Master's Thesis

Supporting Personalization Systems with Collaborative Recommendations Exploiting External Heterogeneous Data

Andreas Felix Hütter, BSc

Institute for Information Systems and Computer Media (IICM),
Graz University of Technology

Supervisor: Univ. Doz. Dipl.-Ing. Dr.techn. Christian Gütl

External Supervisor: Dipl.-Ing. Dr.techn. Herwig Rollet

Graz, April, 2011

Masterarbeit

(Diese Arbeit ist in englischer Sprache verfasst)

Unterstützung von Personalisierungssystemen mittels Collaborative Filtering durch Erschließung Externer Heterogener Daten

Andreas Felix Hütter, BSc

Institut fuer Informationssysteme und Computer Medien (IICM),
Technische Universitaet Graz

Betreuer: Univ.-Doz. Dipl.-Ing. Dr.techn. Christian Gütl
Externer Betreuer: Dipl.-Ing. Dr.techn. Herwig Rollet

Graz, April, 2011

This page intentionally left blank

Abstract

The amount of data sources grows exponentially with every day due to the increasing number of mass media and global communication facilities. With that, the overload of information becomes more and more severe which challenges people to find relevant information of interest among the vast amount of alternatives. Moreover, many people find it difficult to articulate what they want, but can easy recognize it when they see it. Personalization and recommender systems have been developed to address these problems by suggesting people things they possibly might prefer or by helping them articulating their needs and demands. To fulfill their tasks, such systems need a solid knowledge base. As such knowledge bases often grow over time, for instance, through collecting information about customer behavior, there are situations when this knowledge base is not developed well enough. Many recommender systems have to face cold-start problems which arise when there is not enough information available about system users or items these systems aim to recommend. Due to this special data sparsity problem, many recommender systems have problems to generate recommendations that are based on collected user histories.

The goal of this work is to assist the recommendation process of an existing personalization system, especially in cold-start situations. Therefor a recommender system prototype was implemented that is able to provide supplementary collaborative recommendations based on an alternative knowledge source that contains the needed user histories. To obtain the alternative knowledge source valuable data and information about user preferences has been extracted from customer reviews. As customer reviews are usually written in natural language that is not understandable to computational tasks, adequate methods have been developed to automatically process the textual data.

The prototype was implemented based on insights gained from in depth research about established methods and current approaches in the field of natural language processing and recommender systems. To measure the optimization potential of the approach proposed in this work, the collaboration of the prototype and the existing personalization system was evaluated with adequate use-case simulations. The very promising results show a clear improvement of the recommendation process of the existing personalization system in cold-start situations.

Keywords

recommender systems, collaborative filtering, nlp, product feature extraction, cold-start problem, user-clustering

This page intentionally left blank

Kurzfassung

Die Menge an Datenquellen wächst mit jedem Tag exponentiell, nicht zuletzt aufgrund der ständig steigenden Anzahl an Massenmedien und globalen Komumikationsmöglichkeiten. Dadurch kommt es zu einer immer größer werdenden Informationsüberflutung, welche Menschen das Auffinden von relevanten Informationen, angesichts der riesigen Menge von Alternativen, erschwert. Darüber hinaus haben viele Menschen Probleme zu artikulieren was sie wollen, können jedoch das gesuchte leicht erkennen, wenn sie es sehen. Personalisierungs- und Empfehlungssystem wurden entwickelt, um diese Probleme zu addressieren, indem sie Menschen Dinge vorschlagen, welche sie möglicherweise perferieren oder indem sie Menschen helfen ihre Anforderungen und Wünsche zu artikulieren. Um ihre Aufgaben zu erfüllen, benötigen derartige Systeme eine solide Wissensbasis. Da derartige Wissensbasen oft erst mit der Zeit wachsen, zum Beispiel durch das Sammeln von Informationen über das Verhalten von Kunden, gibt es Situationen, in denen diese Wissensbasis noch nicht in benötigtem Umfang entwickelt ist. Viele Empfehlungssysteme werden mit sogenannten cold-start Problemen konfrontiert, welche auftreten, wenn nicht genügend Informationen über Systembenutzer oder Gegenstände, welche von derartigen Systemen vorgeschlagen werden sollen, vorhanden sind. Aufgrund dieses speziellen Problems von Datenspärlichkeit, haben viele Empfehlungssysteme Probleme Empfehlungen, die auf gesammelten Benutzerhistorien basieren, zu generieren.

Das Ziel dieser Arbeit ist es, dem Empfehlungsprozess eines existierenden Personalisierungssystemes zu assistieren, besonders wenn dieses System sich in sogenannten cold-start Situationen befindet. Um dies zu bewerkstelligen, wurde ein prototypisches Empfehlungssystem implementiert, welches die Möglichkeit hat supplementäre kollaborative Empfehlungen basierend auf einer alternativen Wissensbasis, welche die benötigten Benutzerhistorien enthält, zur Verfügung zu stellen. Um diese alternative Wissensbasis verfügbar zu machen wurden wertvolle Daten und Informationen über Benutzer-Präferenzen aus Kundenrezensionen extrahiert. Da Kundenrezensionen normalerweise in natürlicher Sprache verfasst sind, welche nicht von Computern verstanden werden kann, wurden adequate Methoden entwickelt, um diese textuellen Daten automatisiert zu verarbeiten.

Der Prototyp wurde auf Basis gewonnener Erkenntnissen aus eingehender Recherche über etablierte Methoden und aktuelle Ansätze aus Forschungsbereichen, welche sich mit Empfehlungssystemen und der Verarbeitung natürlicher Sprachen beschäftigen, implementiert. Um das Optimierungspotential des in dieser Arbeit angedachten Konzeptes zu messen, wurde die Zusammenarbeit zwischen Prototyp und dem existierenden Personalisierungssystem anhand entsprechender Anwendungsfall-Simulationen evaluiert. Die dabei erzielten Ergebnisse, welche sehr vielversprechend sind, zeigen in cold-start Situationen eine klare Verbesserung des Empfehlungsprozesses des existierenden Personalisierungssystems.

Schlüsselwörter
empfehlungssysteme, collaborative filtering, nlp, extraktion von produktmerkmalen, cold-start problem, user-clustering

This page intentionally left blank

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, April 11th, 2011

Andreas Felix Hütter

This page intentionally left blank

Acknowledgements

At first I want to thank in particular my advisor Christian Gütl for his great supervision of this thesis. He always showed much interest for this work and supported me not only with his technical knowledge, but also with valuable personal advice and a lot of patience.

Equally I want to thank Herwig Rollet for giving me the opportunity to realize this interesting work. Herwig always took time for discussions and supported me with very useful counsels and his comprehensive know-how.

My biggest thanks go to my parents who supported me during my entire study time by any means and in any form with so much patience and understanding. From the beginning to the end they were my strongest backing and I could always count on them.

With all my heart I want to thank my wonderful girlfriend Julia. Without her I would have not been able to manage this very arrogating and stressful final phase of my study. She was always there for me with all her comprehension and patience and helped me to regain my strength after sleepless nights of challenging work.

Also I want to thank my fellow students who were always helpful during the many ups and downs of my study time.

Andreas Felix Hütter

Graz, April, 2011

This page intentionally left blank

Table of Contents

1. Introduction

In our modern times of global communication facilities and mass-media which nowadays is ubiquitous, the amount of data sources grows exponentially. With that the overload of information is becoming more and more severe. This handicaps the user's aptitude to discriminate relevant from irrelevant information and it becomes increasingly difficult for people to find desired information. (Blanco-Fernandez, Pazos-arias, Gil-Solla, Ramos-Cabrer, & Lopez-Nores, 2008; Cosley, Lawrence, & Pennock, 2002) Beyond that, most people may have problems articulating what they want, but they find it easy to recognize it when they see it (Middleton, De Roure, & Shadbolt, 2001). In many cases people need to make choices without having enough personal experience and knowledge of the given alternatives (Resnick & Varian, 1997). As simple questions such as which movie to see, what book to read or what city to visit arise in every day life time, decisions have to be made consistently. There are too many choices and not enough time to explore all of them. This problem is even increased due to the exploding availability of information that is provided by the web. (Rashid et al., 2002)

Various approaches and systems have emerged to support people to find the information they need and things they prefer. Search engines, for instance, often use *Information Retrieval* techniques such as query expansion and query suggestion to facilitate the search tasks of users. Query expansion is used to extend the original user search query with new search terms in order to narrow the search scope. The goal of query suggestion is to recommend full queries made by other users. With that the coherence and integrity in the suggested queries can be preserved. (Gao et al., 2007) For instance, Google[1] provides a feature called *Google Suggest*[2] that offers

[1]http://www.google.com Google Search Engine, last access 03/2011

[2]http://www.google.com/support/websearch/bin/answer.py?answer=106230 Google Suggest, last access 03/2011

search queries based on search activities of other users.

Another kind of system that helps people to discover the most valuable and interesting information are *Recommender Systems*. In daily routine, people trust recommendations from other people they know by reports from news media, travel guides, spoken words, reference letters and so on. The basic idea of a recommender system is to assist and enhance this natural social process. (Su & Khoshgoftaar, 2009) Loh, Lorenzi, Saldana, and Licthnow (2004) define a recommender system as a *"software to aid in the social process of indicating or receiving indication about what options are better suited in a special case for a certain individual."* Ricci, Rokach, Shapira, and Kantor (2010) define recommender systems as *"software tools and techniques providing suggestions for items to be of use to a user"*. The first recommender systems used algorithms to take advantage of recommendations generated by a group or community of users to provide recommendations to the active user searching for suggestions, aiming at imitating this social behavior. This approach where items liked by users with similar tastes or preferences are recommended, is called *Collaborative Filtering*. The collaborative filtering approach is grounded on the basic principle that if the current user agreed with other users in the past and therefore has the same preferences, other recommendations originating from these users with similar taste could be interesting and relevant to the active user as well.

Recommender systems are used in various application domains such as Entertainment, Content, E-Commerce and Services. The domain of entertainment includes recommender systems such as movie or music recommender systems. The content domain covers, for instance, recommendation of websites or documents, applications for e-mail filtering, applications for e-learning and personalized newspapers. E-commerce recommender systems aim to suggest consumers what products, such as cameras, PCs, DVDs, books etc., they should buy. The application domain of services includes recommender systems suggesting experts for consultation, matchmaking services, recommender systems of travel services, or recommendation of houses to rent and similar. There are many popular recommender systems such as *MovieLens*[3], *WhatShouldIReadNext*[4] or the recommender system of *Amazon.com*[5].

