%
Synthetic Tasks (4 TM)	**85.24%**	73.81%	55.48%	55.48%
Analytic vs. Synthetic Tasks (8 TM)	**94.73%**	90.11%	85.00%	85.00%

Table 7.76: Overview of the results of the laboratory experiment 3. The task detection performances elaborated for Analytic & Synthetic, Analytic, Synthetic and Analytic vs. Synthetic in Section 7.6.2, Section 7.6.3, Section 7.6.4 and Section 7.6.5 respectively show that the UICO approach outperformed in terms of accuracy all the existing approaches. The highest achieved accuracy values for each evaluation are marked in bold.

7.6.8.1 Detectability of Knowledge-Intensive Tasks

According to the *CommonKADS* knowledge-intensive task classification knowledge-intensive tasks are *analytic* or *synthetic* tasks. Analytic tasks are further distinguished into *Analysis*, *Classification*, *Diagnosis*, *Assessment*, *Monitoring* and *Prediction*. Synthetic subtypes are *Synthesis*, *Design*, *Configuration Design*, *Assignment*, *Planning*, *Scheduling* and *Modeling*.

The detectability of a subset of the subtypes of analytic and synthetic task types were investigated via an experiment with 18 computer science students in Section 7.5. The accuracy of detecting these subtypes was 86.43% with a combination of 6 features specific to the UICO approach (see Section 7.6.2). This accuracy value was more than 20% higher than the one of the best evaluated existing approach. For studying the detectability of analytic tasks, the synthetic tasks were removed from the dataset (see Section 7.6.3). The evaluation of the detectability of analytic tasks resulted in an accuracy of 97.14% with 7 features specific to the UICO approach. The accuracy of the best classifier runs of the existing approaches ranged from 92.86% to 96.90% which almost reached the same accuracy value of the UICO approach. The synthetic tasks were also studied separately (see Section 7.6.4). Again the highest accuracy value of 85.24% accuracy was reached by 20 features specific to the UICO approach. The existing approaches were worse between 11.43% (Dyonipos) and 29.76% (*SWISH* and *TaskPredictor 1*).

These results showed that (i) subtypes of analytic tasks were easier to detected than sub-types of synthetic task types and that (ii) the feature combination of the UICO approach outperformed all the existing approaches on analytic tasks slightly but significantly on synthetic tasks. The more difficult detectability of synthetic tasks might also had an impact of the overall task detection performance evaluated on the whole dataset in Section 7.6.2.

7.6.8.2 Analytic vs. Synthetic Tasks

While in Section 7.6.8.1 the detectability of the subtypes of the *CommonKADS* knowledge-intensive task classification was reported, this section sums up the result of the task detection performance of *analytic* and *synthetic* tasks types elaborated in Section 7.6.5. With an accuracy of 94.73% a task instance was identified as an analytic or a synthetic task with 6 features specific to the UICO approach. This was a 4.62% to 9.73% higher accuracy than achieved by features and feature combinations of the existing approaches.

7.6.8.3 Remarks about the Experiment

Although the design and execution of the experiment were similar to the one described in Section 7.5 there were significant differences in (i) the types of tasks the participants were asked to perform and (ii) in the specification of these tasks. In the previous experiments described in Section 7.4 and in Section 7.5 the tasks were defined based on the role model of the company employee "Bill Adams" and in the computer science domain based on the one of the student "Georg Kompacher" respectively.

In this experiment the type of the task was described to the participants as well as the task itself. The descriptions of the task did not lead in any direction of how to execute the task. The form of the expected result was also not specified. In other words, there was no exact specification of the *"procedure"* to utilize in order to solve the given tasks. The reasons for this setting were (i) to not influence the participants in any way how to perform the tasks and (ii) to not direct them to the utilization of certain types of applications, resources or types of resources. The challenges with these conditions affected both, the participants and automatic classification of the tasks.

The vague task descriptions made it clear for the participants what the goals of the tasks were but left unclear what the expected form of the result had to look like. Participants asked whether to create an Excel sheet or if a simple text file would be sufficient. It seemed that some participants had issues in dealing with their freedom of choice.

Since the experiment did not specify or lead to an exact procedure but rather encouraged the creative thinking of the experiment's participants for solving the tasks in their own way, various forms of solving the tasks appeared in the recorded usage data.

This resulting diversity in application and resource usage was also a great challenge for automatic task detection. The thought was that the selected individual *procedures* for accomplishing a task would be specific for this task. Since the tasks were designed to be from a certain category (analytic and synthetic or their sub categories) the selected procedure would be specific for the category as well. The differences in the applied procedures would be user specific but the common points accross individual procedures would be category specific. Automatic task detection would then focus on learning the common points of the category specific procedure.

A further challenge was the fact that the sensors did not cover all the application that were installed on the testing machine. This fact can be observed by comparing the available sensors described in Section 3.4.1 and the listing of installed applications on the laboratory computers in Section 7.6. The thought was that automatic task detection would perform well without having *full sensor coverage.* The results on the laboratory experiment 3 dataset suggested that a rather high task detection performance would be possible even without a *full sensor coverage.*

7.7 Concluding Remarks and Open Questions

7.7.1 Discussion about the Proposed Ontology-based Task Detection Approach

The **advantages** of using an ontology-based user context model for task detection are the new possibilities of constructing features (see Section 5.3) for the machine learning algorithms. The experiments reported in this chapter showed that the features engineered from the user interaction context ontology (UICO) outperformed features proposed by existing approaches. Especially well worked the UICO features for detecting knowledge-intensive tasks.

A further advantage of the ontology-specific approach is that the data representation is easy to access. This means it is queryable with a rich query language SPARQL that allows an easy way to create various types of features based on the relationships of two entities or the structure of a set of entities of the user interaction context. In case of the UICO approach it made it possible to create features about the combination of the type of user interaction and the type of resource the user has interacted with. Furthermore time-based features like switching sequences between applications, resources, resource types or interaction types were possible to create.

The **disadvantages** of the utilization of an ontology-based approach for task detection is the amount of data that is required for representing the user interaction context as well as the introduced costs. Firstly, the amount of data is significantly influenced by the utilization of semantic technologies for storing the user interaction context data, i.e., the number of concept instances and relation between them (see Section 3.6). Secondly, following costs of the ontology-based task detection approach were identified:

- **CPU and Memory Costs:** The ontology-based task detection approach requires the UICO populated with user interaction context data. The population of the UICO is done via adding triples to the UICO. Depending on the number of triples already existing in the UICO this step requires a greater amount of memory and CPU usage (see Section 3.6.2). The same applies in the step of retrieving information from the UICO during the feature extraction and training instance construction process. A standard desktop computer with 2 gigabytes of memory and a double core CPU with 2 GHz is suggested to build to classify a training instance for a supervised learning algorithm based on the user interaction context data of a single task execution. For the training phase, i.e., the phase in which a classification model is learned based on several task executions, a server system is suggested in order to not decrease the user's productivity. The server system used for the evaluations described in this chapter was a 64-bit quad core server machine with 2.5 GHz CPUs and with 6 gigabytes of memory.
- **User Privacy & Security Costs:** The user interaction context data stored in the UICO is a very detailed representation of the user's interests, characteristics, utilized resources, the user's social network etc. The UICO not only stores high level concepts but also low-level sensor data. This could be a thread to the user's privacy because the UICO may include sensitive information, e.g., passwords, login information, credit card numbers or the content of a confidential documents or instant messaging conversation. Since the ontology-

based task detection approach extracts features based on the user interaction context data stored in the UICO, it could happen that sensitive information is extracted as part of a feature for training the machine learning algorithm. Possible solutions for securing the user's privacy are (i) to not use context sensors that sense user privacy related data, (ii) to filter the captured usage data based on predefined rules, or (iii) to encrypt the stored user interaction context data in the UICO. Addressing these issues was not in focus of this thesis but are mentioned here in order to raise awareness that such issues are important to focus on when deploying the ontology-based task detection approach in real world settings.

- **Usage Data Labeling Costs:** The ontology-based task detection approach is a supervised learning approach, more specifically a classification approach. This means that a classifier has to be trained with labeled usage data before a detection of a task is possible. The construction of a training dataset for a domain that results in a classification model which achieves good task detection accuracy values introduces costs. Costs are for example the time required to identify the tasks of the domain that should automatically be detected or the time needed for experts to record and label multiple task executions for training the classifiers.

7.7.2 Best Generalizing Context Features

For each of the three laboratory experiments the feature dominance matrices as described in Section 7.3.5.1 were computed. Based on these results a ranking of the features and feature categories was computed and visualized in Table 7.77. The computation of this ranking only considered a feature or feature category appearing once in the best 15 feature and feature categories of the feature dominance matrices of the three laboratory experiments. The average, standard deviation and standard error were also computed and are given in Table 7.77.

Based on the evaluations performed on the three datasets, the stability of the performance achieved by each feature and each feature category was studied (see Table 5.1). By computing a dominance matrix for each experiment (based on how often a feature/feature category outperforms the others) a ranking of the features/feature categories can be obtained. An overview of the results for the top 22 features/feature categories is presented in Table 7.77. Those are the features/feature categories that appear in the top 15 ranking in at least one dataset.

Several interesting insights are provided by Table 7.77. First, the good ranks of features and feature categories partly or totally engineered based on the ontology (cf. column O) clearly signals the positive influence on the task detection performance of adopting our UICO approach. Second, the best results are achieved by the *application category* and by the combination of all 50 features (*All Categories*). The fact that the *application category* performs slightly better also shows that it is not true that the more features are considered, the better the achieved classification accuracy is. Third, the single features achieving the best results are the *acc. obj. name* and the *window title*. Besides, the standard deviation of the *acc. obj. name* feature is one of the lowest, which indicates the good stability of its performance across datasets. The fact that the *acc. obj. name* feature performs slightly better than the well-known *window title* feature also signals the benefits of making use of the features derived from the accessibility objects. Fourth, if one reduces this table by considering only the features that appear in the top 15 rankings produced by the three

R_G	O	T	**Feature / Feature Category**	R_1	R_2	R_3	μ_R	δ_R^2	δ_R
1		x	Application Cat.	1	2	2	1,67	0,33	0,58
2	x	x	All Categories	3	1	1	1,67	1,33	1,15
3		x	acc. obj. name	4	4	3	3,67	0,33	0,58
4		x	window title	2	3	6	3,67	4,33	2,08
5	x	x	Resource Cat.	6	7	4	5,67	2,33	1,53
6		x	used res. metadata	9	6	5	6,67	4,33	2,08
7		x	acc. obj. value	5	12	8	8,33	12,33	3,51
8	x	x	Action Cat.	13	5	7	8,33	17,33	4,16
9	x		datatype properties	8	9	10	9	1	1
10	x		Ontology Str. Cat.	10	8	11	9,67	2,33	1,53
11	x	x	Switching Seq. Cat.	20	11	9	13,33	34,33	5,86
12		x	acc. obj. role	15	15	15	15	0	0
13	x		res. types interact.	19	10	16	15	21	4,58
14		x	Content Cat.	7	27	12	15,33	108,33	10,41
15	x		applications interact.	21	16	13	16,67	16,33	4,04
16	x		concept instances	22	14	14	16,67	21,33	4,62
17	x		res. interact.	31	13	15	19,67	97,33	9,87
18		x	content of EB	11	28	21	20	73	8,54
19		x	content in focus	12	30	18	20	84	9,17
20		x	acc. obj. role des.	14	25	30	23	67	8,19
21		x	used res. interact.	32	15	22	23	73	8,54
22	x	x	resource content	15	35	23	24,33	101,33	10,07

Table 7.77: Computation of the ranking of the features and feature categories. The global ranking R_G is given by the average μ_R of the rankings for the 3 laboratory experiment's datasets (R_1, R_2 and R_3) and by the standard deviation δ_R in case of a draw. The columns O and T are used to indicate which features/feature categories are ontology-based, text-based or both.

datasets, one can isolate what can be considered as being the best performing features: the *acc. obj. name* feature, the *window title* feature, the *used res. metadata* feature, the *acc. obj. value* feature, the *datatype properties* feature and the *acc. obj. role* feature. Because of the low standard deviation values associated with them, the performances of these six features also suggested to be stable across datasets. It is again worth noting that four of these features are new and specific to the UICO approach.

7.7.3 Best Generalizing Classifiers

In the three laboratory experiments described in this chapter the task detection performance in evaluating several research questions were studied. In particular next to the features and feature combinations the performances of the machine learning algorithms the J48 decision tree, the Naïve Bayes, the k-Nearest Neighbor and the linear Support Vector Machines were measured. The ranks achieved by the learning algorithms in the experiments based on the computed dominance matrices are highlighted in Table 7.78. One can observe that the J48 learner obtained the first rank as well as the lowest standard deviation. This indicates that the J48 learner was the most stable across the features and feature combinations of the three datasets. The Naïve Bayes algorithm performed very well on the first two experiment's datasets but not that well on the third experiment's dataset which only contained knowledge-intensive. Since the performance of these knowledge-intensive tasks involved a certain creative freedom of the experiment's participants because of the experiments conditions it seemed that the Naïve Bayes algorithm could not handle this freedom very well. The linear Support Vector Machines showed a rather well performance in experiment one but had a bad performance on the second and third experiments' datasets. The k-Nearest Neighbor algorithm showed a rather constant performance on all datasets but performed best on the knowledge-intensive tasks' dataset of experiment 3. Both the Naïve Bayes and the linear Support Vector Machines are linear classifiers and both performed rather bad on the third experiment's dataset while the k-Nearest Neighbor and the J48 decision tree algorithm performed well. This might be an indication that the decision boundary was non-linear.

R_G	**Classifier**	R_1	R_2	R_3	μ_r	δ_r^2	δ_r
1	J48	2	2	1	1,67	0,33	0,51
2	NB	1	1	5	2,33	5,33	2,04
3	KNN-5	4	5	2	3,67	2,33	1,50
4	KNN-35	6	4	3	4,33	2,33	0,69
5	KNN-1	5	6	4	5,00	1,00	1,00
6	KNN-10	7	3	6	5,33	4,33	1,58
7	SVM-lin	3	7	7	5,67	5,33	0,77

Table 7.78: Ranking of the best classifiers $l \in \{J48,\ KNN\text{-}1,\ KNN\text{-}10,\ KNN\text{-}35,\ KNN\text{-}5,\ NB,\ SVM\text{-}lin\}$ for the 3 laboratory experiment's datasets (R_1, R_2 and R_3) based on the dominance matrices. The average (μ) and the global rank (R_G) across the experiments are also given.