[3]http://movielens.umn.edu/html/tour/index.html MovieLens Website, last access 03/2011
[4]http://whatshouldireadnext.com/faq.php WhatShouldIReadNext Website, last access 03/2011
[5]http://www.amazon.com/gp/help/customer/display.html?ie=UTF8&nodeId=13316081 Amazon.com Website, last access 03/2011

MovieLens uses collaborative filtering to generate movie recommendations. To generate personalized recommendations for a certain user the system uses ratings about movies made by like-minded users with similar opinions. WhatShouldIReadNext makes book recommendations based on collective taste of real readers by using favourites lists. Books in the same favourites list are associated with each other whereas the more frequently certain books appear on different favourites lists, the higher becomes the strength of that association. The recommender system of Amazon.com uses information about purchased and rated items to compare the activity of a customer to that of others. Based on that comparison, items that might be of interest to a user can be recommended.

All mentioned types of systems that aim to support people to detect items and information that are of interest have one thing in common: they need adequate knowledge sources to fulfill their tasks. Many recommender systems collect the required information over time. If the quality or quantity of the required knowledge source is not high enough these systems are not able to make useful recommendations or suggestions to users to find the required information or items they might be interested in.

1.1 Motivation

Recommender systems use various types of knowledge sources to fulfill their tasks (Felfernig & Burke, 2008). In general, most recommender systems try to estimate what services or products are the most suitable for a certain user based on collected user's constraints and preferences. These can either be explicitly expressed via item ratings or deduced by the interpretation of user actions and user behavior. (Ricci et al., 2010) Recommender systems gather information about user preferences in time and try to find things of similar interest automatically, whereby the user's effort to create explicit queries to articulate what he or she wants can be reduced. However, if a system has not enough or no inital information about new users or new items and not enough user ratings have been collected it is difficult or impossible to make useful recommendations. This problem is often referred to as *Cold-Start Problem*. Especially recommendation techniques such as collaborative filtering which generate recommendations based on finding users with similar behavior, suffer from cold-start situations. (Middleton, Alani, & Roure, 2002) The main objective of this work is to support the recommendation task of an existing personalization system, particularly when it is in a cold-start situation where no recommendations can be made based on recorded user histories. To achieve this objective three associated goals have to be fulfilled. The first goal is to extend the knowledge base of the existing system to provide additional recommendable items. The second goal is to develop appropriate strategies to provide additional personalized recommendations based on user histories as a supplement to those generated by the existing system. The third goal is to provide another solution to address cold-start problems.

To reach the first goal by extending the knowledge base of the existing system the problem of finding appropriate data sources has to be solved first. The Internet provides massive data sources and information pools originating from the numerous internet communities and website portals that provide users with easy possibilities to produce huge amounts of data, such as customer reviews about certain products and user comments in social networks. Especially product reviews have more or less a good relation to the topic of recommender systems. Recommender systems try to suggest people items they might prefer and product reviews contain the articulated opinions of customers about certain items and often outline their personal preferences. Exploiting the information about user preferences given by product reviews can help to supply recommender systems with an additional knowledge source. This would solve the problem of finding adequate data sources to achieve the first goal. But to make this data source usable to a recommender system, again another problem has to be solved. Product reviews are mostly existing as texts written in natural language. Although they contain much valuable information, the problem is that they do not provide a suitable structured format. In principle unstructured raw data, such as natural language texts, are not readable and understandable to computers (Moens, 2006). On the basis of natural language processing technologies this kind of unstructured data can be transformed into a format that is better readable and processable by computational tasks (McCallum, 2005). With that, information can be gathered in a structured way to make it available to different information systems, such as a recommender system. This can be done by extracting the product features that are mentioned in product reviews to use them as supplementary items to those already covered by the knowledge base of an existing system. With that, the first goal can be reached.

By achieving the first goal, a new knowledge base can be built from items that are extracted from the product reviews. To achieve the second goal, which is to provide additional personalized recommdations based on user histories, information about preferences of other users has to be available. To solve this problem, additional information is gathered from the product reviews. The customers that wrote the product reviews can serve as users of a recommender system whereby the features articulated in the product reviews are considered as preferred items of that customers. With that, already existing rating histories consisting of ratings about the items outlined in the product reviews can be obtained.

The third goal is achieved by successfully fulfilling the first and second goal. By extending the knowledge base of the target system with additional items and user histories obtained from the product reviews, supplementary recommendations can be generated based on other user's behavior, even if the target system is in a cold-start situation.

To implement the proposed solutions an appropriate concept will be constructed in this work. In order to evaluate the potential of the proposed approach, a recommender system prototype will be implemented to support the recommendation task of the existing personalization system *Xohana*[6]. Xohana is an innovative market place with the goal to help people to better articulate what they really want. In its current implementation, Xohana helps people to articulate and describe the kind of vacation - especially in the domain of health tourism - they desire. Customers can define several requirements to that desired vacation without limitation of expression. Each requirement consists of a criterion with a certain demand. For instance if the customer wants vegetarian food and facilities for disabled he can define requirements such as "*food* should be *vegetarian*" and "*hotel facilities* should be *facilities for disabled*". Xohana facilitates this process of requirement articulation by making intelligent suggestions of that criteria and demands on a much finer grained level instead of suggesting products. These suggestions are personalized and always fit to the current user inquiry which. That is succeded by using both behavior patterns of previous customers (self-learning system) and pre-populated data (for instance sector knowledge modeled in the system and geographical relations). (Rollett, 2008;

[6]http://www.xohana.com Xohana e.U. Website, last access 03/2011

Semantic Web Company, 2009)

To support the suggestion process of Xohana the recommender system prototype provides supplementary recommendations of terms. The data that is used to perform the tasks of the prototypical implementation is at first extracted from a small but for this work sufficient amount of positively rated customer reviews about review objects, such as hotels, appartments, clubs or similar, from the online platform Tripadvisor.com[7]. By means of natural language processing technologies the relevant product features describing the review object and with that the preferences of the customers that composed the reviews are extracted. For the recommender system prototype these extracted features can be considered as items. To enable a collaboration between the prototype and Xohana these items are regarded as the terms Xohana suggests and the customer uses to describe his disired vaction. The reviewer is regarded as user that already provided ratings about the items (or terms) that were extracted from his review. Using the extracted items (or terms) the recommender system prototype can produce personalized recommendations (in addition to that produced by Xohana) of terms and provide additional terms to that already covered by the knowledge base of Xohana. Furthermore the prototype applies a query suggestion method that helps to predict requirement formulations. During the evaluation of the prototype, Xohana does not use any collected user history and is therefore in a cold-start situation.

1.2 Structure of the Work

The remainder of the thesis is structured as follows: Chapter 2 introduces natural language processing technologies that are relevant to this work with a special view on information extraction from unstructured natural language texts. Also this Chapter presents some state-of-the-art approaches for product feature extraction from customer reviews.

Chapter 3 gives an overview of the most popular types of recommender systems and the related sub-topics that are relevant to this work with particular consideration of collaborative filtering techniques and cold-start problems. Furthermore some state-of-the-art approaches that aim to overcome cold-start problems are introduced.

[7]http://www.tripadvisor.com Tripadvisor.com homepage, last access 03/2011

In Chapter 4 the constructed conceptual design that covers the proposed ideas and describes how the goals of the proposed apporach can be achieved. This Chapter introduces the basic ideas about how the product features can be extracted from product reviews and shows how the obtained data sources can be used to deliver the recommender system prototype.

The subsequent Chapter 5 describes implementation issues and the technologies and tools that are used to develop the recommender system prototype. The product feature extraction process and recommendation methods are explained in more detail.

In Chapter 6 several simulated tests of the recommendation process of Xohana in collaboration with the prototype are performed to measure the optimization potential of the recommender system prototype.

The results obtained from Chapter 6 are discussed and interpreted in Chapter 7. Additionally the lessons learned during research and the implementation of the prototype are emphasized.

Finally Chapter 8 makes conclusions about the entire work and looks into to possible future work.

2. Information Extraction from Natural Language Text

One major goal of the approach proposed in thesis is to extract product features and user preferences from customer reviews. These extracted data and information are supposed to build a valuable knowledge base that can be used to supply the supportive recommender system prototype. As the used customer reviews are written in natural language and are therefor only available in an unstructured format which is not understandable to computational tasks. To obtain the needed information from the customer reviews the unstructured texts have to be transformed in a more structured format. *Information Extraction* technologies provide adequate functionality to fulfill this task. Information extraction belongs to the broad field of *Natural Language Processing* which will be shortly briefed in this Chapter. Equally the basic principles and major tasks of information extraction will be introduced. In a next step an overview of technologies that can be used as subtasks to information extraction from natural language texts will be provided. Additionally some state-of-the-art approaches addressing information extraction from unstructured data - in that case product feature extraction or identification from consumer reviews - are introduced. Some of them involve opinion mining, also known as sentimental analysis.

2.1 Natural Language Processing

Existing for more than fifty years *Natural Language Processing* (NLP) encompasses the field of computational linguistics and is considered to be a subfield of artificial intelligence (AI) (Jia-li & Ping-fang, 2010). Jackson and Moulinier (2002) define the term *"Natural Language Processing"* (NLP) as *"the function of software and hardware components in a computer system which analyze or synthesize spoken or written language"*. Human speech and writing have to be distinguished from more formal languages, such as computer languages like C++ and Java and logical or mathematical notations. As a subtopic of NLP *Natural Language Understanding* (NLU) deals with the goal of enabling computer systems to understand natural language the way humans do. (Jackson & Moulinier, 2002)

The ability to automatically decode natural language is becoming more and more important. In addition to focusing on the interactions between natural languages and computer systems, natural language processing concentrates on information sharing, thus nowadays the exchange of information is a very crucial task. As a result many applications and activities are covered by the field of natural language processing, such as information retrieval, foreign language reading support, natural language understanding, data mining, automatic summarization, data integration, optical character integration, electronic dictionary and so forth. (Jia-li & Ping-fang, 2010)

2.1.1 Tasks of Natural Language Processing

Jackson and Moulinier (2002) list different tasks of natural language processing:

- Document retrieval: On the web, document retrieval is considered to be primary task of language processing. The goal of document retrieval is to locate documents that are relevant to a user based on the performed search query. This task can indeed be done without using natural language processing, but to increase sophistication in the tasks of indexing, identifying and presenting documents that are relevant to the user, natural language processing became more and more significant since the 1990s.
- Document routing: As a task that is related to document retrieval and as document classification, document routing aims to classify documents, normally based upon the content.
- Information extraction: In comparison to document retrieval the goal of information extraction is not to find the most relevant documents, but to extract information of interest from a certain document or a set of documents.

- Document summarization: As a subtask of information extraction, document summarization is used to extract the most noticeable information from a document to represent the original document by a surrogate document containing the summarized information.

As a major part of this work focuses on information extraction from unstructured texts, the next section provides some more detailed information about this subfield of natural language processing.

2.1.2 Information Extraction

There are many situations in business tasks, research, studying and other situations in daily life time that all have the request for information in common. Usually the potential answers to this request resides in data sources that are unstructured, such as images and texts. On the one hand humans are not able to process all existing data because of the overload of data sources and information, despite of being aware to understand this kind of data. computers are able to handle a much higher amount of data, but to directly query for the required information data has to be available in a structured format, such as a database. *Information Extraction* (IE), as a subfield of *Artificial Intelligence* (AI), aims to provide solutions for such problems. (Moens, 2006)

Moens (2006) defines information extraction as the *"identification, and consequent or concurrent classification and structuring into semantic classes, of specific information found in unstructured data sources, such as natural language text, making the information more suitable for information processing tasks."* That means, the main goal of information extraction tasks is to identify information of interest within unstructured data, such as spoken text and written natural language text, audio and video, and to represent the extracted information in a format that is more suitable for computers. To make it easier for computers to process the data, the information extraction process adds meaning to raw, unstructured data, thus to provide semi-structured or structured data that is *"computationally transparent"*. (Moens, 2006)

Information extraction covers different subfields such as pattern matching, string matching and part-of-speech tagging. As these information extraction technologies, amongst others, have been used to realize the concept of the implemented prototype they will be introduced in the following sections. Additionally some state-of-the-art approaches dealing with product feature extraction from customer reviews, written in natural language, will be discussed in Chapter 2.

(McCallum, 2005) remarks that information that is locked in natural language has to be converted into a structured and normalized database form. This can be done by information extraction, which can also be regarded as the process of filling the records and fields of a database from text that is unstructured or not strictly formatted. A database that was populated from loosely formatted or unstructured text by the means of information extraction, can be further processed by data mining to discover patterns within that database. From this point of view information extraction can be considered as a prestage of data mining. (McCallum, 2005) lists five major subtasks involved by information extraction:

1. Segmentation: Detecting the starting and ending boundaries of certain textual segments that are of interest to be inserted into a database field, for instance to extract the course title occurring in educational texts.
2. Classification: As there might be different types of information that have to be extracted, this subtask aims to assign the correct database field the extracted text segments. For instance, an educational text could contain information about the course title, the course instructor, the course schedule or similar.
3. Association: This subtask, also referred to as relation extraction, has the goal to determine which entities are associated to each other. For instance, to automatically find out which politicians of which countries had a meeting by extraction the required information from news articles. Usually commercial applications that provide relation extraction are quite rare in comparison to those that only use segmentation and classification for the information extraction task.
4. Normalization: To make information reliable comparable, it has to be put into a standard form. Normalization is important for information including numeric values, such as time formats, and as well for string values, such as full names of persons, where first and last name always should be in the same order.
5. Deduplication: As some identical information might be extracted several times

originating from different sources, this subtask has the goal to remove duplicate database records. For example, in news articles famous politicians can be named differently, although they refer to the same person.

2.2 Information Extraction Technologies

As already mentioned in Section 2.1.2, information extraction aims to identify specific information of interest within unstructured data, such as natural language texts, to represent the gained information in a more structured format that makes it more suitable for computers and better reusable for further processing. There are several technologies within the very broad field of natural language processing, that provide different approaches to process unstructured texts. Technologies, such as *String Pattern Matching* and *Part-of-speech Tagging*, can be utilized as handy tools to extract specific information of interest from natural language texts. The next sections introduce some technologies that can be used as subtasks to information extraction from natural language texts that are related to this work.

2.2.1 String Matching

A string is a sequence of symbols over a finite set or alphabet. *String Matching* can be generalized as the problem of detecting all occurrences of a certain string, called pattern, with certain properties within a given sequence of symbols, called text. The string pattern and the text consist of characters from the same alphabet. As one of the most predominant and oldest problems in computer science, there are plenty of applications that require some kind of string matching. Recently, the interest in string matching problems increased, particularly because of computational biology and information retrieval communities that are growing very fast. The text sizes that have to be managed become larger and search tasks become more and more challenging. In addition to simple strings, search patterns may include regular expressions, wildcards or gaps. In cases where the match of a given search string does not have to be exact, certain differences between the search pattern and its occurrence within the text may be permitted. This type of string matching is called *Approximate Matching.* (Navarro & Raffinot, 2002, p. 1–3)

Generally speaking approximate matching or approximate string matching is the problem of string matching allowing errors. That is to find a text containing a given pattern allowing a certain discrepancy - or in other words a limited number of "errors" - in the matches. Depending on the error model applied by an application, strings are considered to be more or less different. The *Levenshtein distance*, which is also called *edit-distance*, is a pervasive error model, which indicates how many operations have to be done to make both strings equal. (Navarro, 2001)

2.2.2 Pattern Matching with Wildcards

Wild Card characters, also called *"don't cares"*, can be applied for many real-problems that involve pattern matching (or text matching or string matching), such as text indexing, time series data mining, stream data mining, biological sequence analysis, and so forth (Wu, Wu, Min, & Li, 2010). Additionally the usage of wildcards is very common in many fields of computer science, including examples found in operating system shells, SQL and scripting languages such as Python, Awk and Perl. Wild card symbols are often represented as "$*$", "$\#$", "?", or "ϕ" and can be used to match any character of an existing set of symbols. (Rafiei & Li, 2009)

Some approaches for pattern matching witch wildcards enable the usage of wildcards with a constant length, which involves a disadvantage, because in most cases the length of wildcards defined between every two successively characters in a pattern can not be known beforehand. To overcome this limitation other approaches, which gained abundant attention, allow flexible gap constraints so that the length of wildcards is a range instead of a constant. (Wu et al., 2010)

2.2.3 Part-of-speech Tagging

The process of part-of-speech tagging, for short just called *Tagging*, encompasses the tasks of assigning each word in a text corpus the belonging part of speech, also known as word classes, lexical tags, morphological classes or POS, or another syntactical class marker (Jurafsky & Martin, 2009). Part-of-speech is a very crucial element in almost every human language. Already about 100 B.C. a grammatical sketch of Greek that outlined the linguistic knowledge at that time, included a description of eight parts-of-speech (Jurafsky & Martin, 2009):

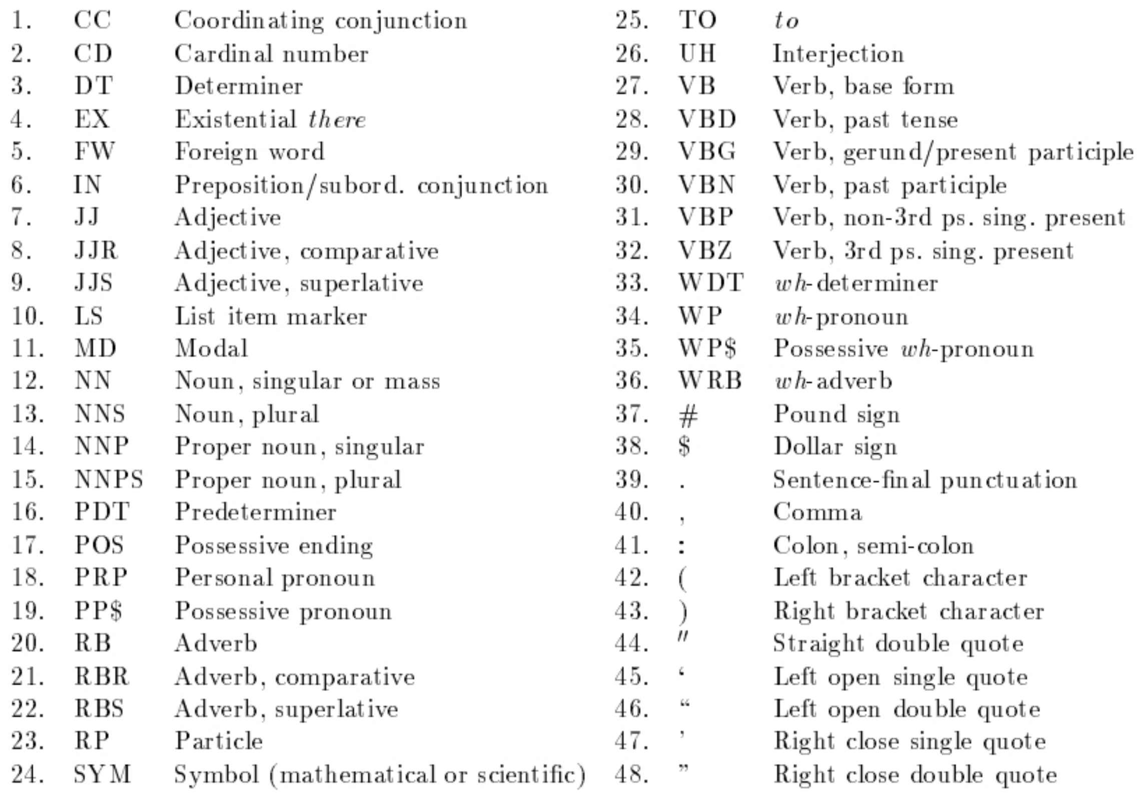
- noun
- verb
- pronoun
- preposition
- adverb
- conjunction
- participle
- article

For the subsequent 2000 years this recorded set served as basis for almost all part-of-speech descriptions of Latin, Greek and most European languages that emerged hereafter. Current lists of parts-of-speech, also known as tagsets, consist of much more word classes. For instance, the *Brown Corpus* contains 87 word classes. (Jurafsky & Martin, 2009) Another well-known corpus is the *Penn Treebank*, a tagset that consists of 48 word classes, where 36 are POS tags and 12 are other tags used for currency symbols and punctuation (see Figure 2.1). The Penn Treebank is based on a modification of the Brown corpus and contains over 4.5 million words of American English. To reduce the lexical and syntactical redundancy the tagset of the Brown Corpus was pared down considerably. (Marcus, Marcinkiewicz, & Santorini, 1993)

According to Jurafsky and Martin (2009) parts-of-speech can be divided into two major subcategories:

- Open class types: There are four main open classes that occur in the existing human languages - nouns, verbs, adjectives and adverbs. Whereas not each language includes all of these four classes, the English language does. The membership of open class words is constantly extended as words of this type are being borrowed or adopted from other languages.
- Closed class types: The membership of closed class words is rather fixed. For instance, there is a fixed set of prepositions in English language.