Exp.	**Without Rank Transformation**
1	$\{J48,NB\} \gg \{KNN\text{-}1,KNN\text{-}5,KNN\text{-}10,KNN\text{-}35\} \gg \{SVM\text{-}lin\}$
2	$\{NB\} \gg \{J48,KNN\text{-}1,KNN\text{-}5,KNN\text{-}10,KNN\text{-}35\} \gg \{SVM\text{-}lin\}$
3	$\{J48\} \gg \{NB,KNN\text{-}1,KNN\text{-}5,KNN\text{-}10,KNN\text{-}35\} \gg \{SVM\text{-}lin\}$ ($\{KNN\text{-}5\} > \{KNN\text{-}10\}$)
Exp.	**With Rank Transformation**
1	$\{J48,NB\} \gg \{KNN\text{-}1,KNN\text{-}5,KNN\text{-}10,KNN\text{-}35\} \gg \{SVM\text{-}lin\}$
2	$\{NB\} \gg \{J48\} > \{KNN\text{-}1,KNN\text{-}5,KNN\text{-}10,KNN\text{-}35\} \gg \{SVM\text{-}lin\}$
3	$\{J48\} \gg \{NB,KNN\text{-}1,KNN\text{-}5,KNN\text{-}10,KNN\text{-}35\} \gg \{SVM\text{-}lin\}$ ($\{KNN\text{-}5\} > \{KNN\text{-}10\}$)

Table 7.79: Partial orders of the best classifiers $l \in \{J48,\ KNN\text{-}1,\ KNN\text{-}10,\ KNN\text{-}35,\ KNN\text{-}5,\ NB,\ SVM\text{-}lin\}$ based on the statistical significance tests for the 3 laboratory experiment's datasets.

The partial orders of the classifiers computed based on the statistical significance tests without and with rank transformation are summarized in Table 7.79. It can be observed that the Naïve Bayes and the J48 decision tree learner outperformed the k-Nearest Neighbor and the Support Vector Machine learners. These results support the results of the classifier dominance matrix evaluations. It can be concluded that for automatic task classification on the datasets of these three laboratory experiments the Naïve Bayes and the J48 decision tree learner were the best ones and the k-Nearest Neighbors learners are the second best ones.

7.7.4 Detectability of Types of Tasks

The results of the three laboratory experiments described this Chapter 7 are illuminated in Table 7.80. Two evaluation methods were used to achieve these results: (i) stratified 10-fold cross-validation and (ii) train/test set evaluation. The evaluations methods were explained in detail in Section 7.3.3.

Evaluations	**Exp. 1**	**Exp. 2**	**Exp. 3**
Stratified 10-Fold Cross-Validation			
Detection of Task Models (5 TM, 7 TM & 8 TM)	88.55%	94.84%	86.43%
Routine vs. Knowledge-Int. (5 & 7 TM)	94.94%	100.00%	-
Routine (4 TM)	-	94.64%	-
Knowledge-Int. (3 TM)	-	100.00%	-
Standard Tasks (5 & 7 TM)	88.41%	98.57%	-
Personal Tasks (5 & 7 TM)	86.00%	94.05%	-
Analytic vs. Synthetic (8 TM)	-	-	94.73%
Analytic (4 TM)	-	-	97.14%
Synthetic (4 TM)	-	-	85.24%
Train/Test Set Evaluation			
Pers. based on Lab. Computer (4 TM)	94.85%	-	-
Pers. based on Std. Task (5 TM & 7 TM)	77.14%	92.42%	-
One Expert and User Group (5 TM)	75.37%[1]	-	-
Expert Group and One User (5 TM)	98.53%[2]	-	-

Table 7.80: Overview of all task detection performance results of the real-world user study and the three laboratory experiments.

[1] Achieved with an extended feature combination evaluation as described in Section 7.4.7. The standard evaluation methodology described in Section 7.3.3 only reached 72.91% because not all feature combinations were evaluated.

[2] Achieved with an extended feature combination evaluation as described in Section 7.4.8. The standard evaluation methodology described in Section 7.3.3 only reached 92.65% because not all feature combinations were evaluated.

Task Model Detection: The performance of detecting the task models of the task instances of the experiments' datasets via stratified 10-fold cross-validation showed that an accuracy between approximately 88% and 95% could be reached. These high accuracy values were achieved on tasks with various granularity levels, from two different domains and observed from over 40 different users.

Routine and Knowledge-Intensive Tasks: Routine and knowledge-Intensive tasks are two distinguished task types. The identifiability of these task types by automatic means was investigated in the first as well as second laboratory experiment and obtained accuracy values of 94.94% and 100.00% respectively. These results show that on these two datasets routine tasks and knowledge-intensive tasks could be detected with a high accuracy. A further result of the second experiment was that the task models of routine tasks and knowledge-intensive tasks were detected with an accuracy of 94.64% and 100.00% respectively. The 100.00% accuracy of detecting knowledge-intensive tasks looked promising but suspicious in two ways: (i) only three knowledge-intensive task models were distinguished and hence only a three-class classification problem had to be solved and (ii) these task models were very different such that it could be too easy for the classifier to find separating features. For a better understanding what kind of knowledge-intensive tasks are detectable a third experiment was designed.

The *CommonKADS* [Schreiber *et al.*, 1999] knowledge-intensive task categorization served as a base for designing the experiment's tasks for studying the detectability of knowledge-intensive tasks. The categorization distinguishes analytic and synthetic tasks. With an accuracy of 94.73% a task instance was classified into one of these task types. Both task types also have multiple subtypes. The detectability of these subtypes was investigated further. Task instances were correctly assigned to one of the eight subtypes with an accuracy of 86.43%. When focusing on the four subtypes of analytic tasks and on synthetic tasks separately an accuracy of 97.14% and 85.24% was obtained. This shows that knowledge-intensive task categories were successfully identified with a high accuracy and hence well distinguishable among each other. The free choice of the participants in the way how to perform and to complete the tasks encourages the generalizability of these results.

7.7.5 Comparison with Related Work

The most popular features identified for having a high discriminative power among tasks are the `window title` feature [Granitzer *et al.*, 2008; Oliver *et al.*, 2006; Shen *et al.*, 2006], the `file path/web page URL` feature [Shen *et al.*, 2006], and the `content in focus` feature [Granitzer *et al.*, 2008]. In this research's findings the feature choice of these approaches could be confirmed and is compared to the novel context features and feature categories introduced by the UICO approach.

In terms of attributes used for training the machine learning algorithms an interval of 200-300 attributes is suggested to be sufficient by [Granitzer *et al.*, 2008; Shen, 2009]. This dissertation research's results also suggest that only a small ratio of attributes are required to successfully identify tasks. The best overall accuracies were obtained on the interval between 100-500 attributes. The results that lead to this interval can be observed in the tables presented in Section 7.4, Section 7.5, Section 7.6 for laboratory experiment 1, 2 and 3 respectively.

In the task detection experiments reported in [Lokaiczyk *et al.*, 2007] the SVM learning algorithm was mentioned as the one with the highest achieved accuracy. In [Granitzer *et al.*, 2008] the good performance of the SVM learning algorithm was confirmed and the high accuracy achieved by the KNN learner highlighted. On the three laboratory experiment's datasets the SVM showed the worst accuracy and f1-measures. The good performance of the KNN learner could be confirmed. In contradiction to [Granitzer *et al.*, 2008] the Naïve Bayes learner performed very well across the three laboratory experiment's datasets.

7.7.6 Open Questions

There are three open questions still requiring further research efforts: (i) the best combination of features, (ii) the best generalizing classifier and (iii) the generalizability of the results regarding which tasks are automatically detectable.

1. Since it is not possible nowadays to investigate all possible combinations of the 50 UICO features as highlighted in Section 8.1.3.4 the question *"What is the best feature combination?"* remains an open question.
2. This research investigated the task detection performance of four types of machine learning algorithms: J48 decision tree, k-Nearest Neighbor, Naïve Bayes and linear Support Vector Machines. There are a lot more classifiers that have to be studied for their applicability to the classification problem *"task detection"* in order to answer *"Which classifier should be used for task detection?"* or *"Which combination of classifiers should be used for task detection?"*.
3. The degree of generalizability of the results of the evaluations of the investigated research questions require further experiments in laboratory as well as real world settings. Especially the influence factors to automatic task detection, like for example the *computer environment* or the *specific* goal, the *type of task*, need to be further investigated for datasets including tasks of other users from other domains. Although the results are promising, the question *"How much do the influencing factors impact automatic task detection?"* remains open.

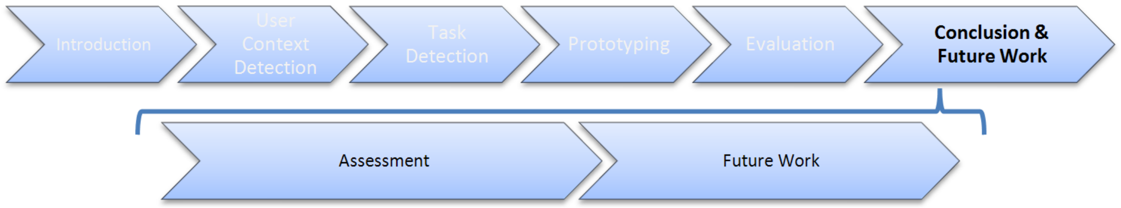
7.8 Summary

This chapter reports on the design and execution of three large-scale laboratory experiments. Through these experiments three task detection datasets consisting of over 500 tasks from over 40 users of two different domains were collected for evaluating the ontology-based task detection approach. Several insights were gained from the performed evaluations:

1. First, combinations of features engineered from the UICO almost always outperformed feature combinations suggested by existing task detection approaches on the evaluation datasets.
2. Second, the J48 decision tree and Naïve Bayes classifiers provide a better classification accuracy than other classifiers.

3. Third, six single features could be isolated that show a good discriminative power for classifying tasks as well as a stable performance across the evaluation datasets. These features are the *acc. obj. name* feature, the *window title* feature, the *used res. metadata* feature, the *acc. obj. value* feature, the *datatype properties* feature and the *acc. obj. role* feature.
4. Fourth, the best overall task detection results were achieved by the *application category* and by the combination of all 50 features (*All Categories*).
5. Fifth, even though it could seem easier to classify routine tasks, the results seem to suggest that knowledge-intensive tasks can be classified as well as routine tasks.

8 Conclusion and Future Work



The more a system knows about the user and her current task the better it can support her. This dissertation introduces and evaluates an ontology-based approach for studying and enhancing automatic task detection on the computer desktop. The final chapter reflects on the goals set at the beginning of this research in form of an assessment regarding the main research question and the sub-research questions. Furthermore the assessment includes a discussion about the generalizability of the achieved results and a discussion about the "value and effort" of the proposed approach. Finally, this chapter highlights future directions in the areas of task detection, work-integrated learning, personal information management, information retrieval and computer-supported collaborative work, inspired by the contributions of this research work.

8.1 Assessment

This section serves as a self-assessment part of this thesis to reflect on the achievements in respect to the asked research questions as listed in Section 1.2. In the following the investigated research questions are reflected from a retrospective viewpoint.

The **main research hypothesis** investigated by this dissertation is:

> *"The accuracy of automatic task detection can be enhanced by features engineered from an ontology-based user interaction context model in comparison to features and feature combinations of existing approaches."*

The proposed ontology-based task detection approach (UICO approach) utilizes an ontology-based user interaction context model, referred to as the user interaction context ontology (see Section 3.3), for engineering features for machine learning algorithms in order to do automatic task detection on the computer desktop (see Section 5.3). The task detection performance of these features was evaluated on three independent datasets (see Chapter 7). The evaluation results show that a high accuracy in correctly detecting different kind of tasks can be achieved. In order to measure whether the features of the UICO approach increase or decrease the task detection performance, three task detection approaches were selected from the literature for comparison purposes. The task detection performance of the features suggested by the DYONIPOS, SWISH and TaskPredictor 1 approaches (see Section 4.4) were evaluated on the same datasets as the UICO features. The comparisons of the resulting accuracy values show that on the first and third datasets the features from the UICO approach achieve higher accuracy values than the ones of the studied approaches. Although the accuracy values of the investigated features of all studied approaches were quite similar on the second dataset, the UICO features outperformed the features of the other approaches in most of the times. The third dataset consists only of knowledge-intensive tasks. The users were given a great amount of freedom in choosing the applications and resources for accomplishing the tasks. On this knowledge-intensive tasks dataset the UICO approach achieved a more than 20% higher accuracy than the other studied approaches. This dissertation research shows that, when considering more sophisticated tasks, involving more freedom in their execution or in the produced result, approaches that rely heavily on the "window title" feature does not achieve good accuracy results during task classification, and that more sophisticated feature combinations are needed.

The results of the evaluations of the UICO approach in comparison to the studied approaches show evidence that the research hypothesis can be accepted based on the studied datasets. However, there are limitations regarding the datasets used for comparing the task detection approaches. One limitation is the utilization of non-standardized datasets. Since there are no standard datasets freely available that allow an objective comparison of the task detection approaches, new datasets were collected and utilized for investigating the research question. Although (i) the tasks studied in the user experiments were freely chosen by experts of the studied domains, (ii) the users were free in the way they performed the tasks, (iii) the sequence of the tasks to be executed was randomized and (iv) the experiments were carried out in a laboratory setting, someone could argue that these datasets have been specifically constructed to favor a specific task detection approach. In order to overcome this limitation the task detection community has to agree on a standard dataset for objectively evaluating task detection approaches. A second limitation is the small number of datasets on which the task detection evaluations took place. One dataset used for testing the research hypothesis was collected during an experiment in the domain of the Know-Center GmbH. with a set of 5 tasks, and the other two datasets in the domain of computer science students of Graz University of Technology with two sets of 7 and 8 tasks respectively. It must be admitted that (i) the investigated tasks did not cover all the tasks of the domain, (ii) only a small set of representatives of the investigated domain participated in the experiments and (iii) only two domains were studied. For resolving these limitations other tasks of the studied domains have to be investigated as well, other representatives of the studied domain are required to perform the same tasks, and further experiments have to be performed in other domains.

This dissertation research successfully identified six features out of 50 features engineered from an ontology-based user interaction context model that show a high accuracy and stable performance across all three studied datasets (see Section 7.7.2). These are the *acc. obj. name* feature, the *window title* feature, the *used res. metadata* feature, the *acc. obj. value* feature, the *datatype properties* feature and the *acc. obj. role* feature. Since feature engineering is a key aspect in a machine learning task, these features can be considered as a valuable suggestion for other machine learning based task detection approaches regarding which features to include in a set of features used for task detection. Furthermore, this dissertation research also investigated into combinations of single features. The results suggest that, only a small set of features are required to achieve high task detection performance. It seems that, this small set when constructed of the best single performing UICO features, referred to as the Top-k feature combinations in this thesis, outperform each single feature as well as the combination of all 50 UICO features. On the other hand, feature engineering is data dependent. For task detection this means that, the set of features achieving high task detection accuracy values can vary for different settings, i.e., for a different set of tasks and for different domains. In order to test the applicability of a feature set for task detection in another setting one has to run experiments for investigating the domain, the tasks of the domain and the users of the domain.

The UICO approach enabled the engineering of a set of 50 features (see Section 5.3) covering various aspects of the user interaction context that can be used for constructing training instances for the machine learning algorithms. Having a pool of features speeds up the adaption process for task detection to another domain because identifying a feature combination with an appropriate accuracy for the requirements of a domain is faster than starting by engineering features. In this dissertation it is shown that the 50 features can successfully adapt to both of the studied domains, the domain of the Know-Center GmbH. and the domain of computer science students of Graz University of Technology. However, it has to be noted that, investigating two domains has limitations in claiming the adaptability of a set of features. Further experiments in other domains would be necessary to accept or reject such a claim.