In language processing, parts-of-speech provide meaningful information about words and their related neighbors. Knowing the part-of-speech of a word can be useful in speech recognition, where it is very important to know how a word has to be pronounced. For instance, the word content is pronounced differently depending on whether it represents the noun or the adjective. Automatic assignment of a



1.	CC	Coordinating conjunction	25.	TO	*to*
2.	CD	Cardinal number	26.	UH	Interjection
3.	DT	Determiner	27.	VB	Verb, base form
4.	EX	Existential *there*	28.	VBD	Verb, past tense
5.	FW	Foreign word	29.	VBG	Verb, gerund/present participle
6.	IN	Preposition/subord. conjunction	30.	VBN	Verb, past participle
7.	JJ	Adjective	31.	VBP	Verb, non-3rd ps. sing. present
8.	JJR	Adjective, comparative	32.	VBZ	Verb, 3rd ps. sing. present
9.	JJS	Adjective, superlative	33.	WDT	*wh*-determiner
10.	LS	List item marker	34.	WP	*wh*-pronoun
11.	MD	Modal	35.	WP$	Possessive *wh*-pronoun
12.	NN	Noun, singular or mass	36.	WRB	*wh*-adverb
13.	NNS	Noun, plural	37.	#	Pound sign
14.	NNP	Proper noun, singular	38.	$	Dollar sign
15.	NNPS	Proper noun, plural	39.	.	Sentence-final punctuation
16.	PDT	Predeterminer	40.	,	Comma
17.	POS	Possessive ending	41.	:	Colon, semi-colon
18.	PRP	Personal pronoun	42.	(	Left bracket character
19.	PP$	Possessive pronoun	43.	)	Right bracket character
20.	RB	Adverb	44.	"	Straight double quote
21.	RBR	Adverb, comparative	45.	‘	Left open single quote
22.	RBS	Adverb, superlative	46.	“	Left open double quote
23.	RP	Particle	47.	’	Right close single quote
24.	SYM	Symbol (mathematical or scientific)	48.	”	Right close double quote

Figure 2.1: The Penn Treebank tagset (Marcus et al., 1993)

word's part-of-speech is also useful for stemming for information retrieval, improving applications of information retrieval that aim to select important words of a certain type (for example nouns) from a document, word sense disambiguation, applications of information extraction, parsing and many other tasks. (Jurafsky & Martin, 2009)

The Part-of-speech Tagging Process

As mentioned in the previous section, part-of-speech tagging aims to automatically identify the parts-of-speech to the words (or tokens) within an input text. Mitkov (2003, p. 221–222) describes the general parts of the architecture behind this task that many taggers have more or less in common:

1. Tokenization: Before the input text that is passed to the tagging algorithm can be annotated with the proper parts-of-speech, a task called tokenization has to be done. Tokenization breaks down the input text into meaningful elements, called tokens. These tokens are more suitable for further analysis to identify utterance boundaries, word-like units and punctuation marks.

2. Ambiguity look-up: In this phase a lexicon and a guesser to associate the given tokens to their part-of-speech are used. In its simplest form, such a lexicon contains a list of word forms with their possible parts-of-speech. The guesser is used to analyze tokens that are not represented in the lexicon.
3. Ambiguity resolution or disambiguation: In this phase the tagger tries to resolve ambiguous meanings of the tagged tokens. To enable this, two different information sources are used. One information source contains information about the respective word itself. For instance, the information that a certain word occurs more frequently as a verb than as a noun. The information source contains information about word/tag sequences. For instance, if the predecessor of a word is an article or a preposition, noun analysis could be preferred over verb analysis. Resolving word ambiguities is still one of the most challenging tasks in part-of-speech tagging.

The following examples show how the output of a tagged input text returned by a part-of-speech tagging tool might look like. The text was tagged with the English part-of-speech tagger library *EngTagger*[1], which is a Ruby port of the *Lingua::EN::Tagger*[2]. The *Lingua::EN::Tagger* is a probability based and corpus-trained tagger that uses a set of probability values and a lookup dictionary. It uses statistical information about parts-of-speech from the Penn Treebank and a bigram (two-word) Hidden Markov Model for guessing the proper part-of-speech. The example output produced by the EngTagger library is returned in XML-format. The XML-tags represent the parts-of-speech and the value of the XML-tags represent the tagged tokens:

- Given the example input text *"The birds fly to the south in winter."* produces the following output: *"<det>The</det> <nns>birds</nns> <vbp>fly</vbp> <to>to</to> <det>the</det> <nn>south</nn> <in>in</in> <nn>winter</nn> <pp>.</pp>"*
- Given the example input text *"There is a fly in my soup."* produces the following output: *"<ex>There</ex> <vbz>is</vbz> <det>a</det> <nn>fly</nn> <in>in</in> <prps>my</prps> <nn>soup</nn> <pp>.</pp>"*

[1]http://engtagger.rubyforge.org EngTagger Library project homepage, last access 03/2011

[2]http://search.cpan.org/ acoburn/Lingua-EN-Tagger/Tagger.pm Lingua::EN::Tagger Library project homepage, last access 03/2011

One can see, that the word *"fly"* was tagged as a verb in the first sentence and as a noun in the second sentence. The part-of-speech tagset used by the EngTagger library is a modified version of the Penn Treebank tagset. One can compare the tag names in the examples shown above with the tag names of the Penn Treebank tagset from Figure 2.1 where the part-of-speech tag *"det"* returned from the EngTagger library corresponds to the part-of-speech tag *"DT"* of the Penn Treebank tagset.

Part-of-speech Tagging Algorithms

Various part-of-speech tagging systems have emerged so far, applying stochastic models, linguistic rules or a combination of both. Part-of-speech tagging methods can either be supervised or unsupervised. Based on pre-tagged corpora supervised tagging aims to learn tagging rules or to unburden the disambiguation process. Unsupervised methods do not need a pre-tagged corpus, but apply advanced computational techniques (such as the *Baum-Welch* algorithm), automatically generate tagsets, transformation rules and so forth. Unsupervised tagging methods use the gained information for the generation of contextual rules that are required by rule-based or transformation-based systems or the calculation of the probabilistic information that is required by stochastic methods. Part-of-speech tagging algorithms can further be divided into two main classes, that most approaches fall into: rule-based and stochastic algorithms. (Kumar & Josan, 2010)

To fulfill their task, rule-based systems usually use a large set of manually-constructed rules to resolve word ambiguities. Such a rule could, for instance, dictate that an ambiguous word should be tagged as a noun rather than as a verb if the subsequent word is a determiner. (Jurafsky & Martin, 2009) The hand-written disambiguation rules also consider contextual information and the morpheme ordering (Kumar & Josan, 2010).

Stochastic approaches employ a training corpus that enables the computation of the probability of a certain word that has a given tag in a given context. With that tagging ambiguities can be resolved. (Jurafsky & Martin, 2009) In other words, an unambiguously tagged text is used for estimating the likelihoods to choose the most probably sequence. The n-gram probability and the lexical generation probability is regarded to select the maximum likelihood probability. The *Viterbi-Algorithm*, that follows a Hidden Markov Model, is one of the most common algorithms applied for n-gram approaches. (Kumar & Josan, 2010)

Another noteworthy and one of the most frequently applied part-of-speech taggers is the so-called *Brill Tagger*, that combines statistical methods and machine learning on the basis of transformation based learning (Mohammad & Pedersen, 2003). The Brill Tagger is a trainable rule-based tagger where the training of the data is completely automated. In comparison to trainable stochastic taggers, relevant linguistic information is provided by using a small amount of non-stochastic rules. (Brill, 1994)

2.3 Product Feature Extraction from Customer Reviews

A main part of this work deals with possibilities to reuse information residing in existing customer reviews and catalog descriptions of tourism objects, such as hotels, holiday clubs, apartments, etc. In this context the goal was to extract data that describes features or attributes of a certain review object. Especially product reviews provide additional information about features of the rated object that are unknown or have not been explicitly expressed in a way that is easily accessible and locatable for humans or computers. Additionally, customer reviews do not only give some indication about how satisfied they were with the entire product, which mostly is obvious from an overall rating, but they also express what things they liked in particular. In an adequate context, customer reviews can reveal the customers preferences not only about a certain product, but also about his or her taste within the given domain. For instance, by processing travel and tourism customer reviews, useful information about what kind of holiday trip the customer prefers can be uncovered.

As computers are not aware to directly apply queries on unstructured data, these potential and useful information rests unsuitable for computational tasks. Thus, adequate methods, being aware of converting unstructured data into a structured format to use the desired information for fulfilling the required tasks, are needed. Various approaches concentrate on the mining and summarization of customer reviews, which, according to Somprasertsri and Lalitrojwong (2008), include 3 main tasks (see Figure 2.2):

1. Feature extraction: In a first step, object features that occur in each review are identified and extracted.
2. Polarity determination: Sentiment classification is performed to determine whether the extracted features have a positive or negative polarity.
3. Result summarization: To represent the results in a more effective way and to capture the opinion of customers, the results are summarized and visualized.

A major part of this work concentrates on product feature extraction from free format reviews (corresponding to the first task of the review mining and summarization process mentioned above) for further analysis and utilization for user preference determination. The next sections outline some state-of-the-art approaches addressing product feature extraction from customer reviews, where some of them also involve opinion mining.

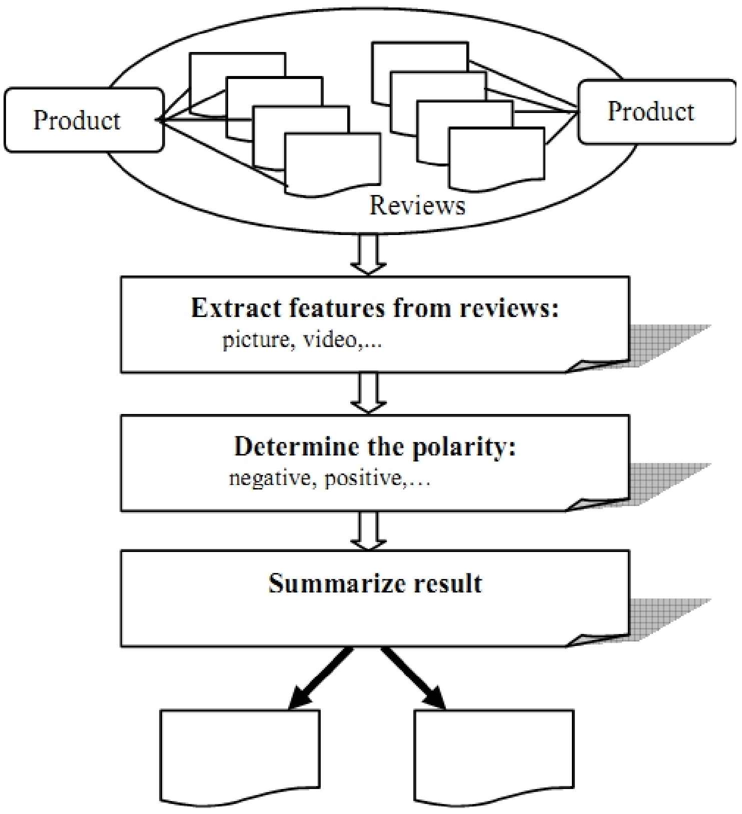


Figure 2.2: Process of review mining and summarization(Somprasertsri & Lalitrojwong, 2008)

2.3.1 Product Feature Mining with Nominal Semantic Structure

Zhan and Li (2010) describe an approach that aims to enable finer-grained extraction of product features from customer reviews on sentence level. By using a dependency tree an intrinsic structure with regard to the nominal semantic neighborhood is defined. With that the semantic dependency relations that exist between nominal and non-nominal terms, such as adjectives and nouns, can be taken in advance. The goal of this approach is to represent product features that are finer-grained. This features result from pairs of clusters, which can be clusters of nouns or their semantic neighbors. The methods used in this approach are applied on a data set of 710 digital camera reviews originating from "dpreview.com".

As mentioned by Zhan and Li (2010), in the field of opinion mining, features are referred to as all attributes and components characterizing products in a certain product domain. This part of the approach focuses on the problem of the extraction of specific product features via group of indicating the lexicons for each feature. Given the following example sentence *"Good camera produces favorite image quality"* it is obvious that unigram nouns are not sufficient to detect features, consisting of two or more nouns (in that case the feature *"image quality"*) , in lexicalized product feature representation. Likewise, given the example sentence *"Construction of the camera is very solid!"*, Zhan and Li (2010) claim that 3-grams or 4-grams are not adequate to properly represent the product feature *"construction of camera"* because the complete phrase contains the token "the". To overcome this Zhan and Li (2010) introduce noun fragments, which they define as *"largest subtree with nodes tagged as NN (noun) according to parts-of-speech (POS) or lexicalize as "of"."*

To perform opinion mining and to create opinionated pairs the most typical and important semantic neighbors, which include verb-predicate types and adjectives, for the distinction between noun fragments are selected. Adjectives, such as *good* and *excellent* can be used for opinion classification. Similarly verb predicates can be effectively used to classify noun phrases. Additionally certain verbs, such as *like* (or the negated forms, for instance *do not like*) can be utilized to build opinionated pairs. The authors refer this part of the presented approach as nominal semantic structure parsing. The upper part of Figure 2.3 shows how the semantic neighborhood with the sentential dependency tree is generated. The lower part of Figure 2.3 illustrates the corresponding structure of the semantic neighborhood. Noun fragments are represented by square boxes, where elliptical circles represent the semantic neighbors. The arrows show the dependency between the entities.

Selecting the most important semantic neighbors for the differentiation between noun fragments allows further co-clustering analysis. Given the opinionated pairs, clusters, that represent fine-grained product features, can be extracted by using a matrix factorization method. The authors state, based on their results, that their model is superior to baseline models with respect to defined nominal terms and bag-of-word unigram. (Zhan & Li, 2010)

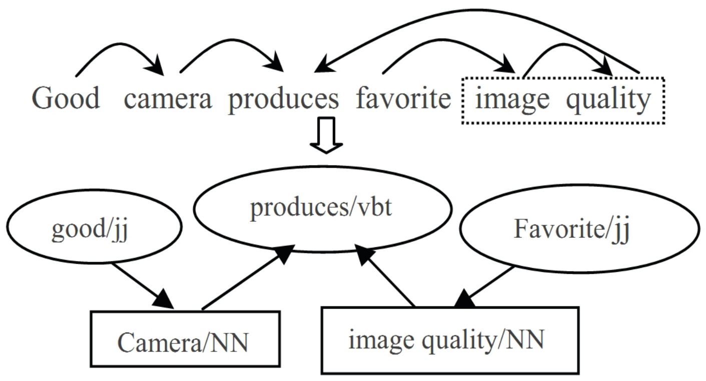


Figure 2.3: Semantic neighborhood structure (Zhan & Li, 2010)

2.3.2 Product Feature Extraction with a Combined Approach

To enable product feature extraction, which can be seen as the first phase of product review mining, Li (2010) introduces a combined approach that is based on bootstrapping and ID3, an algorithm used for feature selection in the iteration of bootstrapping. Bootstrapping is a method of semi-supervised learning and is quite popular in in the field of information extraction. It is an iterative process beginning with a manually composed seed set, consisting of positive samples, that is used to detect textual patterns, which again are used to extract new seeds. Li (2010) mentions that the experimental results, that are based on a review corpus of 1000 reviews of digital cameras from www.Amazon.com, show an affective performance of the introduced approach.

In this approach the design of the textual pattern structure is seen as the process of selecting features. Thus ID3 performs as an algorithm for feature selection that is used to build textual patterns that are differently structured. The ID3 algorithm is combined with bootstrapping, that allows to control the iterative process for the extraction of textual patterns. Li (2010) states that with that the design of similarity methods between textual patterns and the design of textual pattern structures are no longer necessary. Li (2010) divides the product feature extraction system into two stages: Data preprocessing and the automatic extraction of product features. Figure 2.4 provides a graphical representation of the entire product feature extraction system that is explained in the following sections.

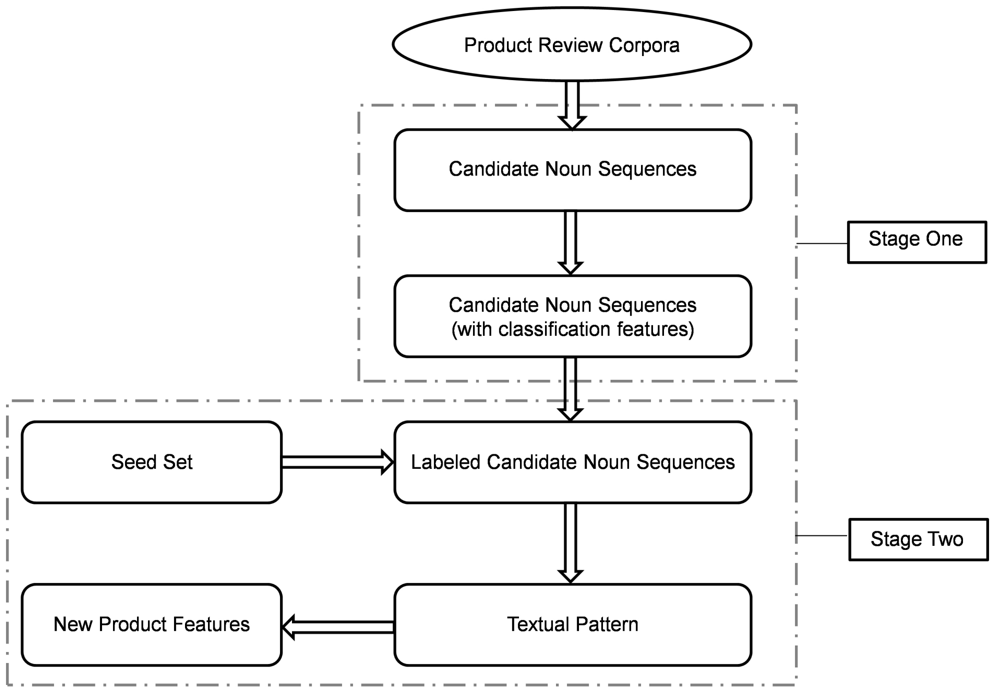


Figure 2.4: Graphical representation of the product feature extraction system, adapted from (Li, 2010)

Data Preprocessing

In this stage the approach proposed by Li (2010) extracts consecutive noun sequences from product reviews as candidate noun sequences, which are expressed via classification features. The sentences of the reviews are analyzed using lexical tools. While consecutive nouns sequences consisting of more than one word are further analyzed, thus a part of it and the whole consecutive noun sequence can be a candidate, single word consecutive noun sequences are immediately regarded as candidates. To determine if a certain subsequence or the whole sequence of the consecutive noun sequence is better suited to be used as a candidate noun sequence, the strength of the conjunction of the different subsequences performs as the decision criterion. To enable this, the mutual information among two subsequences is computed, whereas the conjunction of the two subsequences is regarded as a candidate noun sequence if the computated mutual information is bigger than a certain threshold. If it is smaller the subsequence having the bigger possibility is used as candidate noun sequence. (Li, 2010) Deciding if a candidate noun sequence is product feature or not is a classification task. To construct the contexts that are applied as classification features Li (2010) uses the syntactic tree containing the hierarchical view of the sentences and the dependency relation representing the relation between the words in a sentence. After the construction of classification features for the extraction of product features the candidate noun sequences are expressed using the classification features.

Automatic Extraction of Product Features

Li (2010) uses the seed set of product features, that was initially composed by hand, to tag all candidate noun sequences with positive or negative labels. If a candidate occurs in the seed set is tagged with a positive, otherwise with a negative label. The ID3 algorithm, a supervised learning algorithm, implements building a decision tree and creating a decision rule. In this work the ID3 algorithm is applied as a method for feature selection that enables the generation of textual patterns of different structures. The decision tree is built from candidate noun sequences including classification feature and labels. Each leaf in the constructed decision tree can have three different situations (Li, 2010):

1. All candidate noun sequences have positive labels - No new product features can be found, as the product features detected by this decision rule already exist in the seed set.
2. All candidate noun sequences have negative labels - This decision rule generated from leaves containing only negative labels can not decide if the candidate is a product feature or not. As the seed set may differ in another iteration (because the seed set may receive more and more features), the candidate noun sequence might be labeled to positive in one of the next iterations.

3. The leaves have positive and negative labels - Decision rules are generated from this situation and are viewed as textual patterns, that are added to the set of the candidate textual patterns.