Task detection is classically seen as a machine learning problem. This dissertation agrees on this view and confirms on the studied datasets that, the task detection problem can be solved with a high accuracy with classification algorithms. Four different classifiers with different parameter settings have been evaluated throughout this dissertation research: the Naïve Bayes (NB), the Linear Support Vector Machine (SVM) with cost parameters $c \in \{2^{-5}, 2^{-3}, 2^{-1}, 2^{0}, 2^{1}, 2^{3}, 2^{5}, 2^{8}, 2^{10}\}$, the J48 decision tree (J48) and the k-Nearest Neighbor (KNN-k) algorithm with $k \in \{1, 5, 10, 35\}$. The results of the performed task detection evaluations show that, the J48 decision tree algorithm and the Naïve Bayes learner achieve higher task detection accuracies than the k-Nearest Neighbor and the Linear Support Vector Machine algorithms on the studied datasets. The J48 decision tree and the Naïve Bayes algorithms also show a good stability across the studied datasets (see Section 7.7.3). However, next to the limitations introduced by the number of datasets studied there are many more machine learning algorithms available in the machine learning community that could prove to be valuable for task detection. By studying other classifiers or combinations of classifiers it could be possible to further increase the task detection performance.

An ontology-based task detection approach has several **advantages** for the feature engineering for the machine learning problem "task detection": (i) the possibility to construct various types of features based on the populated ontology ranging from text-based, ontology structure-based, time-based to graph-based features, (ii) the easy access of the data representation with the rich query language SPARQL and (iii) the flexible data representation that allows to easily adapt to the requirements of other domains. The **disadvantages** of this approach are (i) the massive amount of user interaction context data processed and stored in the ontology which has an impact on the CPU and memory requirements, (ii) the fine-granular storage of data about the user which can be seen as a threat to the user's privacy, and (iii) the effort introduced by the requirement of labeled training data for supervised machine learning. A more elaborated discussion about advantages and disadvantages of ontology-based task detection can be found in Section 3.6.1. For a discussion about the value and effort of the UICO approach and when to use it, the reader is referred to Section 8.1.4.

8.1.1 Automatic User Context Detection

Various existing approaches to **user context detection** were presented in the literature overview in Chapter 2. In this research work the following sub-research questions were investigated:

- To what extent is it possible to automatically and unobtrusively observe how users interact with resources, applications and the operating system available on their computer desktops?

⇒ From the software technology point of view, there is a wide area of possibilities to observe the user's interactions with resources and applications. This is due to the fact that an operating system, in this research's case Microsoft Windows XP and Vista, exposes interfaces in form of *dynamic link libraries* (DLL) or the *.Net Framework* to hook into their internal event queues. Applications also expose such interfaces and libraries. Furthermore most of the applications provide a plug-in mechanism in order to access the internal application states and the user triggered events, i.e., the user's usage data. Evidence for this is provided by the comprehensive set of developed context sensors and their capabilities of sensing the user interaction context. A challenge discovered during the development of the sensors was to hook into Java[1], Microsoft Silverlight[2] and Adobe Flash[3] applications from an external context sensing program.

- Which aggregation levels can be reached with what kind of techniques when utilizing the automatically captured contextual information about the user's interactions with resources, applications and the operating system on the computer desktop?

⇒ The first aggregation level when applying the proposed conceptual model, the *Semantic Pyramid*, is the event level that comprises a single user interaction with a resource. Discovering resources in the raw context sensor data stream is highly dependent on the reliability and capabilities of the context sensors. Techniques for resource discovery are regular expressions, information extraction and direct resource identification as described in Section 3.5.2. The aggregation to the second level, the event block level, which groups together user interactions on the same resource, can also be successfully achieved by a rule based approach (see Section 3.5.3). The next aggregation step is the task level, which is difficult to achieve with a rule-based approach. Reasons for this are (i) the subjective definition of a task by

[1]Java, `http://java.sun.com/`
[2]Microsoft Silverlight, `http://silverlight.net/`
[3]Adobe Flash, `http://www.adobe.com/products/flash/`

each user, (ii) the large variety of different tasks and (iii) the multiple ways a task can be performed and successfully completed. The standard approach of handling these issues is to model task detection as a machine learning problem which was also followed in this research. The results of this research work support this standard approach to automatic task detection.

8.1.2 Automatic Task Detection

In the area of **task detection** several research approaches were presented and discussed in Chapter 4. Most of them modeled task detection as a machine learning problem, more specifically as a classification problem. Next to the overall research question, the following sub-research questions were also investigated:

- What kind of tasks can be detected?

⇒ The automatic detectability of over 500 different tasks from over 40 users from two different domains (Know-Center GmbH. and computer science student at Graz University of Technology) ranging from *"plan a trip"* to *"algorithm programming"* was investigated. High accuracy values were reached with the proposed ontology-based task detection approach on the gathered datasets. About 88% to 95% of the task instances were correctly classified to the corresponding task models, based on the evaluation on three large-scale laboratory user experiments' datasets. Routine tasks could be distinguished from knowledge-intensive tasks with an accuracy ranging from about 95% to 100%, the evaluations based on two laboratory datasets. A special focus was put on knowledge-intensive tasks in the third laboratory experiment in which the detection of different types of knowledge-intensive tasks according to the *CommonKADS* [Schreiber *et al.*, 1999] classification were investigated. An accuracy of about 95% was obtained for classifying a task instance into an analytic or a synthetic task. Task instances of eight sub-categories of the *CommonKADS* classification were correctly identified by about 86% accuracy. Personal and standard tasks as defined in Section 7.4 were almost as well detectable. An accuracy between 86% to 99% was reached on two datasets. This research successfully started to explore what kind of tasks are detectable with a machine learning approach by starting the investigation with routine and knowledge-intensive tasks. For getting a deeper understanding how these results generalize to other tasks, task types, users and domains, further experiments in laboratory and real-world settings are required.

- Are there user interaction context features/feature combinations automatically observable by context sensors on the computer desktop that influence the performance of automatic task detection more than others?

⇒ This hypothesis can be accepted. The task detection performance of features and feature combinations specific to the ontology-based task detection approach showed a difference of the discriminative power of features and feature combinations for the classification of tasks. Evidence for this are the task detection performance evaluations on the conducted experiments' datasets as well as the statistical significance tests and the dominance matrix calculations carried out in this dissertation. The positive influence of specific user context features on task detection performance could be an indication that it is not necessary to sense *"everything"* about the user's interactions with her computer desktop. This would have an impact on what kind of sensors have to be developed, i.e. which context features have to be sensed to achieve a reasonable task detection performance. It would also impact the user's system performance because capturing less data normally leads to less CPU requirements. Furthermore, if the well-performing features for supervised machine learning algorithms in laboratory settings are known, they can provide a first hint about which features should be utilized in an unsupervised learning approach and in real world settings. Finding the best user interaction context feature combination for the ontology-based task detection approach is still an open research question since with today's computers it is not possible to evaluate all possible feature combinations (see Section 8.1.3.4). Further research work is necessary to approximate a good feature combination as well as discovering novel features with a high discriminative power.

- Which classifiers should be used for automatic task detection?

⇒ The results of the task detection performance evaluations based on three independent laboratory datasets suggest that the J48 decision tree learner and the Naïve Bayes algorithm are the ones to choose for task detection. Since only four types of algorithms, namely the J48 decision tree learner, the Naïve Bayes algorithm, k-Nearest Neighbor algorithms and linear Support Vector Machines, were investigated, this question can not be answered completely in respect to the variety of available machine learning algorithms. The investigation of the applicability of other classifiers is part of future research work.

- Does the computer desktop environment influence the performance of automatic task detection?

⇒ In the first laboratory experiment described in Section 7.4.5 this question was investigated. The results suggested that there is a small influence of about 8% in terms of accuracy on the experiment's dataset. The results showed that the task detection on a personal computer was better than on the laboratory computer. This was in contrast to the expected result from the experiment design point of view since a laboratory computer seemed to be a more controlled environment in comparison to an employee's self-administrated personal computer. The following reasons might be responsible for this effect: (i) the classifiers favored the variability of the observed task execution data on the personal computers in the training process, (ii) the peculiarities of the tasks and (iii) too few tasks were studied. A longer discussion about this topic can be found in Section 7.4.11.1. In order to understand this effect better further experiments with more and different tasks would be required.

- Can a single expert train a classifier in advance such that it detects other users' tasks?

⇒ The results of the evaluation in Section 7.4.7 on a single dataset showed that an accuracy of 72.91% can be reached when utilizing the UICO's *Top* $k = 2$ best single performing feature combination on this dataset. Through an extended evaluation of all feature combinations consisting of 2-6 features of the 10 best performing single features an accuracy increase of 2.34% was achieved and resulted in an accuracy of 75.37%. This can be seen as an indication that (i) although calculating multiple combinations of the best single performing features only leads to a small accuracy increase and (ii) that one expert is not enough for classifier training in order to reach a high accuracy. The preliminary answer to this research question is *yes*, but with a limited accuracy. Further experiments are required to study this question in greater detail.

- Can a group of experts train a classifier such that it recognizes the tasks of another user?

⇒ The task detection performance evaluations on the dataset with 203 tasks from multiple experts and 68 tasks from a single user resulted in an accuracy of about 93% with the standard evaluation and of about 98.5% with the extended evaluation as described in Section 7.4.8. These results suggested that the answer to this research question is *yes* on this dataset, but it has to be taken into account that only 5 task models from a single domain were studied.

- Can a classifier be trained with predefined standard task executions classify personal tasks correctly?

⇒ In the laboratory experiments 1 and 2 the evaluation of the task detection performance of training on predefined standard task executions and testing on personal ones resulted in accuracy values of about 77% (see Section 7.4.4) and 92% (see Section 7.5.3) respectively. These results seems promising since (i) the datasets included 218 tasks (113 standard/ 105 personal) for the laboratory experiment 1 and 134 tasks (68 standard/ 66 personal) for the laboratory experiment 2, (ii) there were 14 and 10 users tested respectively and (iii) the tasks were from two different domains. For accepting the hypothesis further experiments are necessary.

8.1.3 Generalizability

This section discusses the generalizability of the results obtained in the ontology-based automatic task detection experiments described in Section 7.4, in Section 7.5 and in Section 7.6. Some of the discussions presented here were also published in [Rath *et al.*, 2009a] and [Rath *et al.*, 2009d].

8.1.3.1 Generalizability to Other User Context Capturing Frameworks

User context capturing frameworks, CAM frameworks [Wolpers *et al.*, 2007], that observe user contextual information differ in terms of utilized sensors and of granularity of the captured CAM. The ontology-based user interaction context observation approach proposed in this research is very fine-granular since not only the path name, the URL of a document [Dragunov *et al.*, 2005] or the window title of the application in focus [Oliver *et al.*, 2006] but also the user's interactions with all desktop elements and application controls (accessibility objects) as listed in Section 3.4 are observed. In this research's approach every single interaction of the user with an application and a resource is important and hence captured, stored and analyzed. Using a different CAM framework could result in leaving out context features with a good task discriminative power (see Section 7.3.5.1) and could hence have a negative impact on the task detection performance. However, context features that are not listed in Table 7.77 showed a low discriminative power on the evaluation datasets and hence can be omitted.

8.1.3.2 Generalizability to Other Tasks and Domains

The generalizability of the results obtained by this research work to other tasks and domains is of course not completely possible because (i) only two domains with (ii) selected tasks were studied and (iii) only a sample of experts of the domain were involved in the experiments. However, this research work successfully discovered novel features and feature combinations engineered based on an ontology-based user interaction context model proposed in this research work. It also evaluated their task detection performance with different configurations of four types of classifiers. The results were compared to features and feature combinations proposed by other approaches and showed that the proposed ontology-based task detection approach achieved a higher accuracy value in most of the cases. Studying a task detection approach in such great detail and based on multiple experiments' datasets has not been done before based on the knowledge of the author of this thesis. A comparison of the task detection performance of the features and feature combinations unveiled that some of them performed really well across all three laboratory experiments/datasets. These results suggested that these features and feature combinations would also work well in other domains with other tasks and users. Among these features was also the well-known and well-performing `window title` feature already identified in previous research [Granitzer *et al.*, 2008; Lokaiczyk *et al.*, 2007; Oliver *et al.*, 2006; Shen & Dietterich, 2007]. A comparison of the classifier performances across the laboratory experiments showed that the J48 decision tree learner and the Naïve Bayes algorithm performed best. Lokaiczyk *et al.* [2007] also reported good performances for the Naïve Bayes algorithm on a single domain dataset with 1 user and 5 tasks.

8.1.3.3 Generalizability to Other Ontologies

The user interaction context ontology (UICO) proposed in this research (i) for automatic user interaction context detection and (ii) as a base for engineering features in order to enhance automatic task detection was built as a bottom up approach starting on the sensor data level. It stores very fine-granular information, like e.g., data and metadata about resources, applications and user interactions. Using another ontology would mean that the mapping mechanisms have to be adapted in order to fit the new ontology from the user interaction context observation perspective. From the automatic task detection perspective a new ontology would have an impact to the task detection performance because it would not be possible to construct some features based on the new ontology. The structure of the ontology would also have an influence. An interesting case would be to combine the UICO with a higher level ontology, e.g., the PIMO [Sauermann *et al.*, 2007] or the LIP ontology [Schmidt, 2007]. In that case it would be possible to develop further abstraction and aggregation algorithms that automatically populate the upper ontology based on the populated UICO. Based on the resulting combination it would then be possible to construct further features. However, the generalizability to other ontologies is not that straightforward because of (i) the underlying conceptual model which is specific to this research's approach and (ii) the fine-granular information encapsulated in the UICO and its relations.

8.1.3.4 Finding the Best Feature Combination

Evaluating all possible feature combinations for the UICO approach is not reasonable. The method for finding the best feature combination of the UICO approach for detecting tasks comprised the evaluation of the feature categories, single performing features and the *Top k* best performing single features is limited such that not all combinations of the 50 features were studied. From a theoretical point of view this seems to be a limitation because there could be special feature combinations which might have outperformed the studied combinations. From a practical point of view it is not reasonable to compute all possible feature combinations of 50 features because suppose n indicates the number of different features and k the maximal number of different features to combine then the total number of possible combinations is

$$\sum_{k=1}^{n} \frac{n!}{k! * (n - k!)}. \tag{8.1}$$

The number of all combinations of 50 features (n=50) results in

$$\sum_{k=1}^{50} \frac{50!}{k! * (50 - k)!} \approx 1.13 * 10^{15} \tag{8.2}$$

feature combinations. Suppose one can evaluate a single feature combination in 1 second, it would take one approximately $2.14 * 10^9$ years to evaluate all combinations for a single classifier configuration. Hence evaluating all possible 50 feature combinations is not reasonable with the computing power available for this research work. This means that the evaluation method chosen for the evaluation of the UICO approach might not have discovered the best feature combination possible for the task classification problems. However, the achieved results suggested that, even without exploring all feature combinations, high accuracy values can be reached.

As evidence that a higher accuracy can be reached by calculating multiple feature combinations, extended evaluations of feature combinations for single expert and multiple expert training were presented in Section 7.4.7 and in Section 7.4.8 respectively. In these evaluations all combinations of up to 6 features (k=2...6) among the best 10 single performing UICO features (n=10) were computed. The number of feature combinations in this case was

$$\sum_{k=2}^{6} \frac{10!}{k! * (10 - k)!} = 837. \tag{8.3}$$

The evaluation of the feature combinations took between 6 and 80 minutes per feature combination for evaluating the UICO pipeline with different numbers of selected attributes and different combinations of classifiers on a 64-bit quad core server machine with 2.5 GHz CPUs and with 6 gigabytes of memory. The duration was for example dependent on the number of combined features, and the type of features including the corresponding applied preprocessing steps. The best result of the extended evaluation was 98.53% accuracy and hence 5.88% better in terms of accuracy than the standard evaluation method employed.

The goal of this dissertation was to find features that work well in different situations. Focusing on optimizing task detection for a special domain, i.e., finding “the” best feature and classifier configuration, is considered as *fine-tuning* and hence was not in focus of this dissertation research.

8.1.4 Discussion about Value and Effort

The utilization of an ontology for user context detection and task detection has several advantages and disadvantages as discussed in Section 3.6 and in Section 7.7.1 respectively. This section elaborates on the value and the effort associated with the use of the UICO for task detection.

The UICO is **valuable** for task detection because (i) it enables the engineering of novel and different types of features for the machine learning problem *"task detection"* and (ii) it eases and speeds up the fine-tuning process of task detection for tasks of other domains. First, in the feature engineering process all the concepts, concept instances and properties of the UICO can be utilized to construct features. These can be structure-based, text-based or sequence-based features as shown in Section 5.3. Kröll *et al.* [2007] suggested to use graph-based features for detecting tasks. Since the UICO can be seen as a graph, this type of features can also be constructed from the UICO. Second, the UICO allows an easy and fast adaption to new task detection settings because the amount of data stored in the UICO can be configured in a flexible way. This means that it is not necessary to deploy all the context sensor and user interaction context analysis algorithms in order to be able to do task detection with the UICO. Since the ontology is a flexible data schema it is possible to choose a sub-set of the user interaction context that is stored in the UICO. An example here is to only store the features with the highest discriminative power for detecting tasks of a certain domain. This would reduce the overall memory and storage consumption as well as CPU requirements. On the other hand the UICO is easily extendable for new context information which allows an easy adaption to new context sensors required for specific domains, e.g., task detection for graphical designers.

Furthermore, the UICO approach has 50 different features that can be tested for their applicability for distinguishing tasks of a certain domain. This speeds-up the fine-tuning process since a pool of features is already available and does not have to be engineered from scratch. Hence this reduces the effort of fine-tuning to selecting the appropriate combination of features for a specific domain.

The **efforts** for utilizing the UICO are (i) the modeling of the UICO as well as the design and implementation of the automatic population mechanisms, (ii) the demanding CPU and memory requirements for populating and querying the UICO and (iii) the data overhead of utilizing semantic technologies for the data representation and for the data storage.

When should the UICO be used for task detection?

50 features were engineered based on the UICO and tested for their discriminative power for detecting different kind of tasks. Six of the features showed a good and stable performance across three datasets. These were the *acc. obj. name* feature, the *window title* feature, the *used res. metadata* feature, the *acc. obj. value* feature, the *datatype properties* feature and the *acc. obj. role* feature. All except the *datatype properties* feature are text-based features that would also work without the UICO. When focusing only on task detection then it is not suggested to use the UICO in case the achievable task detection accuracy with these five text-based features is satisfiable for the requirements of the domain. If this is not the case, using the UICO approach

with its pool of 50 features is suggested for adjusting task detection to a certain domain.

However, feature engineering is a key aspect in a machine learning task and data dependent. Hence the features and feature combinations achieving the highest accuracy for task detection can vary for different tasks and domains. The best discriminating features and feature combinations identified based on three independent datasets in the course of this dissertation research should be considered as a starting point for building an appropriate feature combination for a certain domain and not as the perfect ones for all tasks and all domains. Especially the feature combination should be fine-tuned for each domain separately since sometimes the combination of good and bad single performing features is better than the combination of just the best ones.

8.2 Future Work

This section highlights future research challenges and directions in the areas of task detection, work-integrated learning, personal information management, information retrieval and computer-supported collaborative work that are inspired by this research work.

8.2.1 Scaling of the Ontology-based Task Detection Approach

The ontology-based task detection approach proposed in this dissertation was evaluated on three independent datasets originating from three laboratory experiments in two different domains. These datasets consisted of task instances belonging to 2 to 8 task models/task categories. The ontology-based task detection approach achieved good accuracy values on these datasets. On an organizational level there are many more tasks to be handled by task detection algorithms. Part of future work will be to evaluate if the ontology-based task detection approach scales up to a higher number of tasks in order to answer the question *"What about detecting 2000 tasks in an organization?"*. So far, there is no evidence in the literature that task detection can handle such a high number of different tasks.

8.2.2 Real-Time Task Detection on the Computer Desktop

Task detection belongs to the field of *activity recognition* [Andrews *et al.*, 2004; Horvitz *et al.*, 1998, 1999; Philipose *et al.*, 2004]. In activity recognition the system observes a sequence of events, tries to determine and to understand the goals of the user, and responds to it. The topic online task detection belongs to this research field and is also part of user interaction context detection. The ontology-based user context observation approach proposed in this research work can capture parts of the task execution that involve interactions with the user's computer desktop. Based on these observations the task performed by the user should be automatically detected after the execution of the task (*task classification*). Detecting the task a user is performing in real-time is another challenge because (i) the user can utilize multiple resources at a time in different tasks or (ii) can switch between various tasks (*multi-tasking*). This real-time task detection is similar to task classification because detecting boundaries between tasks is also based on automatically sensed context features that discriminate tasks very well.

When looking at the features proposed and studied during this research the best ones do not include *"time"*, e.g. task length, which could limit their applicability to real-time task detection.

Lokaiczyk *et al.* [2007] studied the influence of the window size to classify parts of tasks and found out that the size of about 30 seconds work well. A similar interval was also proposed by Shen [2009]. In the Dyonipos task detection approach [Granitzer *et al.*, 2008] of this research the classification of event block was investigated with four text-based context features based on a real world settings dataset. Some drawbacks were found by using the boundaries introduced by event blocks for studying feature and classifier performance (see Section 7.2.2). Now there is a clearer picture about the discriminative power of features for classifying tasks. Future research will be to study their applicability to real-time task detection and task switch detection.

8.2.3 Better Support for Work-Integrated Learning

Work-integrated learning (WIL) [Lindstaedt *et al.*, 2008a,b, 2009b; Smith, 2003] strives to enhance task performance by fostering learning at the workplace. For supporting the learner well in her current situation her *user profile* (characteristics [Fischer, 2001; Ulbrich *et al.*, 2006], her interests [Goecks & Shavlik, 2000], her competencies [Ley *et al.*, 2008] etc.) as well as her *user context* [Lindstaedt *et al.*, 2008a; Schmidt, 2007; Wolpers *et al.*, 2007]) represent valuable information in order to improves the quality and accuracy of the support mechanisms. The outcomes of this dissertation research, more specifically the automatically observed user interaction context available trough the UICO, (i) allow the construction of detailed user profiles and (ii) provide an in depth description of the user's current situation.

User profiles can also be computed with the help of data mining techniques. These can be enriched further based on the detailed user interaction context information available in the UICO. Examples are the utilization of the user interaction context to identify the user's social network, the recognition of the favored (work and learn) resources, discovering information flows or unveiling the topics/resources of interest.

The user's current situation is also described in great detail such as, i.e., the current task, email or instant messaging communications as well as a list of recently utilized documents, files, folder, web pages, calendar entries, contacts, etc. for providing context-aware work-integrated learn support. Enhancing context-aware information retrieval [Fuhr, 2005] in general and for learning objects [Duval & Hodgins, 2003; Ochoa & Duval, 2006] specifically as well as utilizing this detailed information to characterize and automatically detect *"learning situations"* are envisioned.

8.2.4 Better Support for Personal Information Management

Information management is one of the great challenges of today's information society in which everyone has access to and stores a massive amount of data of various forms. Organizing this amount of data takes a long time and requires a lot of effort on the user side. Personal information management (PIM) is the research area that deals with finding, keeping, organizing, maintaining and evaluating personal information and making sense of the information [Jones, 2007]. Although several PIM tools [Bernstein *et al.*, 2008; Dumais *et al.*, 2003; Teevan *et al.*, 2008] were created to study and improve PIM, the user still has to do a lot of things manually. PIM sometimes fails because of this additionally introduced user effort to add relations between PIM objects [Sauermann *et al.*, 2007]. In order to overcome this additionally introduced user workload an automated relation creation mechanism would be beneficial. The automatic user interaction context observation mechanisms as developed and studied in this research would not only benefit the user in saving a lot of time in building these relations but would also enable PIM researchers to add the observed user interaction context as a new dimension to existing PIM tools. What kind of ways of leveraging real-time and long-term user interaction context information for PIM tools will be part of future research work.

8.2.5 Better Support for Information Retrieval

In the ACM SIGIR[4] workshop report [Allan *et al.*, 2003] contextual retrieval was defined as *"combining search technologies and knowledge about query and user context into a single framework in order to provide the most appropriate answer for a user's information needs"*. In a recent ACM SIGIR Forum meeting an information retrieval research agenda was elaborated, it mentioned three important elements regarding context [Callan *et al.*, 2007]: (i) *understanding the user* who is asking questions (search), (ii) the *underlying information domain* which represents the relationships between documents as well as other rich entities within them, and (iii) the *larger task* the user tries to accomplish. This thesis research effort contributes to these three elements by providing a up-to-date representation of the current user interaction context in real-time in form of an ontology-based user interaction context model as well as insights about how to recognize a user's task with a high accuracy. The design and development of information need detection algorithms and context-aware information delivery mechanisms were not in focus of this research but seemed to have a great potential as explored in [Rath *et al.*, 2008, 2007]. One of the future research directions will be the utilization of the rich user interaction context information in order to personalize search, disambiguate search queries, as well as improve ranking algorithms. A further interesting research direction is to explore automatic task detection in order to do *activity-based desktop search* [Chernov, 2008].

8.2.6 Better Support for Computer-Supported Collaborative Work

The computer supported collaborative work area has recognized the concept of *"awareness"* as a critical issue to focus on [Schmidt, 2002] since *"users who work together require adequate information about their environment"* [Gross & Prinz, 2003]. The environment of an individual encompasses her connections with other people, as well as with digital resources and actions (tasks or processes). If connections are not clear or hidden to the individual or to the group, the cost is a lack of awareness in the organization [McArthur & Bruza, 2003], which not only leads to inefficient cooperation but can even prevent it from being started. Unveiling the connections between persons, topics, tasks and processes to computer workers facilitates cooperative work by increasing the awareness of the personal social networks and the role of an individual in the organization, a project, or a group. These connections can be created and modeled manually but since *"organizations create, store, transfer and use more and more data within their boundaries"* [Maier & Sametinger, 2007], a better approach is to develop semi-automatic or even automatic tools to create and share them [McArthur & Bruza, 2003]. Based on emails, McArthur & Bruza [2003]

[4]ACM SIGIR: Special Interest Group on Information Retrieval, `http://www.sigir.org`

computed such kind of connections, and suggested using more global corpora as well as taking into account dynamic ones.

The proposed ontology-based user interaction context detection approach in this research work can support awareness by [Rath *et al.*, 2009c]: (i) detecting the user interaction context of a single user (i.e. the connections between her tasks, her used digital resources and her social network) and (ii) combining multiple user interaction contexts from several users for global awareness. Two advantages for increasing awareness can be gained: (i) representing the relations between the user and her close environment (*individual view*) and (ii) merging multiple individual user interaction context models into a global one (*organizational view*). Coming up with creative ideas, applications and services in order to enhance the awareness as well as computer supported collaborative work will be part of future work.

8.3 Summary

This chapter compared in form of a self-assessment the set goals of answering the research questions asked in Section 1.2 to the achieved goals of this research work. Finally, interesting future research challenges and directions suggested by this research work were presented in this chapter. Ultimately, the goal of this research work was to get to know the user and her tasks better for enabling a task-specific support in today's knowledge economy. The outcomes and results of this dissertation research allows the development of new applications and services to exploit the automatically and unobtrusively detected user interaction context including the user's task for various purposes (e.g., context-aware information retrieval, task detection, annotation of learning resources and user interruptions) as well as for various domains (e.g., technology-oriented knowledge management, technology enhanced learning and work-integrated learning, computer supported collaborative work, personal information management and semantic desktop systems).

Bibliography

Aalst, W. M. P. v. d., & Weijters, A. J. M. M. 2004. Process mining: a research agenda. *Computers in Industry*, **53**(3), 231–244. Elsevier Science B.V.

Aalst, W. M. P. v. d., Weijters, A. J. M. M., & Maruster, L. 2004. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, **16**(9), 1128–1142. IEEE Educational Activities Department.

Aalst, W. M. P. v. d., Weske, M., & Grünbauer, D. 2005. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, **53**(2), 129–162. Elsevier Science B.V.

Allan, J., Aslam, J., Belkin, N., Buckley, C., Callan, J., Croft, B., Dumais, S., Fuhr, N., Harman, D., Harper, D. J., Hiemstra, D., Hofmann, T., Hovy, E., Kraaij, W., Lafferty, J., Lavrenko, V., Lewis, D., Liddy, L., Manmatha, R., McCallum, A., Ponte, J., Prager, J., Radev, D., Resnik, P., Robertson, S., Rosenfeld, R., Roukos, S., Sanderson, M., Schwartz, R., Singhal, A., Smeaton, A., Turtle, H., Voorhees, E., Weischedel, R., Xu, J., & Zhai, C. 2003. Challenges in information retrieval and language modeling: report of a workshop held at the center for intelligent information retrieval at University of Massachusetts Amherst in September 2002. *SIGIR Forum*, **37**(1), 31–47. ACM.

Anderson, J. R. 1983. *The Architecture of Cognition.* Cambridge, MA, USA: Harvard University Press.

Ando, R. K., & Lee, L. 2001. Iterative residual rescaling. *Pages 154–162 of: SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York, NY, USA: ACM.

Andrews, S., Cai, L., Gondek, D., Greenwald, A., Grollman, D., Jonsson, A. M., Hall, K., Lease, M., Ng, B., Raiti, J., Sweetser, V., & Turner, J. 2004. Astrology: the study of astro teller. *In: Physiological Data Modeling Workshop at ICML '04: International Conference on Machine Learning.*

APOSDLE. 2006. *APOSDLE - Advanced Process-Oriented Self-Directed Learning Environment. Website: `http://www.aposdle.org`.*

Balamurugan, S. A., & Rajaram, R. 2008. Learning to Classify Threaten E-mail. *Pages 522–527 of: AMS '08: Proceedings of the 2008 Second Asia International Conference on Modelling & Simulation (AMS).* Washington, DC, USA: IEEE Computer Society.