Figure 2.5 shows an example of the decision tree which is built of twenty samples with positive and negative labels, where all samples have the three features f1, f2, and f3, that can either have the value 1 or -1. As only leaf 2 and 4 have positive and negative labels decision rules are only constructed from these leaves and added to the candidate textual patterns set. (Li, 2010) To provide a good performance, in each iteration of bootstrapping the approach of Li (2010) selects only the best candidate textual patterns by estimating their confidence. This confidence estimation is computed under the hypothesis that the more new product features that occur in the seed set are extracted by a certain candidate textual pattern, the higher is the confidence of that pattern. Using the textual patterns in the textual pattern set, candidate noun sequences are extracted and added to the candidate product feature set as a product feature candidate. Each textual pattern that was used to extract a candidate noun sequence is recorded.

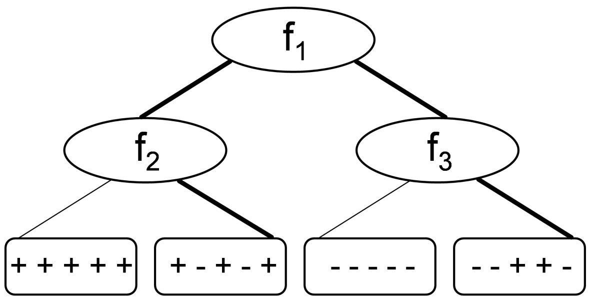


Figure 2.5: Example of the decision tree, adapted from (Li, 2010)

2.3.3 Automatic Product Feature Extraction from Online Product Reviews using Maximum Entropy with Lexical and Syntactic Features

Somprasertsri and Lalitrojwong (2008) propose an approach for extracting product features from online product reviews that combines syntactical and lexical features with a maximum entropy model, a framework that is used for classification by using integrated information that origins from several heterogeneous sources. The introduced approach concentrates on reviews that are written in free form. The maximum entropy model is used to fulfill the classification task, that aims to indicate whether a word in a sentence is a product feature or not. To constrain the maximum entropy model important information and features that function as learning features and that have to be distinguished from product features, are defined. Several features are computed for each word from the training data automatically (Somprasertsri & Lalitrojwong, 2008):

- Word: Defines the target words and the belonging parts-of-speech.
- Rare: Frequent noun/noun phrases are more likely to be product features than rare words (infrequent noun/noun phrases).
- Alphanumeric: The information whether a word contains letters and numerals can be useful for discriminating product features from non-product features.
- Dependency: Using a dependency tree that is derived from the syntactic parse tree provides valuable information about what other words are dependent on a certain word.

Somprasertsri and Lalitrojwong (2008) divide the introduced system into training module and product feature extraction module. At first the training module performs parsing tasks and manually annotation of the product features to prepare the training data. Then learning features of all words comprised in the training data set are extracted and finally, the model is trained by means of the maximum entropy model and provides as result the weights of all feature functions. Somprasertsri and Lalitrojwong (2008) name three tasks that are conducted by the product feature extraction module:

1. Product feature candidate selection: After the parsing of a sentence is completed, words that are likely to be product features, such as nouns and adjectives, are selected from a tagged sentence.
2. Product feature extraction using the maximum entropy model: By using the previously trained model to predict the candidates for product features from the unlabeled reviews, the class with the highest conditional probability is chosen.
3. Postprocessing: To find the remaining product features each word in the product reviews is matched against the list of product features that have been extracted.

Figure 2.6 shows a graphical representation of the introduced system. (Somprasertsri & Lalitrojwong, 2008) Somprasertsri and Lalitrojwong (2008) used product reviews about one MP3 player and one digital camera from the Amazon Website. 1,500 sentences were used for the product feature extraction experiments, where the data was split into 80% training set and 20% testing set. Somprasertsri and Lalitrojwong (2008) mention that precision and recall of their approach are higher than those obtained by the approach of Hu and Liu.

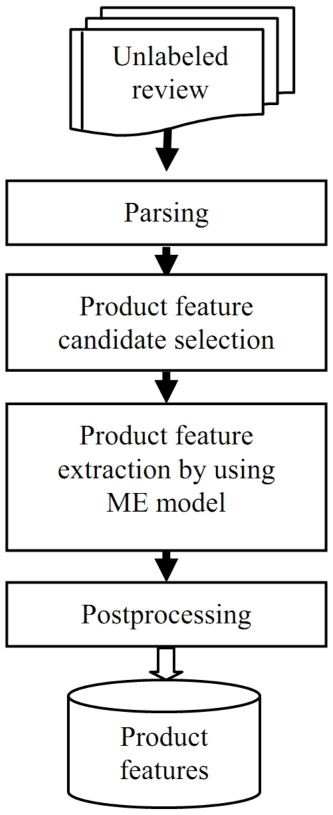

Unlabeled review
Parsing
Product feature candidate selection
Product feature extraction by using ME model
Postprocessing
Product features

Figure 2.6: Product feature extraction process (Somprasertsri & Lalitrojwong, 2008)

2.4 Conclusion

There are various natural language processing technologies that can be used to conduct the task of information extraction from unstructured natural language texts. Especially part-of-speech tagging establishes very valuable possibilities to identify specific information of interest within natural language texts. A major goal of this work is to extract useful product features from customer reviews in the domain of tourism. The introduced state-of-the-art approaches that use, amongst other things, parts-of-speech to identify and extract product features from user comments and reviews have shown that part-of-speech tagging is very useful tool to fulfill this task of information extraction. To identify product features from customer reviews written in natural language the introduced state-of-the-art approaches try to identify sequences of nouns or noun phrases. Additionally Somprasertsri and Lalitrojwong (2008) consider the frequency of noun phrases that occur in the used corpus to infer if the certain noun phrases are likely to be product features or not. Zhan and Li (2010) observe the semantic neighbors of noun phrases for the noun phrase classification. To perform opinion mining Zhan and Li (2010) utilize certain verbs such as "like" that occur together with noun phrases.

The approach to product feature extraction proposed in this work does not have to consider opinion mining or sentiment analysis as only positive ratings with almost 100% customer recommendation are used for the product feature extraction task. Moreover the approach proposed in this work does not aim to infer opinions and ratings about specific products but tries to find useful product features that correspond to the preferences of the customer. Extracting noun phrases consisting of adjective-noun pairs or noun-noun pairs using part-of-speech tagging has shown to be very useful in this task. Additionally certain verbs that occur together with noun phrases are regarded. In comparison to the approach of Zhan and Li (2010) these verbs are not used for opinion mining but for detecting the correlations between adjectives and verbs. Similar to the approach of Somprasertsri and Lalitrojwong (2008) the frequency of occuring noun phrases is considered and recorded but in a post-processing manner to supply the supporting recommendation engine with additional information when performing the collaborative filtering tasks.

String matching with wildcards which was also briefed in this Chapter (see Section 2.2.2) allows a looser matching of strings. This technology has exposed to be very useful to match the different items of interest (in this case requirement terms articulating what the active user wants) between the term formulations entered via the interface of the existing personalization system Xohana and the term (noun phrase) database of the Prototype. This matching of strings with wildcards is also used for the query suggestion method mentioned in Chapter 1. Chapter 4 provides more details about how the above mentioned outcomes have been deployed in the prototypical implementation.

The research topics and state-of-the-art approaches that were discussed in this Chapter provide a valuable knowledge base to develop strategies for product feature extraction from customer reviews that are written in natural language. Based on the extracted product features a database can be built that can be applied by the recommender system prototype to generate supplementary recommendations in order to support the existing personalization system Xohana. To implement the tasks of the recommender system prototype, efficitive and appropriate recommendation strategies have to be developed. The next chapter provides an overview of established recommender system approaches and in-depth research about the topics that are relevant to this work to determine what technologies and techniques are adequate to fulfill the tasks of the recommender system prototype.

3. Recommender Systems

This Chapter provides an overview of the most popular types of recommender systems and more detailed information about the sub-topics of recommender systems research and current approaches that are related to this work. To get a better idea of this relatively new research field the major goals and motivations for developing and using recommender systems as well as the general components and available knowledge sources of recommender systems are explained at first. Afterwards, the functionality of the different types of recommendation techniques together with their advantages and drawbacks are discussed. As this work aims to support existing personalization systems by providing recommendations using collaborative filtering this type of recommendation technique and the most fundamental collaborative filtering methods are discussed in more detail. Another goal of this work is to provide another solution to overcome the well-known cold-start problem. Therefore different State-of-the-art approaches addressing this special challenge to recommender systems are presented in the last section of this Chapter.

3.1 Motivation for Recommender System Research

Compared to other research fields of established information system techniques and tools, recommender systems as an independent research area is relatively new and emerged in the mid-1990s (Ricci et al., 2010). There are several facts that reflect the increasing interest in the topic of recommender systems (Ricci et al., 2010):

- For many successful Internet sites such as YouTube, Netflix, Amazon.com, Tripadvisor, IMDb and Last.fm recommender systems are very important and on top of that many media enterprises are developing recommender systems and offer them as a service to their users.

- Workshops and dedicated conferences that are related to the research area of recommender systems have emerged, e.g. the ACM Recommender Systems (RecSys) that was founded in 2007 and which has become the premier annual meeting addressing recommender system applications and technologies.
- Graduate and undergraduate courses at higher education institutions completely attend to the topic of recommender systems.
- Several academic journals such as AI Communications (2008), IEEE Intelligent Systems (2007), International Journal of Electronic Commerce (2006) dealt with issues dedicated to the research field of recommender systems.

3.2 Goals and Tasks for Recommender Systems

Depending on the point of view, there are different tasks and goals for applying recommender systems. A user of a recommender system that want's to identify the information that is most valuable for him or her or to find the most interesting item has different requirements than the service provider of the used recommender system. This Section outlines the various motivations, goals and tasks for the different roles that participate in the recommendation process.