Baldauf, M., Dustdar, S., & Rosenberg, F. 2007. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, **2**(4), 263–277. Inderscience Publishers.

Bardram, J. E. 2005. *The Java Context Awareness Framework (JCAF)- A Service Infrastructure and Programming Framework for Context-Aware Applications.* Lecture Notes in Computer Science, vol. 3468. Springer. Pages 98–115.

Belizki, J., Costache, S., & Nejdl, W. 2006. Application independent metadata generation. *Pages 33–36 of: CAMA '06: Proceedings of the 1st international Workshop on Contextualized Attention Metadata: Collecting, Managing and Exploiting of Rich Usage Information.* New York, NY, USA: ACM.

Bellotti, V., & Smith, I. 2000. Informing the design of an information management system with iterative fieldwork. *Pages 227–237 of: DIS '00: Proceedings of the 3rd Conference on Designing interactive systems.* New York, NY, USA: ACM.

Bernstein, M., Van Kleek, M., Karger, D., & Schraefel, M. C. 2008. Information scraps: how and why information eludes our personal information management tools. *ACM Transactions on Information Systems*, **26**(4), 1–46. ACM.

Biedert, R., Schwarz, S., & Roth-Berghofer, T. R. 2008. Designing a Context-Sensitive Dashbord for an Apaptive Knowledge Worker Assistant. *Pages 51–62 of: HCP '08 Proceedings, Part II, MRC '08: Fifth International Workshop on Modelling and Reasoning in Context.* TELECOM Bretagne.

Boardman, R., & Sasse, M. A. 2004. "Stuff goes into the computer and doesn't come out": a cross-tool study of personal information management. *Pages 583–590 of: CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* New York, NY, USA: ACM.

Boone, G. 1998. Concept features in Re:Agent, an intelligent email agent. *Pages 141–148 of: AGENTS '98: Proceedings of the Second International Conference on Autonomous Agents.* New York, NY, USA: ACM.

Bortz, J., & Döring, N. 2006. *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler.* 4 edn. Springer.

Braun, S., Schmidt, A., & Hentschel, C. 2007. Semantic Desktop Systems for Context Awareness - Requirements and Architectural Implications. *Pages 1–12 of: SemDesk: 1st Workshop on Architecture, Design, and Implementation of the Semantic Desktop held at ESWC '07: 4th European Semantic Web Conference.*

Bron, C., & Kerbosch, J. 1973. Algorithm 457: finding all cliques of an undirected graph. *Communications ACM*, **16**(9), 575–577. ACM.

Budzik, J., Hammond, K. J., & Birnbaum, L. 2001. Information access in context. *Knowledge-Based Systems*, **14**(1-2), 37–53. Elsevier Science B.V.

Bush, V. 1996. As we may think. *interactions*, **3**(2), 35–46. ACM.

Callan, J., Allan, J., Clarke, C. L. A., Dumais, S., Evans, D. A., Sanderson, M., & Zhai, C. 2007. Meeting of the MINDS: an information retrieval research agenda. *SIGIR Forum*, **41**(2), 25–34. ACM.

Catarci, T., Dix, A. J., Katifori, A., Lepouras, G., & Poggi, A. 2007. *Task-Centred Information Management.* Lecture Notes in Computer Science, vol. 4877. Springer. Pages 197–206.

Chang, C.-C., & Lin, C.-J. 2001. *LIBSVM: a library for support vector machines.* Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chernov, S., Demartini, G., Herder, E., Kopycki, M., & Nejdl, W. 2008. Evaluating Personal Information Management Using an Activity Logs Enriched Desktop Dataset. *Pages 1–8 of: PIM '08: Workshop on Personal Information Management Workshop held at CHI '08: SIGCHI Conference on Human Factors in Computing Systems.*

Chernov, S. 2008. Task detection for activity-based desktop search. *Pages 894–894 of: SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York, NY, USA: ACM.

Chirita, P. A., Costache, S., Gaugaz, J., & Nejdl, W. 2006. Desktop Context Detection Using Implicit Feedback. *In: PIM '06: Workshop on Personal Information Management held at SIGIR '06: ACM SIGIR Conference on Research and Development in Information Retrieval.*

Choudhury, T., Borriello, G., Consolvo, S., Haehnel, D., Harrison, B., Hemingway, B., Hightower, J., Klasnja, P., Koscher, K., LaMarca, A., Landay, J. A., LeGrand, L., Lester, J., Rahimi, A., Rea, A., & Wyatt, D. 2008. The Mobile Sensing Platform: An Embedded Activity Recognition System. *Pervasive Computing*, **7**(2), 32–41. IEEE Computer Society.

Coutaz, J., Crowley, J. L., Dobson, S., & Garlan, D. 2005. Context is key. *Communications ACM*, **48**(3), 49–53. ACM.

Cselle, G., Albrecht, K., & Wattenhofer, R. 2007. BuzzTrack: topic detection and tracking in email. *Pages 190–197 of: IUI '07: Proceedings of the 12th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Czerwinski, M., Horvitz, E., & Wilhite, S. 2004. A diary study of task switching and interruptions. *Pages 175–182 of: CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* New York, NY, USA: ACM.

Dengel, A., Abecker, A., Bähr, J., Bernardi, A., Dannenmann, P., Elst, L. V., Klink, S., Maus, H., Schwarz, S., & Sintek, M. 2002. *Evolving Personal to Organizational Knowledge Spaces.* Project Proposal, DFKI GmbH Kaiserslautern, `http://www.dfki.de/epos/`.

Dey, A. K. 2000. *Providing architectural support for building context-aware applications.* Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, USA.

Dey, A. K., Abowd, G. D., & Salber, D. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, **16**(2), 97–166. L. Erlbaum Associates Inc.

Dourish, P. 2004. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, **8**(1), 19–30. Springer.

Dragunov, A. N., Dietterich, T. G., Johnsrude, K., McLaughlin, M., Li, L., & Herlocker, J. L. 2005. TaskTracer: a desktop environment to support multi-tasking knowledge workers. *Pages 75–82 of: IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Dredze, M., Lau, T., & Kushmerick, N. 2006. Automatically classifying emails into activities. *Pages 70–77 of: IUI '06: Proceedings of the 11th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Drucker, P. F. 1999. Knowledge-Worker Productivity: The Biggest Challenge. *California Management Review*, **41**(2), 79–94.

Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., & Robbins, D. C. 2003. Stuff I've seen: a system for personal information retrieval and re-use. *Pages 72–79 of: SIGIR '03: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and development in informaion retrieval.* New York, NY, USA: ACM.

Duval, E., & Hodgins, W. 2003. A LOM research agenda. *Pages 1–9 of: WWW '03: In Proceedings of the 12th International Conference on World Wide Web.* ACM.

DYONIPOS. 2006. *DYONIPOS - DYnamic ONtology based Integrated Process OPtimiSation. Website: http://www.dyonipos.at.*

Edmunds, A., & Morris, A. 2000. The problem of information overload in business organisations: a review of the literature. *International Journal of Information Management*, **20**(1), 17–28.

EL-Manzalawy, Y., & Honavar, V. 2005. *WLSVM: Integrating LibSVM into Weka Environment.* `http://www.cs.iastate.edu/~yasser/wlsvm`.

Elst, L. 2006. *Mymory - Situated Documents in Personal Information Spaces, http://www.dfki.uni-kl.de/mymory/.*

Elst, L., Aschoff, F.-R., Bernardi, A., Maus, H., & Schwarz, S. 2003. Weakly-structured Workflows for Knowledge-intensive Tasks: An Experimental Evaluation. *Pages 340–345 of: KMDAP '03: Proceedings of the Workshop Knowledge Management for Distributed Agile Processes: Models, Techniques, and Infrastructure held at 12th IEEE International Workshops on Enabling Technologies.* IEEE Computer Society.

Eraut, M. 2004. Informal learning in the workplace. *Studies in Continuing Education*, **26**(2), 247–273. Carfax Publishing.

Favela, J., Tentori, M., Castro, L. A., Gonzalez, V. M., Moran, E. B., & Martínez-García, A. I. 2007. Activity recognition for context-aware hospital applications: issues and opportunities for the deployment of pervasive networks. *Mobile Networks and Applications*, **12**(2-3), 155–171. Kluwer Academic Publishers.

Feldman, S. 2004. The high cost of not finding information. *KMWorld Magazine*, **13**(3). Available at: `http://www.kmworld.com/articles/readarticle.aspx?articleid=9534`.

Fenstermacher, K. D. 2005. *Revealed Processes in Knowledge Management.* Lecture Notes in Computer Science, vol. 3782. Springer. Pages 443–454.

Fenstermacher, K. D., & Ginsburg, M. 2002. A Lightweight Framework for Cross-Application User Monitoring. *Computer*, **35**(3), 51–59. IEEE Computer Society.

Fernandez-Garcia, N., Sauermann, L., Sanchez, L., & Bernardi, A. 2006. PIMO Population and Semantic Annotation for the Gnowsis Semantic Desktop. *Pages 1–12 of: CEUR-WS '06: Proceedings of the Semantic Desktop and Social Semantic Collaboration Workshop held at the ISWC '06: International Semantic Web Conference*, vol. 202.

Fischer, G. 2001. User Modeling in Human-Computer Interaction. *User Modeling and User-Adapted Interaction*, **11**(1-2), 65–86. Kluwer Academic Publishers.

Fuhr, N. 2005. Information Retrieval - From Information Access to Contextual Retrieval. *Pages 47–57 of: Designing Information Systems. Festschrift für Jürgen Krause.* UVK Verlagsgesellschaft.

Fürnkranz, J., & Widmer, G. 1994. Incremental Reduced Error Pruning. *Pages 70–77 of: ICML '04: Proceedings of International Conference on Machine Learning.* Morgan Kaufmann.

Gievska, S., & Sibert, J. L. 2004. A Framework for Context-Sensitive Coordination of Human Interruptions in Human-Computer Interaction. *Pages 418–425 of: User Interfaces for All.* Lecture Notes in Computer Science, vol. 3196. Springer.

Goecks, J., & Shavlik, J. 2000. Learning users' interests by unobtrusively observing their normal behavior. *Pages 129–132 of: IUI '00: Proceedings of the 5th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Granitzer, M., Rath, A. S., Kroell, M., Seifert, C., Ipsmiller, D., Devaurs, D., N., W., & Lindstaedt, S. 2009a. Machine Learning based Work Task Classification. *Journal of Digital Information Management*, **7**(5), 306–314. Digital Information Research Foundation.

Granitzer, M. 2008. *KnowMiner - Konzeption & Entwicklung eines generischen Wissenserschließungsframeworks.* VDM Verlag Dr. Müller.

Granitzer, M., Kröll, M., Seifert, C., Rath, A. S., Weber, N., Dietzel, O., & Lindstaedt, S. N. 2008. Analysis of machine learning techniques for context extraction. *Pages 233–240 of: ICDIM '08: Proceedings of International Conference on Digital Information Management.*

Granitzer, M., Granitzer, G., Tochtermann, K., Lindstaedt, S. N., Rath, A. S., & Groiß, W. 2009b. Automating Knowledge Transfer and Creation in Knowledge Intensive Business Processes. *Pages 678–686 of: Business Process Management Workshops*. Lecture Notes in Business Information Processing, vol. 17. Springer.

Grebner, O., Ong, E., Riss, U., Brunzel, M., Bernardi, A., & Roth-Berghofer, T. 2007. *NEPOMUK deliverable D3.1: task management model, `http://nepomuk.semanticdesktop.org/xwiki/bin/view/Main1/D3-1`.*

Gross, T., & Prinz, W. 2003. Awareness in context: a light-weight approach. *Pages 295–314 of: ECSCW'03: Proceedings of the Eighth Conference on European Conference on Computer Supported Cooperative Work*. Norwell, MA, USA: Kluwer Academic Publishers.

Groza, T., Handschuh, S., Moeller, K., Grimnes, G., Sauermann, L., Minack, E., Mesnage, C., Jazayeri, M., Reif, G., & Gudjónsdóttir, R. 2007. The NEPOMUK Project - On the way to the Social Semantic Desktop. *Pages 201–211 of: I-SEMANTICS '07 and I-MEDIA '07: Proceedings of International Conferences on new Media Technology and Semantic Systems*. JUCS.

Gruber, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition.*, **5**(2), 199–220. Academic Press Ltd.

Gutschmidt, A., Cap, C. H., & Nerdinger, F. W. 2008. Paving the Path to Automatic User Task Identification. *Pages 1–7 of: CSKGOI '08: Proceedings of the Workshop on Common Sense Knowledge and Goal-Oriented Interfaces*, vol. 323. CEUR-WS.org.

Herman, I., Swick, R., & Brickley, D. 2008. *Resource Description Framework (RDF), `http://www.w3.org/RDF/`.*

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. 1998. The lumiere project: bayesian user modeling for inferring the goals and needs of software users. *Pages 256–265 of: Proceedings of 14th Conference on Uncertainty in Artificial Intelligence*. San Francisco, USA: Morgan Kaufmann.

Horvitz, E., Jacobs, A., & Hovel, D. 1999. Attention-Sensitive Alerting. *Pages 305–313 of: UAI '99: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*. San Francisco, USA: Morgan Kaufmann.

Horvitz, E., Koch, P., & Apacible, J. 2004. BusyBody: creating and fielding personalized models of the cost of interruption. *Pages 507–510 of: CSCW '04: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*. New York, NY, USA: ACM.

Huynh, D., Karger, D. R., Quan, D., & Sinha, V. 2003. Haystack: a platform for creating, organizing and visualizing semistructured information. *Pages 323–323 of: IUI '03: Proceedings of the 8th International Conference on Intelligent User Interfaces*. New York, NY, USA: ACM.

Huynh, T., & Schiele, B. 2005. Analyzing features for activity recognition. *Pages 159–163 of: sOc-EUSAI '05: Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence*. New York, NY, USA: ACM.

INTELLEXT Inc. 2007. *Watson project website, `http://www.intellext.com`.*

Joachims, T., Cristianini, N., & Shawe-Taylor, J. 2001. Composite Kernels for Hypertext Categorisation. *Pages 250–257 of: ICML '01: Proceedings of the 18th International Conference on Machine Learning.* San Francisco, CA, USA: Morgan Kaufmann.

Jones, W. 2007. *Keeping Found Things Found: The Study and Practice of Personal Information Management.* Academic Press.

Jones, W., Klasnja, P., Civan, A., & Adcock, M. L. 2008. The personal project planner: planning to organize personal information. *Pages 681–684 of: CHI '08: Proceeding of the SIGCHI Conference on Human Factors in Computing Systems.* New York, NY, USA: ACM.

Kellar, M., & Watters, C. 2006. Using web browser interactions to predict task. *Pages 843–844 of: WWW '06: Proceedings of the 15th International Conference on World Wide Web.* New York, NY, USA: ACM.

Kellar, M., Watters, C., & Shepherd, M. 2007. A field study characterizing web-based information-seeking tasks. *Journal of the American Society for Information Science and Technology*, **58**(7), 999–1018. John Wiley & Sons, Inc.

Kersten, M., & Murphy, G. C. 2006. Using task context to improve programmer productivity. *Pages 1–11 of: SIGSOFT '06/FSE-14: Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering.* New York, NY, USA: ACM.

Kiritchenko, S., & Matwin, S. 2001. Email classification with co-training. *Pages 1–10 of: CASCON '01: Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research.* IBM Press.

Kleek, M., & Shrobe, H. E. 2007. A Practical Activity Capture Framework for Personal, Lifetime User Modeling. *Pages 298–302 of: UM '07: Proceedings of the 11th International Conference on User Modeling.* Springer.

Kleek, M., Bernstein, M., Karger, D. R., & schraefel, m. 2007. Gui – phooey!: the case for text input. *Pages 193–202 of: UIST '07: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology.* New York, NY, USA: ACM.

KnowSe. 2009. *KnowSe - Knowledge Services Framework. Know-Center GmbH.. Website:* `http://en.know-center.at/forschung/knowledge_services`.

Kompacher, G. 2008. *Erstellung von graphischen Benutzeroberflächen für ein Work-Integrated Learning Environment basierend auf Eclipse RCP.* Master Project. Knowledge Management Institute, Graz University of Technology, Austria.

Kompacher, G. 2010. *Identifikation von relevanten Konzepten in einem Ontologie-basierten Kontextmodel.* Master Thesis. Knowledge Management Institute, Graz University of Technology, Austria (in progress).

Kröll, M., Rath, A. S., Weber, N., Lindstaedt, S. N., & Granitzer, M. 2007. Task Instance Classification via Graph Kernels. *Pages 1–4 of: MLG '07: The 5th International Workshop on Mining and Learning with Graphs.*

Kröll, M., Rath, A. S., Granitzer, M., Lindstaedt, S. N., & Tochtermann, K. 2006. Contextual Retrieval in Knowledge Intensive Business Environments. *Pages 115–119 of: LWA '06: Lernen - Wissensentdeckung - Adaptivität.* Hildesheimer Informatik-Berichte, vol. 1. University of Hildesheim, Institute of Computer Science.

Kushmerick, N., & Lau, T. 2005. Automated email activity management: an unsupervised learning approach. *Pages 67–74 of: IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Lansdale, M. W. 1988. The psychology of personal information management. *Applied Ergonomics*, **19**(1), 55–66. Elsevier Science B.V.

Leitner, M. 2007. *Analyse und Vergleich von Werkzeugen zur Kontext-Erkennung.* Bachelor Thesis. Knowledge Management Institute, Graz University of Technology, Austria.

Leont'ev, A. N. 1978. *Activity, Consciousness, and Personality.* Prentice Hall.

Ley, T., Ulbrich, A., Scheir, P., Lindstaedt, S. N., Kump, B., & Albert, D. 2008. Modelling Competencies for Supporting Work-integrated Learning in Knowledge Work. *Journal of Knowledge Management*, **12**(6), 31–47. Graz University of Technology.

Lindstaedt, S. N., Ley, T., & Mayer, H. 2005. Integrating Working and Learning with APOSDLE. *Pages 1–5 of: Proceedings of the 11th Business Meeting of Forum Neue Medien.* Forum Neue Medien.

Lindstaedt, S. N., Ley, T., Scheir, P., & Ulbrich, A. 2008a. Applying Scruffy Methods to Enable Work-integrated Learning. *Upgrade: The European Journal of the Informatics Professional*, **9**(3), 44–50. Novática.

Lindstaedt, S. N., Scheir, P., Lokaiczyk, R., Kump, B., Beham, G., & Pammer, V. 2008b. Knowledge Services for Work-Integrated Learning. *Pages 234–244 of: EC-TEL '08: Proceedings of the 3rd European Conference on Technology Enhanced Learning.* Springer.

Lindstaedt, S. N., Beham, G., Kump, B., & Ley, T. 2009a. Getting to Know Your User - Unobtrusive User Model Maintenance within Work-Integrated Learning Environments. *Pages 73–87 of: EC-TEL '09: Proceedings of the 4th European Conference on Technology Enhanced Learning.* Lecture Notes in Computer Science, vol. 5794. Springer.

Lindstaedt, S. N., Aehnelt, M., & de Hoog, R. 2009b. Supporting the Learning Dimension of Knowledge Work. *Pages 639–644 of: EC-TEL '09: Proceedings of the 4th European Conference on Technology Enhanced Learning.* Lecture Notes in Computer Science, vol. 5794. Springer.

Lokaiczyk, R., & Goertz, M. 2009. Extending Low Level Context Events by Data Aggregation. *Pages 118–125 of: I-KNOW '09: Proceedings of the 9th International Conference on Knowledge Management.*

Lokaiczyk, R. 2008. On Resource Acquisition in Adaptive Workplace-Embedded E-Learning Environments. *International Journal Advanced Corporate Learning*, **1**(1), 23–26. International Association of Online Engineering.

Lokaiczyk, R., Faatz, A., Beckhaus, A., & Görtz, M. 2007. Enhancing Just-in-Time E-Learning Through Machine Learning on Desktop Context Sensors. *Pages 330–341 of: CONTEXT '07: Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context*. Lecture Notes in Computer Science, vol. 4635. Springer.

Lowd, D., & Kushmerick, N. 2009. Using salience to segment desktop activity into projects. *Pages 463–468 of: IUI '09: Proceedings of the 14th International Conference on Intelligent User Interfaces*. ACM.

Maier, R. 2005. Modeling Knowledge Work for the Design of Knowledge Infrastructures. *Journal of Universal Computer Science*, **11**(4), 429–451.

Maier, R., & Sametinger, J. 2007. A Top-Level Ontology for Smart Document Access. *Pages 153–164 of: ICKM '07, Proceedings of the 4th International Conference on Knowledge Management*. Singapore: World Scientific.

Maier, R., & Schmidt, A. 2007. Characterizing Knowledge Maturing: A Conceptual Process Model for Integrating E-Learning and Knowledge Management. *Pages 325–334 of: WM '07: Proceedings of the 4th Conference Professional Knowledge Management - Experiences and Visions*, vol. 1. GITO GmbH.

Manning, C. D., & Schutze, H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Maruster, L., Weijters, A. J. M. M., Aalst, W. M. P. v. d., & Bosch, A. v. d. 2002. Process Mining: Discovering Direct Successors in Process Logs. *Pages 364–373 of: DS '02: Proceedings of the 5th International Conference on Discovery Science*. Lecture Notes in Computer Science, vol. 2534. Springer.

Maurer, H., & Tochtermann, K. 2002. On a New Powerful Model for Knowledge Management and its Applications. *Journal of Universal Computer Science*, **8**(1), 85–96. JUCS.

McArthur, R., & Bruza, P. 2003. Discovery of implicit and explicit connections between people using email utterance. *Pages 21–40 of: ECSCW'03: Proceedings of the 8th Conference on European Conference on Computer Supported Cooperative Work*. Norwell, MA, USA: Kluwer Academic Publishers.

McFarlane, D. 2002. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. *Human-Computer Interaction*, **17**(1), 63–139. L. Erlbaum Associates Inc.

Medeiros, A. d., Aalst, W. M. P. v. d., & Weijters, A. J. M. M. 2003. Workflow Mining: Current Status and Future Directions. *Pages 389–406 of: On The Move to Meaningful Internet Systems*. Lecture Notes in Computer Science, vol. 2888. Springer.

Microsoft. 2009. *Microsoft Active Accessibility, `http://msdn.microsoft.com/en-us/accessibility`*.

MIT. 2009. *SIMILE Widgets, Massachusetts Institute of Technology, `http://www.`*

`simile-widgets.org`.

Mitchell, T. M., Wang, S. H., Huang, Y., & Cheyer, A. 2006. Extracting knowledge about users' activities from raw workstation contents. *Pages 181–186 of: AAAI '06: Proceedings of the 21st National Conference on Artificial Intelligence.* AAAI Press.

Mock, K. 2001. An experimental framework for email categorization and management. *Pages 392–393 of: SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York, NY, USA: ACM.

Nair, R., Voida, S., & Mynatt, E. D. 2005. Frequency-based detection of task switches. *Pages 94–99 of: HCI '05: Proceedings of the Human Computer Interaction Conference*, vol. 2.

Najjar, J., Wolpers, M., & Duval, E. 2006. Attention Metadata: Collection and Management. *Pages 1–4 of: Proceedings of Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection held at WWW '06: World Wide Web Conference.*

Ochoa, X., & Duval, E. 2006. Use of contextualized attention metadata for ranking and recommending learning objects. *Pages 9–16 of: CAMA '06: Proceedings of the 1st International Workshop on Contextualized Attention Metadata: Collecting, Managing and Exploiting of Rich Usage Information.* New York, NY, USA: ACM.

Oliver, N., Smith, G., Thakkar, C., & Surendran, A. C. 2006. SWISH: semantic analysis of window titles and switching history. *Pages 194–201 of: IUI '06: Proceedings of the 11th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

OpenAnzo.org. 2008. *Open Anzo - an open source RDF store, query engine and related middleware for the development of semantic applications.*, `http://www.openanzo.org`.

OSGI2008. 2008. *Open Service Gateway Initiative*, `http://www.osgi.org`.

Ötztürck, P., & Amodt, A. 1997. Towards a model of context for case-based diagnostic problem solving. *Pages 198–208 of: inContext '97: Proceedings of the Interdisciplinary Conference on Modeling and Using Context.*

OWL. 2007. *Web Ontology Language (OWL)*, `http://www.w3.org/2004/OWL/`.

Pedersen, E. R., & McDonald, D. W. 2008. Relating documents via user activity: the missing link. *Pages 389–392 of: IUI '08: Proceedings of the 13th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Philipose, M., Fishkin, K. P., Perkowitz, M., Patterson, D. J., Fox, D., Kautz, H., & Hahnel, D. 2004. Inferring Activities from Interactions with Objects. *IEEE Pervasive Computing*, **3**(4), 50–57. IEEE Computer Society.

Pichler, T. 2007. *Automatische Aufbereitung von E-Mail Verkehr zur Wissenskonservierung.* Bachelor Thesis. Knowledge Management Institute, Graz University of Technology, Austria.

Pichler, T. 2010. *Visualisierung von User-Kontext-Daten.* Master Thesis. Knowledge Management Institute, Graz University of Technology, Austria (in progress).

Porter, M. F. 1997. An algorithm for suffix stripping. *Readings in information retrieval*, 313–316. Morgan Kaufmann.

Porter, M. 2009. *Snowball project website,* `http://snowball.tartarus.org`.

ProM Framework. 2007. *The ProM Framework,* `http://prom.sf.net/`.

Protégé. 2009. *The Protégé Ontology Editor and Knowledge Acquisition System,* `http://protege.stanford.edu`.

Prud'Hommeaux, E., & Seaborne, A. 2008. *SPARQL Query Language for RDF,* `http://www.w3.org/TR/rdf-sparql-query/`. World Wide Web Consortium.

Rath, A. S., Kröll, M., Andrews, K., Lindstaedt, S., Granitzer, M., & Tochtermann, K. 2006. Synergizing Standard and Ad-Hoc Processes. *Pages 267–278 of: Practical Aspects of Knowledge Management.* Lecture Notes in Computer Science, vol. 4333. Springer.

Rath, A. S., Weber, N., Kröll, M., Granitzer, M., Dietzel, O., & Lindstaedt, S. 2008. Context-Aware Knowledge Services. *In: PIM '08: Workshop on Personal Information Management Workshop held at CHI '08: SIGCHI Conference on Human Factors in Computing Systems.*

Rath, A. S., Devaurs, D., & Lindstaedt, S. N. 2009a. Detecting Real User Tasks by Training on Laboratory Contextual Attention Metadata. *Pages 1–9 of: EUAM '09: Exploitation of Usage and Attention Metadata held at Informatik '09.*

Rath, A. S. 2007. A Low-Level Based Task and Process Support Approach For Knowledge-Intensive Business Environments. *Pages 35–42 of: DECIS '07: Doctoral Consortium held at ICEIS' 07: 5th International Conference on Enterprise Information System.* INSTICC Press.

Rath, A. S., Kröll, M., Lindstaedt, S., & Granitzer, M. 2007. Low-Level Event Relationship Discovery for Knowledge Work Support. *In: ProKW '07: Proceedings of the 4th Conference on Professional Knowledge Management.* Productive Knowledge Work - Management and Technological Challenges. GITO GmbH.

Rath, A. S., Devaurs, D., & Lindstaedt, S. N. 2009b. Contextualized Knowledge Services for Personalized Learner Support. *In: EC-TEL '09: Fourth European Conference on Technology Enhanced Learning.* Lecture Notes in Computer Science. Springer.

Rath, A. S., Devaurs, D., & Lindstaedt, S. N. 2009c. KnowSe: Fostering User Interaction Context Awareness. *Pages 9–10 of: ECSCW '09: Supplementary Proceedings of the 11th European Conference on Computer Supported Cooperative Work.*

Rath, A. S., Devaurs, D., & Lindstaedt, S. N. 2009d. UICO: an ontology-based user interaction context model for automatic task detection on the computer desktop. *Pages 1–10 of: CIAO '09: Proceedings of the 1st Workshop on Context, Information and Ontologies.* New York, NY, USA: ACM.

RaVis. 2009. *RaVis Relational Analysis Components, http://code.google.com/p/birdeye/wiki/RaVis.*

Rechberger, A. 2007. *Sensoren zur Kontexerkennung - Sensoren für Mozilla Thunderbird und*

Novell Groupwise. Bachelor Thesis. Knowledge Management Institute, Graz University of Technology, Austria.

Resanovic, D. 2008. *Service Integration Layer für Aposdle und Dyonipos Services im Rahmen des KnowSe Projektes.* Master Project. Knowledge Management Institute, Graz University of Technology, Austria.

Rhodes, B. J. 2000. *Just-In-Time Information Retrieval.* Ph.D. thesis, MIT Media Laboratory, Cambridge, MA.

Rijsbergen, C. 1979. *Information retrieval.* 2 edn. London: Butterworths.

Riss, U., Rickayzen, A., Maus, H., & Aalst, W. M. P. v. d. 2005. Challenges for Business Process and Task Management. *Journal of Universal Knowledge Management*, **0**(2), 77–100.

Riss, U. V., & Grebner, O. 2006. Service-Oriented Task Management. *Pages 354–358 of: I-KNOW '06: Proceedings the 6th International Conference on Knowledge Management.* JUCS.

Sachs, L., & Hedderich, J. 2006. *Angewandte Statistik. Methodensammlung mit R.* 12 edn. Springer.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. 1998. A Bayesian Approach to Filtering Junk E-Mail. *In: Learning for Text Categorization: Papers from the 1998 Workshop.* Madison, Wisconsin: AAAI Technical Report WS-98-05.

Salton, G., & McGill, M. J. 1986. *Introduction to Modern Information Retrieval.* New York, NY, USA: McGraw-Hill, Inc.

Sauermann, L. 2003. *The Gnowsis - Using Semantic Web Technologies to build a Semantic Desktop.* Diploma thesis, Technical University of Vienna.