3.2.1 User Goals and Tasks for Recommender Systems

Herlocker, Konstan, Terveen, and Riedl (2004) and Ricci et al. (2010) list and descibre some of the domain-independent end-user goals and tasks:

- Annotation in Context: Within a certain context, such as a list of items, the system tries to emphasize some of these items dependent on the long-term preferences of the user. For instance, a television recommender system could comment what TV shows or movies shown in the electronic program guide (EPG) could meet the user's taste. (Ricci et al., 2010)
- Find some good items: To fulfill this core recommendation task, that recurs in several commercial systems and research, users are provided with a ranked list of recommended items. Whereas it is common to show prediction values indicating which items the user would most likely prefer, in several commercial systems the predicting values are hidden and only the "best bet" recommen-

dations are provided. (Herlocker et al., 2004)

- Find all good items: If the recommender system is mission-critical, such as in financial or medical applications it may be inadequate to recommend only some good items (Ricci et al., 2010). For example, for lawyers that are searching precedent cases it can be very crucial not to miss a single possible case. In that regard it is very important not overlook any item and that the user can make sure that the rate of false negatives becomes low enough. Hence, coverage plays an important role in this task. (Herlocker et al., 2004)
- Recommend a sequence: Sometimes it is more efficient to recommend a series of items that is satisfying as a whole rather than concentrating on the creation of a single recommendation. Typical examples of this task include recommending a compilation of pop songs, recommending a TV series or similar. (Ricci et al., 2010)
- Just Browsing: This task reflects situations where a user is browsing the available items without an ulterior motive or any purchase imminent. In this context the interface, the nature and level of information that is provided by the specific system and the ease of use is more important than the accuracy of the applied algorithms. (Herlocker et al., 2004)
- Find credible recommender: Because certain users do not trust recommender systems they give them a trial and primarily play with them to see how good the generated recommendations are. Due to this, some recommender systems offer additional specific functions that allow the users to test the behavior of the system. (Ricci et al., 2010)
- Improve the profile: This rating task that is assumed by most recommender systems is based on the assumption that users believe that they are enhancing their profile providing information about what they prefer by contributing ratings on items. With that the quality of the recommendations they will receive can be improved. (Herlocker et al., 2004)
- Express self: For some users it is important to express their opinions by contributing their ratings. Very often these users do not care about receiving useful recommendations - they post reviews because it feels good. This effect, that particularly emerges on sites such as Amazon.com, can lead to the provision of more data and with that to the improvement of the quality of

recommendations. (Herlocker et al., 2004)

- Help others: Because some users think that a community profits from contributions they made, they are happy to provide ratings in the recommender system. (Herlocker et al., 2004)
- Influence others: In web-based recommender systems there might be some users, that try to manipulate other users with the goal to explicitly influence them to purchase specific products (Herlocker et al., 2004).

3.2.2 Service Provider Goals and Tasks for Recommender Systems

Ricci et al. (2010) list the goals and tasks for recommender systems from the service provider's, marketer's and other system stakeholder's point of view as follows:

- Increase the number of items sold: Increasing the conversion rate, for example increasing the number of users that consume an item and accept the recommendation, in comparison to the number of users that simply navigate through the available information, can be seen as the primary goal of using a recommender system. This goal is reflected by this task.
- Sell more diverse items: A further main function of a recommender system is to support the user in finding items that might not be easily found without receiving an accurate recommendation.
- Increase the user satisfaction: Combining accurate and effective recommendations and an interface featuring a good usability can increase the user's subjective system evaluation is another important task from the service provider's point of view. Thereby the system usage and the likelihood that the provided recommendations will be approved can be increased.
- Increase user fidelity: A web site should recognize a loyal user as an old and valuable customer. Computing recommendations can be done by using information that is obtained from previous user interactions, for example item ratings. The longer a user interacts with the recommender system the better the system can represent the preferences of the user. Thus the user model becomes more refined and more customized recommendations that match the user's taste or preferences can be generated.

- Better understand what the user wants: Gathering information about the user's preferences, either predicted by the system or collected explicitly is another crucial function of a recommender system. The knowledge originating from this kind of information can be reused for different other tasks, such as enhancing the management of the production or item's stock.

3.3 Recommender System Data Objects

Ricci et al. (2010) mention three various kinds of data objects that are relevant in order to build recommendations: items, users, and transactions, that is relations between users and items. According to the implemented recommendation algorithms, data used in recommender systems can be very simple and knowledge poor, e.g. user evaluations or ratings for items, or very knowledge intensive. For example, these kinds of techniques use social activities and relations of the users, constraints or ontological descriptions of the items or the users.

3.3.1 Items

Items are referred to as the objects that are suggested by the recommender system. An item can be characterized by its value or utility and its complexity. Depending on how useful the item is to the user, it may be assigned a positive value or a negative one, as far as the selection of the item was a wrong a decision and it is not applicable for the user. When developing a recommender system the complexity and value of the existing items has to be considered. There are items with low value and complexity, such as books, CDs, movies, web pages and items with a larger value and complexity, such as mobile phones, PCs, digital cameras and so on. Items that have been considered to be the most complex are travels, jobs, financial investments and insurance policies. (Ricci et al., 2010)

Depending on the technology of a recommender system, it may use various features and properties of the existing items. For instance, a recommender system for movies would use features such as the genre of the movie, the director, the producer, the actors, to describe an item. To represent items different representation approaches and information may be used, for example, a set of attributes, a single id code or even a concept in an ontological description of the domain. (Ricci et al., 2010)

3.3.2 Users

As users have different characteristics and goals it is important to gather a sufficient amount of information about them, thus to personalize the interaction between human and computer and the recommendations. Again, depending on the recommendation technique this kind of user information can be modeled and structured in different ways. No matter what user modeling approaches are applied by a recommender system, a convenient user model is inevitable to provide personalized recommendations. For example, in collaborative filtering approaches user data is modeled using an ordinary list of ratings the user provided for some items. Some kinds of recommender systems describe users by their behavior pattern data. For instance, a travel recommender system may use travel search patterns and a web-based recommender system often uses browsing patterns to describe the users of the system. (Ricci et al., 2010)

3.3.3 Transactions

Ricci et al. (2010) define a transaction as a *"a recorded interaction between a user and the RS"*. Transactions record important informations that are created while the user interacts with the recommender system. The collected informations are very useful for the applied recommendation algorithm. For example, a transaction log may include a description of the given context (for instance, the user query or the goal of the user) for that specific recommendation and a reference to the item the user selected. Additionally, if present, a transaction can also contain an explicit feedback that was provided by the user. The most established transaction data collected by recommender systems are ratings, which are kind of explicit feedback. Collecting these ratings can be done in an explicit or implicit manner. To explicitly collect ratings, the user is requested to provide her view about an item using a rating scale. (Ricci et al., 2010)

User ratings can take different shapes (Ricci et al., 2010):

- Numerical ratings: For example, the 1-5 stars used by the book recommender system employed by Amazon.com
- Ordinal ratings: If ordinal ratings are applied, the user is asked to choose one of various terms provided by the system, such as "strongly disagree, disagree, agree, strongly agree", to express his or her opinion.
- Binary ratings: The user is simply asked if he likes or dislikes a certain item.
- Unary ratings: The user either purchased or observed an item, or provided a positive rating for it.

To implicitly collect ratings the recommender system tries to conclude the opinion of the user analyzing the user's actions. For instance, if a user searches for a book by the keyword "Cooking" at Amazon.com he or she will receive a list of books. If the user chooses an item of the provided list to get more information about it, the system may assume that the user is in some way interested in that item. (Ricci et al., 2010)

3.4 Knowledge Sources of Recommender Systems

To fulfill the recommendation task recommender systems - like all intelligent systems - use various types of knowledge sources. On the one hand the knowledge utilized by a recommender system can be implicit, such as the knowledge that is encoded in an algorithm or collected information about user estimations over a group of items. On the other hand recommender systems can use deductive knowledge that is explicitly encoded or ontological knowledge. According to Felfernig and Burke (2008) knowledge that is required to generation recommendations can originate from four sources:

- From the users themselves
- From other peer users in the system
- From data about the items that are recommended
- From the domain of recommendation itself (that is knowledge about what needs are accomplished by the recommended items and how they are used)

Built on this distinction of the origination of knowledge sources a *taxonomy of knowledge sources* can be modeled which is shown in Figure 3.1. Generally speaking the entity *"Content"* may be referred to as any type of knowledge that is not generated by the user. (Felfernig & Burke, 2008)

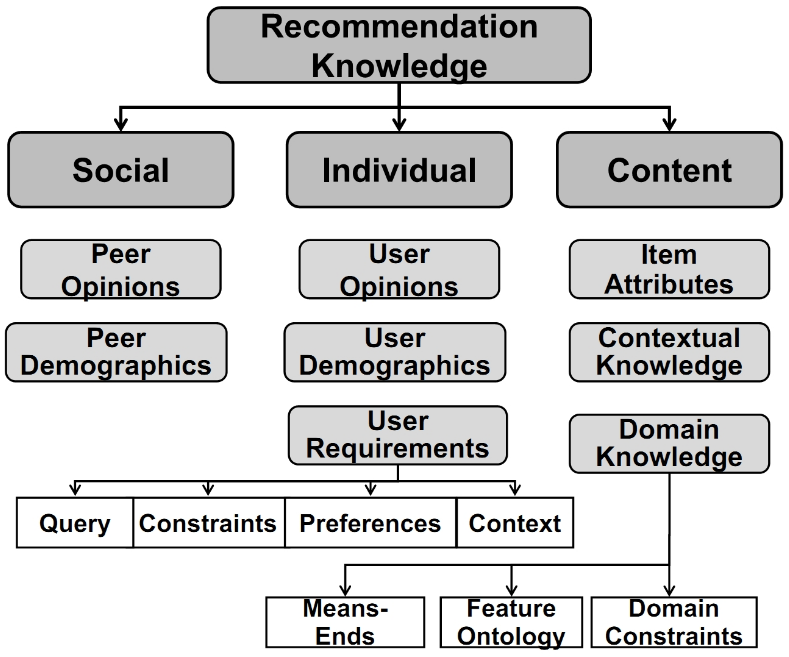


Figure 3.1: A taxonomy of knowledge sources in recommender systems. (Felfernig & Burke, 2008)

3.5 Overview of Recommendation Techniques

As already mentioned in Section 3.5 recommender systems apply different recommendation techniques. This chapter provides a brief description of the most established recommendation techniques. *Collaborative Filtering* recommender systems will be discussed in more detail in Section 3.7. Burke (2007) divides recommendation techniques into four different classes based on their knowledge sources (see Figure 3.2):

- Collaborative
- Content-based
- Demographic
- Knowledge-based

Hybrid approaches combine logic or components of different types of recommendation techniques (Burke, 2007). In addition to the recommendation techniques