Sauermann, L., Bernardi, A., & Dengel, A. 2005. Overview and Outlook on the Semantic Desktop. *Pages 1–18 of: Proceedings of the 1st Workshop on The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure held at ISWC '05: International Semantic Web Conference.*

Sauermann, L., Dengel, A., v. Elst, L., Lauer, A., Maus, H., & Schwarz, S. 2006a. Personalization in the EPOS project. *Pages 42–52 of: Proceedings of the Semantic Web Personalization Workshop held at ESWC '06: European Semantic Web Conference.*

Sauermann, L., Grimnes, G. A., Kiesel, M., Fluit, C., Maus, H., Heim, D., Nadeem, D., Horak, B., & Dengel, A. 2006b. Semantic Desktop 2.0: The Gnowsis Experience. *Pages 887–900 of: ISWC '06: Proceedings of the 5th International Semantic Web Conference.* Lecture Notes in Computer Science, vol. 4273. Springer.

Sauermann, L. 2005. The Gnowsis Semantic Desktop for Information Integration. *Pages 39–42 of: IOA '05: Workshop on Intelligent Office Appliances held at WM '05: Professional Knowledge Management Conference.* Lecture Notes in Computer Science. Springer.

Sauermann, L., Elst, L., & Dengel, A. 2007. PIMO - A Framework for Representing Personal

Information Models. *Pages 270–277 of: Proceedings of I-MEDIA '07 and I-SEMANTICS '07 International Conferenceson New Media Technology and Semantic Systems as part of TRIPLE-I 2007.* JUCS.

Schilit, B., Adams, N., & Want, R. 1994. Context-Aware Computing Applications. *Pages 85–90 of: Proceedings of the Workshop on Mobile Computing Systems and Applications.* Santa Cruz, CA, US: IEEE Computer Society.

Schmidt, A. 2005a. Bridging the Gap Between Knowledge Management and E-Learning with Context-Aware Corporate Learning. *Pages 203–213 of: LOKMOL '05: Proceedings of Learner-Oriented Knowledge Management and KM-Oriented E-Learning Workshop held at WM '05: Professional Knowledge Management Conference.* Lecture Notes in Computer Science, vol. 3782. Springer.

Schmidt, A. 2005b. Knowledge Maturing and the Continuity of Context as a Unifying Concept for Knowledge Management and E-Learning. *Pages 424–431 of: I-KNOW '05: Proceedings of the International Conference on Knowledge Management.* JUCS.

Schmidt, A. 2007. Impact of Context-Awareness on the Architecture of E-Learning Solutions. *Pages 306–319 of: Architecture Solutions for E-Learning Systems.* IGI Publishing.

Schmidt, K. 2002. The Problem with 'Awareness': Introductory Remarks on 'Awareness in CSCW'. *Computer Supported Cooperative Work*, **11**(3), 285–298. Kluwer Academic Publishers.

Schreiber, G., Akkermans, H., Anjewierden, A., Dehoog, R., Shadbolt, N., Vandevelde, W., & Wielinga, B. 1999. *Knowledge Engineering and Management: The CommonKADS Methodology.* The MIT Press.

Schwarz, S., Abecker, A., Maus, H., & Sintek, M. 2001. Anforderungen an die Workflow-Unterstützung für wissensintensive Geschäftsprozesse. *Pages 11–30 of: Proceedings of Geschäftsprozessorientiertes Wissensmanagement Workshop held at WM '01: Professional Knowledge Management Conference.* Baden-Baden, Germany: DFKI GmbH.

Schwarz, S. 2006. A Context Model for Personal Knowledge Management Applications. *Pages 18–33 of: MRC '05: Proceedings of Modeling and Retrieval of Context.* Lecture Notes in Computer Science, vol. 3946. Springer.

Segal, R., Crawford, J., Kephart, J. O., & Leiba, B. 2004. SpamGuru: An Enterprise Anti-Spam Filtering System. *Pages 1–7 of: CEAS '04: Proceedings of the First Conference on E-mail and Anti-Spam.*

Shen, J. 2009. *Activity Recognition in Desktop Environments.* Ph.D. thesis, Oregon State University.

Shen, J., & Dietterich, T. G. 2007. Active EM to reduce noise in activity recognition. *Pages 132–140 of: IUI '07: Proceedings of the 12th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Shen, J., Li, L., Dietterich, T. G., & Herlocker, J. L. 2006. A hybrid learning system for recognizing user tasks from desktop activities and email messages. *Pages 86–92 of: IUI '06: Proceedings of the 11th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Shen, J., Li, L., & Dietterich, T. G. 2007. Real-Time Detection of Task Switches of Desktop Users. *Pages 2868–2873 of: IJCAI '07: Proceedings of International Joint Conference on Artificial Intelligence.*

Shen, J., Irvine, J., Bao, X., Goodman, M., Kolibaba, S., Tran, A., Carl, F., Kirschner, B., Stumpf, S., & Dietterich, T. G. 2009. Detecting and correcting user activity switches: algorithms and interfaces. *Pages 117–126 of: IUI '09: Proceedings of the 13th International Conference on Intelligent User Interfaces.* New York, NY, USA: ACM.

Sifry, D., Marks, K., & Celik, T. 2007. *AttentionXML specifications, `http://developers.technorati.com/wiki/attentionxml`.*

Sintek, M., v. Elst, L., Grimnes, G., Scerri, S., & Handschuh, S. 2007. Knowledge Representation for the Distributed, Social Semantic Web: Named Graphs, Graph Roles and Views in NRL. *Pages 1–14 of: WoMO '07: Proceedings of Workshop on Modular Ontologies held at K-CAP '07: Fourth International Conference on Knowledge Capture.*

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, **5**(2), 51–53. Elsevier Science B.V.

Smith, P. J. 2003. Workplace Learning and Flexible Delivery. *Review of Educational Research*, **73**(1), 53–88.

Strang, T., & Linnhoff-Popien, C. 2004. A Context Modeling Survey. *Pages 1–8 of: Proceedings of Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp '04: International Conference on Ubiquitous Computing.*

Teevan, J. 2008. How people recall, recognize, and reuse search results. *ACM Transactions on Information Systems*, **26**(4), 1–27. ACM.

Teevan, J., Jones, W., & Capra, R. 2008. Personal information management (PIM) 2008. *SIGIR Forum*, **42**(2), 96–103. ACM.

The Apache Software Foundation. 2009 (08). *The Apache Commons Mathematics Library, Version 2.1, http://commons.apache.org/math/.*

Tochtermann, K., Reisinger, D., Granitzer, M., & Lindstaedt, S. 2006. Integrating Ad Hoc Processes and Standard Processes in Public Administrations. *In: eGov '06: Proceedings of the OCG eGovernment Conference.* OCG, vol. 203.

Tochtermann, K. 2003. Personalization in Knowledge Management. *Pages 29–41 of: Metainformatics.* Lecture Notes in Computer Science, vol. 2641. Springer.

Truong, H.-L., & Dustdar, S. 2009. A Survey on Context-aware Web Service Systems. *International Journal of Web Information Systems*, **5**(1), 5–31. Emerald Group Publishing Limited.

Tuffield, M. M., Loizou, A., & Dupplaw, D. 2006. The semantic logger: supporting service building from personal context. *Pages 55–64 of: CARPE '06: Proceedings of the 3rd ACM Workshop on Continuous Archival and Retrival of Personal Experiences*. New York, NY, USA: ACM.

Ulbrich, A., Scheir, P., Lindstaedt, S. N., & Görtz, M. 2006. A Context-Model for Supporting Work-Integrated Learning. *Pages 525–530 of: Innovative Approaches for Learning and Knowledge Sharing*. Springer.

Wechtitsch, S. 2008. *Handling User Interruption and Notification in KnowSe*. Master Project. Knowledge Management Institute, Graz University of Technology, Austria.

Weijters, A. J. M. M., & Aalst, W. M. P. v. d. 2001. Process Mining: Discovering Workflow Models from Event-Based Data. *Pages 283–290 of: BNAIC '01: Proceedings 13th Belgium-Netherlands Conference on Artificial Intelligence.*

Wen, L., Wang, J., Aalst, W. M. P. v. d., Huang, B., & Sun, J. 2008. A novel approach for process mining based on event types. *Journal of Intelligent Information Systems*, **32**(2), 163–190. Springer.

Whittaker, S., & Sidner, C. 1996. Email overload: exploring personal information management of email. *Pages 276–283 of: CHI '96: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM.

Witten, I. H., & Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques*. 2 edn. San Francisco, USA: Morgan Kaufmann.

Wolpers, M., Najjar, J., Verbert, K., & Duval, E. 2007. Tracking Actual Usage: the Attention Metadata Approach. *Educational Technology & Society*, **10**(3), 106–121.

Xiao, H., & Cruz, I. F. 2005. A multi-ontology approach for personal information management. *In: Proceedings of The Semantic Desktop Workshop - Next Generation Personal Information Management and Collaboration Infrastructure held at ISWC '05: International Semantic Web Conference.*

Yang, Y., & Liu, X. 1999. A re-examination of text categorization methods. *Pages 42–49 of: SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM.

Yang, Y., & Pedersen, J. O. 1997. A Comparative Study on Feature Selection in Text Categorization. *Pages 412–420 of: ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann.

Zhao, Y., Karypis, G., & Fayyad, U. 2005. Hierarchical Clustering Algorithms for Document Datasets. *Data Mining and Knowledge Discovery*, **10**(2), 141–168. Kluwer Academic Publishers.

Zimmermann, A., Lorenz, A., & Oppermann, R. 2007. An Operational Definition of Context. *Pages 558–571 of: CONTEXT '07: 6th International and Interdisciplinary Conference on*

Modeling and Using Context. Lecture Notes in Computer Science, vol. 4635. Springer.

Zimmermann, A., Specht, M., & Lorenz, A. 2005. Personalization and Context Management. *User Modeling and User-Adapted Interaction*, **15**(3-4), 275–302. Springer.

Appendix

8.4 Task Models of the Laboratory Experiment 1

8.4.1 [Task 1] Filling in the official journey form

Task Model (Description):
The employee fills in the details about the planned journey in the official journey form, prints it and gives it to her division manager.

Task Instance (Task Example):
Suppose you are a researcher named Bill Adams, who is working in the Knowledge Services division on the KnowSe project. You are traveling to the Computer Human interaction (CHI 2009) conference in Boston, USA, with a colleague of yours. The conference starts at the 4th April 2009 and ends at the 9th April. Your secretary has already compiled some information about your trip. Here are the details:

- Project *KnowSe*
- Your division: Knowledge Services
- Your name: Bill Adams
- Your colleague's name: Mary Jones
- Flight 3rd April from Graz (Austria) to Frankfurt (Germany) at 06:00
- Flight 3rd April from Frankfurt (Germany) to Boston (USA) at 12:25
- Flight 10th April from Boston (USA) to Frankfurt (Germany) at 21:40
- Flight 11th April from Frankfurt (Germany) to Graz (Austria) at 09:15
- CHI 2009 conference fees are 1000 Euros per person
- Hotel costs are 500 Euros
- Traveling costs are 950 Euros per person (flight and bus)
- All expenses will have to be payed in advance by yourself and will be refunded after the journey.

8.4.2 [Task 2] Filling in the cost recompense form for the official journal

Task Model (Description):
The employee fills in the details about the journey and its expenses in the official journey form,

prints it and gives it to her division manager.

Task Instance (Task Example):
Suppose you are a researcher named Bill Adams, who is working in the Knowledge Services division on the KnowSe project. You were traveling to the EC-TEL 2007 - Second European Conference on Technology Enhanced Learning conference with a colleague of yours, named Mary Jones. The conference was located in Crete (Greece) and started at the 17th September 2007 and ended at the 20th September 2007. Your secretary has already compiled some information about your trip. Here are the details:

- Project *KnowSe*
- Your division: Knowledge Services
- Your name: Bill Adams
- Your colleague's name: Mary Jones
- Flight 16th September from Graz (Austria) to Vienna (Austria) from 06:00 to 06:50
- Flight 16th September from Vienna (Austria) to Crete (Greece) from 11:00 to 13:30
- Flight 20th September from Crete (Greece) to Vienna (Austria) from 15:00 to 17:30
- Flight 20th September from Vienna (Austria) to Graz (Austria) from 18:15 to 19:05
- EC-TEL 2007 conference fees were 500 Euros per person
- Traveling costs are 700 Euros (flight and bus)
- Hotel costs were 500 Euros
- All expenses have been payed in advance by yourself and will be refunded.

8.4.3 [Task 3] Creating and handing in an application for leave

Task Model (Description):
The employee fills in the details about the planned vacation, prints it and gives it to the leader of the employee's division.

Task Instance (Task Example):
Suppose you are a researcher named Bill Adams, who is working in the Knowledge Services division on the KnowSe project. You would like to take some time off and hence apply for a vacation from the 10.11.2008 until the 18.11.2008. For this you have to fill in the official application for leave form, print it and hand it to your supervisor.

8.4.4 [Task 4] Planning an official journey

Task Model (Description):
The employee organizes an official journey. This means to plan for transportation, accommodation and organizational things at the location if applicable. At the end the employee fills in the details and expenses of the journey in the official journey form, prints it and gives it to the supervisor of the employee's division.

Task Instance (Task Example):
Suppose you are a researcher named Bill Adams, who is working in the Knowledge Services division on the KnowSe project. Your plan is to participate in the annual project steering

committee meeting in Bern, Switzerland. The meeting takes place from the 25.11.2008 until the 27.11.2008 and is located in the HOTEL AMBASSADOR SPA (http://www.ambassadorbern.ch), Seftigenstrasse 99, 3007, Bern, Switzerland. Unfortunately the hotel has no room available any more because of a big congress at the same time such that you have to find your own accommodation. The accommodation expenses should be not more than 800 Euros for the whole stay.

8.4.5 [Task 5] Organization of a project meeting

Task Model (Description):
The employee organizes a project meeting including the creation of the agenda, sending emails to the meeting participants, the reservation of the room and the projector, creating the presentation and so on.

Task Instance (Task Example):
Suppose you are a researcher named Bill Adams, who is working in the Knowledge Services division on the KnowSe project. You are planning to introduce a new idea for the KnowSe project and hence organizing a project meeting in which you would like to present your ideas. The organization of the project meeting involves finding an appropriate date for the KnowSe project members and a suitable place for the project meeting. You also need to place a reservation for the project meeting room and the projector. Furthermore you have to compile an agenda and distribute it to the project members before the meeting. For presenting your ideas create a short Microsoft PowerPoint presentation.

8.5 Task Models of the Laboratory Experiment 2

8.5.1 [Task 1] Register for an examination

Task Model (Description):
The registration for an exam is a task in which the student has to sign in to the TUGonline system and to register himself to a particular examination. Registration is required for doing and passing an exam at the university.

Task Instance (Task Standard):
Suppose you are a student named Georg Kompacher studying *Software Development and Business Management* at Graz University of Technology in Austria. You want to register for a specific examination named *“Verifikation und Testen”* taking place on the 15th May 2009. To register you have to use the TUGonline system. Use the Microsoft Internet Explorer and visit the web site `http://online.tugraz.at` and sign in with the following account: user name: `<wttestaccountuser>` and password: `<wttestpassword>`. Search for the examination for *“Verifikation und Testen”* and register for that exam.

Task Instance (Task Personal):
Open the browser of your choice. Search for an examination at the Graz University of Technology

you want to participate and register for that exam using the TUGonline system.

8.5.2 [Task 2] Finding course dates

Task Model (Description):
Find the dates of a lecture and copy them to the clipboard.

Task Instance (Task Standard):
Open the Microsoft Internet Explorer and visit the web site `http://online.tugraz.at`. Find out when the classes of the course *"Wissenstechnologie"* of Graz University of Technology are held in this term. Mark and copy the timetable to the clipboard.

Task Instance (Task Personal):
Open the Internet Explorer and find out, when the classes of a lecture of your choice are held. Mark and copy the timetable to the clipboard.

8.5.3 [Task 3] Reserve a book in the university's library

Task Model (Description):
Finding a book in the university's library is mandatory for most students to broad their horizon. A reservation is necessary if the searched book has already been lent.

Task Instance (Task Standard):
Suppose you are a student named Georg Kompacher studying *Software Development and Business Management* at Graz University of Technology. You are interested in the domain of XML and you want to get some further readings to that topic. Use the Microsoft Internet Explorer and visit the web site `http://castor.tugraz.at` and search for the book *"XML Bible"* written by Elliotte Rusty Harold and reserve that book for you. If the book is already lent, prebook the book for you. If you get asked for a user account, sign in with the following: user name: `<wttestaccountuser>` and password: `<wttestpassword>`.

Task Instance (Task Personal):
Open the browser of your choice. Search for a book to a topic of a course you are currently taking and reserve it at your university's library. Prebook the book for loan.

8.5.4 [Task 4] Course Registration

Task Model (Description):
The registration to a course is one of the most used tasks for students. They have to be registered to get the permission to make an exam. For registration the authentication of a student to TUGonline is necessary.

Task Instance (Task Standard):
Suppose you are a student named Georg Kompacher studying *Software Development and Business Management* at Graz University of Technology. You want to register for a specific

course named *"Softwaretechnologie Tools"*. To register to that course you have to use the TUGonline system. Use the Internet Explorer and visit the web site `http://online.tugraz.at` and sign in with the following account: user name: `<wttestaccountuser>` and password: `<wttestpassword>`. Search for the course *"Softwaretechnologie Tools"* and register for that course.

Task Instance (Task Personal):
Use the Microsoft Internet Explorer and visit the web site `http://online.tugraz.at`. Search for a course of this semester you are interested in and register for that course.

8.5.5 [Task 5] Algorithm programming

Task Model (Description):
Solve a given computational problem programmatically. You can use your favorite development tools.

Task Instance (Task Standard):
Open your favorite Java IDE/editor and write a small program which takes an integer and calculates the factorial of this number. Test your program with the input and print out the result in the console.

Task Instance (Task Personal):
Choose a program language of your choice and write a small program which calculates the factorial of a given number. Test your program with a number of your choice and display the result on the display.

8.5.6 [Task 6] Prepare a scientific talk

Task Model (Description):
Presenting a presentation about a scientific paper has to be done by all students. In this task a presentation about a scientific topic has to be prepared.

Task Instance (Task Standard):
Go to the web site `http://kmi.tugraz.at/blogs/wissenstechnologie/` and search for the slides of the lecture 2008/09. Search for the lecture where the topic "Semantic Web" was covered. Create a Microsoft Power Point presentation about this with at least 5 slides.

Task Instance (Task Personal):
Prepare a scientific presentation about a topic of your choice regarding a lecture you are visiting this year. The presentation shall have at least 5 slides.

8.5.7 [Task 7] Plan a study trip

Task Model (Description):
Before visiting a conference a student has to plan the trip very carefully. This means to plan

transportation, accommodation and organizational things at the location if applicable.

Task Instance (Task Standard):
Suppose you are a student named Max Mustermann studying *Software Development and Business Management* at Graz University of Technology. Your plan is to participate in the annual *European Students Conference* in Berlin, Germany. The conference takes place from the 07.04.2009 until the 09.04.2009 and is located in the Berliner Congress Center (`http://www.bcc-berlin.de`), Alexanderstr. 11, 10178 Berlin, Germany. For your accommodation choose a hotel that costs less than € 60 per day. Choose the aircraft as means of transportation. Plan the complete trip for less than € 400.

Task Instance (Task Personal):
You are participating in a 2 day lasting congress in Vienna (7th and 8th of April 2009). Plan a study trip including the outward and the return trip as well as an accommodation that is as cheap as possible.

8.6 Task Models of the Laboratory Experiment 3

This section gives details about the studied tasks and the descriptions that were handed to the experiment's participants. Only the *Task Instances* descriptions that are described bellow were given to the participants. The *Task Model* descriptions, which were taken from the *CommonKADS* book [Schreiber *et al.*, 1999], are also listed bellow to allow the reader to understand the category of the types of analytic and synthetic tasks better. For a detailed description and further discussions about the *CommonKADS* task classification scheme it is referred here to the *CommonKADS* book [Schreiber *et al.*, 1999].

8.6.1 [Task Analytic 1] Task Classify

Duration: max. 5 min.

Task Model (Description):
In classification, an object needs to be characterized in terms of the class to which it belongs.

Task Instance:
The following list contains 10 well-known computer specific terms. Please classify them into the following categorization schema: Hardware (with the sub-categories input device and non-input device) and Software (with the sub-categories game, office application and operating system) Your classification schema shall be stored on the computer. Choose a program of your choice and save the result on the computer.

Terms to categorize: Ahead Nero 9.0, Adobe Acrobat 9.0 Professional, Adobe Photoshop CS4, Call of Duty 5 - World at War, Intel Core 2 Duo E8400 (C0), 2x 3.0GHz, Logitech Classic Keyboard 200, Logitech MX 518 Optical Gaming Mouse Refresh, Microsoft Office 2007

Professional, Microsoft Windows Vista Business 32Bit, Seagate Barracuda 7200.11 1500GB, SATA II

8.6.2 [Task Analytic 2] Task Diagnose

Duration: max. 15 min.

Task Model (Description):
Diagnosis differs from classification in the sense that the desired output is a malfunction of the system.

Task Instance:
Finding the origin/origins of the malfunction in a program: Where is/are the malfunction/s in the program. Freely choose and use the tools you require to find it/them. You find the source code of the program in the "`task_diagnose_program_file_shop.doc`" file, the "`task_diagnose_program_file_customer.doc`" file and the "`task_diagnose_program_file_item.doc`" file on your desktop. Save your diagnosis result on the computer. You do not have to fix the malfunction/s. You just have to find them. Save your results on the computer.

8.6.2.1 Experiment Material for Task A2 Diagnose

8.6.2.1.1 class Customer

```
public class Customer {
    private static int COUNTER = 0;

    private final int id;
    private String firstName;
    private String lastName;

    Customer(String firstName, String lastName) {
        this.id = ++COUNTER;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public int getId() {
        return id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
```

```
public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}
```

8.6.2.1.2 class Item

```
public class Item {

private String name;
private double price;
private int quantity;

Item(String name, double price, int quantity) {
    this.name = name;
    this.price = price;
    this.quantity = quantity;
}

public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}

public String getName() {
    return name;
}

public int getQuantity() {
    return quantity;
}

public int decreaseQuantity(int quantity) {
    this.quantity -= quantity;
    return this.quantity;
}

public int increaseQuantity(int quantity) {
    this.quantity += quantity;
    return this.quantity;
}
}
```

8.6.2.1.3 class Shop

```
public class Shop {

    List<Customer> customers;
    List<Item> items;

    Shop() {
        customers = new ArrayList<Customer>();
        items = new ArrayList<Item>();
    }

    public Customer addCustomer(String firstName, String lastName) {
        Customer customer;
        if ((customer = getCustomer(firstName, lastName)) == null) {
            customer = new Customer(firstName, lastName);
            customers.add(customer);
            System.out.println("Customer " + firstName +
                             " " + lastName + " added to customers.");
        }
        return customer;
    }

    public Customer getCustomer(String firstName, String lastName) {
        for (Customer currentCustomer : customers) {
            String currentFirstName = currentCustomer.getFirstName();
            String currentLastName = currentCustomer.getLastName();
            if (currentFirstName.equals(firstName)
                && currentLastName.equals(lastName)) {
                return currentCustomer;
            }
        }
        return null;
    }

    public Item addItem(String itemName, double itemPrice, int itemQuantity) {
        Item item;
        if ((item = getItem(itemName)) == null) {
            item = new Item(itemName, itemPrice, itemQuantity);
            items.add(item);

            System.out.println("Item " + itemName + " added " +
                                itemQuantity + " times to itemlist.");
        }

        return item;
    }

    public Item getItem(String name) {
        for (Item currentItem : items) {
            String currentItemName = currentItem.getName();
            if (currentItemName.equals(name)) {
                return currentItem;
            }
        }
        return null;
    }

    public boolean buyItem(Customer customer, Item item, int quantity) {
        String customerFirstName = customer.getFirstName();
        String customerLastName = customer.getLastName();
        Customer currentCustomer = getCustomer(customerFirstName, customerLastName);

        String itemName = item.getName();
        Item currentItem = getItem(itemName);

        if (currentCustomer != null && currentItem != null) {
            System.out
                    .println("Customer \"" + customerFirstName + " " +
                     customerLastName + "\" buys " + quantity +
                     " \"" + itemName + "\"");
            return true;
        } else {
            return false;
        }
    }

    public static void main(String[] args) {
        Shop shop = new Shop();

        shop.addCustomer("John", "Miller");
        shop.addCustomer("Franc", "Magnamera");
        shop.addCustomer("Doris", "Francini");
        shop.addCustomer("Sophie", "Whatson");
        shop.addCustomer("Michael", "Foster");

        shop.addItem("Samsung LE-40A656", 211.86, 3);
        shop.addItem("Sony KDL-40W4500", 1158.66, 34);
        shop.addItem("Apple iPhone 3G 8GB", 589, 13);
        shop.addItem("Nokia 5800 XpressMusic blue", 329, 2);
        shop.addItem("Nintendo Wii Fit inkl. Wii Balance Board", 79.37, 17);

        Customer customer = shop.getCustomer("Michael", "Foster");
        Item item1 = shop.getItem("Samsung LE-40A656");
        Item item2 = shop.getItem("Sony KDL-40W4500");
        shop.buyItem(customer, item1, 3);
        shop.buyItem(customer, item2, 30);
    }
}
```

8.6.3 [Task Analytic 3] Task Assess

Duration: max. 10 min.

Task Model (Description):
The goal of assessment is to characterize a case in terms of a decision class. The underlying knowledge typically consists of a set of norms or criteria that are used for the assessment.

Task Instance:
Tuition Fees: Suppose you are the student "Franz Maier" from Styria, Austria with the stated profile below. Assess if you have to pay tuition fees for the summer term 2009 or not. Save your decision and the evidence for your decision on the computer. Student profile: He is studying since 2004 summer term only Bacc. Software Development and Economics, he was working since 2 years as at a company as an IT administrator and at which he earns 300 euros per month, has done 25 (week) hours lectures in winter term 2008/2009, he has already finished his first part (in German "1. Abschnitt") of his Bacc. curriculum in the summer term of 2005. Useful links to start: `http://studienbeitrag.htu.tugraz.at`, `http://www.tugraz.at/studienbeitrag`.

8.6.4 [Task Analytic 4] Task Predict

Duration: max. 15 min.

Task Model (Description):
In prediction, one analyzes current system behavior to construct a description of the system state at some future point in time. A prediction task is often found in knowledge-intensive modules of teaching systems. The inverse of prediction also exists and is called retrodiction.

Task Instance:
Suppose you take the exam for the course "Economics (BWL)" next month. Based on historical exam questions available at `http://pbs.htu.tugraz.at/wiki/index.php/Betriebswirtschaftslehre` your task is to predict 3 questions which will probably be asked at the next exam. Save your results on the computer.

8.6.5 [Task Synthetic 1] Task Design

Duration: max. 12 min.

Task Model (Description):
Design is a synthetic task in which the system to be constructed is some physical artifact. Design

tasks in general can include creative design of components. In order for system construction to be feasible, we generally have to assume that all components of the artifact are predefined. This subtype of design is called configuration design. Building a boat from a set of *Lego* blocks is a well-known example of a configuration-design task. Another example is the configuration of a computer system.

Task Instance:
Create a simple conceptual design of the software for an elevator system. The software should be able to handle standard elevator control commands and movements. Do not implement the program, just design it. Save your results on the computer.

8.6.6 [Task Synthetic 2] Task Assign

Duration: max. 10 min.

Task Model (Description):
Assignment is a relatively simple synthetic task, in which we have two sets of objects between which we have to create a (partial) mapping. The assignment has to be consistent with constraints

Task Instance:
A professor has 8 tutors (study assistants) that support him/her to assists students in a programming course. 160 students are registered for this course. 6 tutors (T1,...,T6) can support at most 25 students each. Tutor T7 can only take care of 10 students because of time-constraints. Student S1 and S5 would like to join the group of tutor T1. Student S81 is not allowed to join the group of tutor T5 or T6. Save your results on the computer.

8.6.7 [Task Synthetic 3] Task Plan

Duration: max. 13 min.

Task Model (Description):
Planning shares many features with design, the main difference being the type of system being constructed. Whereas design is concerned with physical object construction, planning is concerned with activities and their time dependencies. Again, automation of planning tasks is usually only feasible if the basic plan elements are predefined. Because of their similarity, design models can sometimes be used for planning and vice versa.

Task Instance:
Plan a software project for the development of a document management system. Plan the activities you have to do and generate a sequence in which they have to be executed. Save your results on the computer.

8.6.8 [Task Synthetic 4] Task Schedule

Duration: max. 10 min.

Task Model (Description):
Scheduling often follows planning. Planning delivers a sequence of activities; in scheduling, such sequences of activities ("jobs") need to be allocated to resources during a certain time interval. The output is a mapping between activities and time slots, while obeying constraints ("A should be before B") and conforming as much as possible with the preferences ("lectures by C should preferably be on Friday"). Scheduling is therefore closely related to assignment, the major distinction being the time-oriented character of scheduling.

Task Instance:
Scheduling a software project: Suppose you are responsible for the design and development of an electronic library book lending system this term together with four student colleagues of yours. The software project starts on the 9th of March and must be finished at the 20th of June. Generate a schedule in which you state on which activities you and your colleagues will work on. You do not have to do the activities. You should just schedule them. Save your schedule on the computer.

Following deadlines have to be taken into account:

- Register the group for the course (Deadline 30th March)
- Hand in the design document (Deadline 30th May)
- Hand in the completed project (Deadline 20th June)

Following activities have to be scheduled in the given order:

1. Find group members
2. Read about known technologies and similar projects and choose the most appropriate
3. Define the requirements of the system
4. Design the system
5. Split the tasks and assign them to particular course members
6. Implement the system
7. System testing
8. Bug fixing