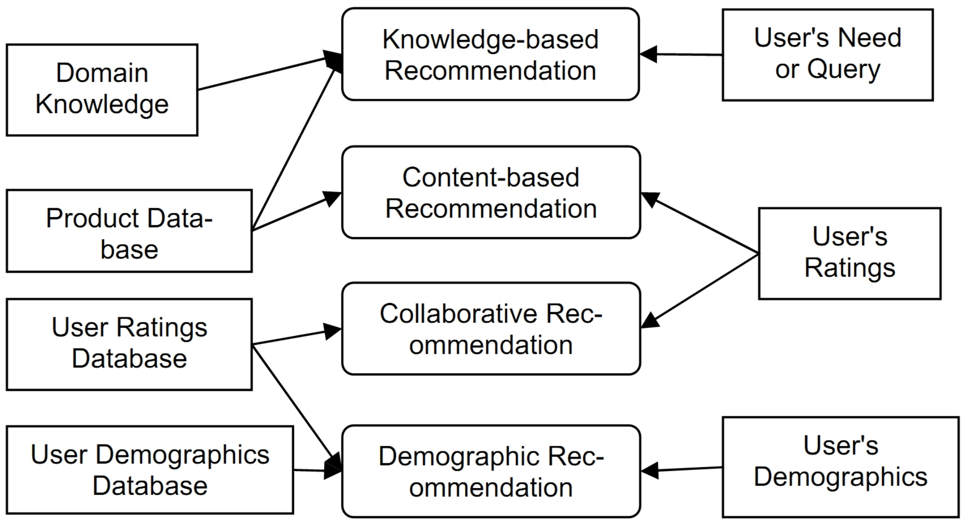

Domain Knowledge
Product Data-base
User Ratings Database
User Demographics Database
Knowledge-based Recommendation
Content-based Recommendation
Collaborative Rec-ommendation
Demographic Rec-ommendation
User's Need or Query
User's Ratings
User's Demographics

Figure 3.2: Classification of recommender systems based on their knowledge sources. (Burke, 2007)

mentioned above, Ricci et al. (2010) discuss community-based recommendation techniques and context-aware approaches. These types of recommendation techniques are only named here for the sake of completeness.

3.5.1 Content-based Recommendation Techniques

The basic idea behind content-based filtering recommender systems is that if a user preferred certain items in the past he or she would probably prefer items that are similar in the future. So as to predict the preferences of a user content-based filtering techniques determine item characteristics and compare them with the user's profile of interest. (Shih & Liu, 2005) Content-based recommendation techniques aim to learn a certain classification rule for each user in the system, in order to predict as likely as possible if a certain existing item is of the user's interest or not. To enable this, the classification rule is learned on the basis of the attributes of each item and the user's rating information. (Felfernig & Burke, 2008)

Content-based recommender systems have a quite good accuracy and do not suffer from the *new item problem* (see 3.8.1) because they have access to item features, such as category information or keywords (Blanco-Fernandez et al., 2008; Felfernig & Burke, 2008). Nevertheless, the content-based filtering technique itself is limited by the applied similarity metrics and still has to face the *new user problem* (see

3.8.2), since adequate user profiles have to be build up through the aggregation of a sufficient number of ratings, depending on the learning algorithm and the feature set (Blanco-Fernandez et al., 2008; Felfernig & Burke, 2008). Additionally, as mentioned by Blanco-Fernandez et al. (2008), traditional content-based filtering approaches are at risk to suffer from over-specialization (see 3.6.4).

3.5.2 Collaborative Filtering Recommendation Techniques

The main goals of collaborative filtering approaches concentrate on generation and providing predictions or item recommendations that are based on the opinions and preferences of other users with similar tastes. User opinions can be either received explicitly by the user, expressed through rating scores (usually based on numerical scales), or implicitly by mining web hyperlinks, analyzing timing logs, purchase records and so forth. (Sarwar, Karypis, Konstan, & Reidl, 2001)

The term *"collaborative filtering (CF)"* was firstly used by the developers of Tapestry, one of the first recommender systems. Collaborative filtering grounds on the fundamental assumption that if two users rated a certain number of items similarly or if they acted similar, for instance in watching or buying items, they might have similar interests in other items. (Su & Khoshgoftaar, 2009) Section 3.7 provides deeper insights about collaborative filtering approaches.

3.5.3 Demographic Recommendation Techniques

According to Nageswara Rao and Talwar (2008) demographic recommender systems seek to learn relationships between a certain item and the type of users who preferred it. Recommendations are generated on the basis of previously obtained knowledge on demographic information about users and their opinions and ratings on suggested items. As demographic recommender systems assume that all users related to a specific demographic group are sharing the same tastes and preferences, they are stereotypical. For example, LifeStyle Finder, tries to assign the active user to one of several pre-existing clusters of demographic user groups to suggest web pages and items based on information about users belonging to that cluster. Similar to collaborative filtering techniques demographic recommender systems try to build correlations between users, however, demographic techniques use different data. (Nageswara Rao & Talwar, 2008)

In comparison to collaborative filtering and content-based filtering techniques demographic techniques do not require user rating histories and provide a straightforward and fast approach for making assumptions that are based on limited observations. Additionally the implementation of such a system can be done easy and quick. Nevertheless demographic recommendation techniques have some drawbacks. The efficiency of demographic filtering depends on the completeness of the collected demographic information about the user. As the gathering of such information concerns issues about privacy this could be a very difficult task. Beyond that, despite from suffering from both the new item problem (see 3.8.1) and the new user problem (see 3.8.2), the provided recommendations are too general, since the demographic clusters used to categorize users are based on a generalization of user preferences and users with similar demographic profiles are recommended the same items. (Nageswara Rao & Talwar, 2008)

3.5.4 Knowledge-based Recommendation Techniques

Knowledge-based recommendation techniques have the ability to draw conclusions about how a specific item conforms to a specific user requirement by making some kind of inference and using functional knowledge, that is knowledge about the correlation between a particular user need and a possible recommendation. The inference made by this recommendation technique is supported via the user profile which can be any kind of knowledge structure ranging from a very simple representation such as a query the user has formulated, as used in Google, or a more fine-grained representation of the user preferences and needs. (Fürnkranz & Hüllermeier, 2010, p. 395)

As traditional recommendation techniques, such as collaborative filtering and content-based filtering approaches, are appropriate to make recommendations of products depending on quality and taste, such as news, movies or books, they are not that suited to recommend items that are not purchased and with that rated very frequently, such as apartments, financial services, computers or cars. (Ricci et al., 2010) Knowledge-based recommender systems do not have to collect information about a specific user because the generation of recommendations is not based on individual tastes and user ratings (Burke, 2000).

Knowledge-based recommendation approaches try to gather deep knowledge about the specific product domain and to exploit concrete user requirements to calculate recommendations and user requirements are directly determined during a recommendation session. Therefore these approaches do not suffer from cold-start problems (see 3.8) like collaborative filtering and content-based filtering recommender systems. Nevertheless, even knowledge-based recommender systems have a certain drawback: they suffer from a problem called knowledge acquisition bottleneck. Knowledge-based recommendation techniques use knowledge based on knowledge occupied by domain-experts and it is very challenging and expensive for knowledge engineers to transform these knowledge bases into a formal, executable representation. (Ricci et al., 2010)

Ricci et al. (2010) divide knowledge-based recommender systems into two basic types: constraint-based and case-based recommenders. Although both approaches use more or less the same kind of knowledge they differ in the way recommendations are calculated. Constraint-based recommender systems principally use pre-existing recommender knowledge bases that provide specific rules about how item features have to be associated with user requirements. In turn, case-based recommendation techniques generate recommendations based on similarity metrics. (Ricci et al., 2010)

3.5.5 Hybrid Approaches

Hybrid recommender systems try to combine two or more of the recommendation techniques mentioned above. On the one hand hybrid systems aim to join the advantages of the used recommendation techniques and on the other hand they try to eliminate their drawbacks. Several approaches have been proposed for creating new hybrid systems. For example, collaborative filtering approaches suffer from the new item problem which means they are not able to recommend items that have not been rated yet. A hybrid system combining collaborative filtering and content-based filtering techniques could address this problem, since recommendations of new items made by content-based techniques are based on item features - which usually can be easily obtained - and not on user ratings on these items. (Ricci et al., 2010)

3.6 Major Challenges and Problems of Recommender Systems

Depending on the various recommendation techniques different challenges and problems arise within this topic. Nevertheless, there are some problems that nearly all types of recommender systems have to deal with. Ghazanfar and Prugel-Bennett (2010) and Devi, Samy, Kumar, and Venkatesh (2010) list scalability, data sparsity and cold-start problems as the major problems of recommender systems. Another typical problem of recommender systems, especially concerned with content-based recommendation techniques, is called *"Over-specialization"*. (Shahabi, Banaei-Kashani, Chen, & McLeod, 2001) The following sections provide a brief description of the mentioned challenges and problems. As this thesis introduces an alternative solution to address the cold-start problem, this topic will be discussed in more detail in Chapter 3.8.

3.6.1 Scalability

With respect to recommender systems scalability issues include very large problem sizes as well as real-time latency requirements. For example, a recommender system applied by a frequently-used web site, has to be able to generate recommendations within a split of a second and to serve a huge amount of users at the same time. (Schafer, Konstan, & Riedl, 2001)

Recommender systems applying collaborative filtering techniques create recommendations based on a subset of users with the highest similarity to active user. For every new recommendation that is provided to the active user, the similarity to all other existing users has to be recomputed to identify users with similar preferences and behavior. In most cases collaborative filtering recommender systems have serious problems to scale up the computation process with an increase of the number of both items and users. (Papagelis, Rousidis, Plexousakis, & Theoharopoulos, 2005)

Suppose a system managing tens of millions of users and millions of different catalog items. Applying a collaborative filtering algorithm with a complexity of O(n) would go beyond acceptable and practical levels. Moreover many recommender systems need to be capable of providing recommendations to new online requirements independently of the rating history or purchase information of active users. To fulfill these requirements collaborative filtering recommender systems need to have a high scalability. (Su & Khoshgoftaar, 2009)

3.6.2 Sparsity

The sparsity problem, which is the most obvious one recommender systems do suffer from, occurs if information about correlations between customers and the number of existing ratings on items are sparse. In that case, computing the degree of similarity between users without having enough ratings on items, is almost impossible. (Bergholz, 2003) As each user usually only rates a few number of items, the user/item matrix tends to be very sparse. It is a very challenging task to compute accurate similarities between users and to make effective predictions of ratings if the number of reviews is limited. (He & Chu, 2009) Amongst others, an approach addressing the problem of sparse rating matrices is Singular Value Decomposition (SVD), a dimensionality reduction technique, that enables the dimensionality reduction of sparse rating matrices by removing insignificant or unrepresentative items or users (Su & Khoshgoftaar, 2009).

3.6.3 The Cold-start Problem

The cold-start problem is a special case of the data sparsity problem. It emerges when a new item or user has just entered the recommender system and there is not enough information - that is there are not enough ratings on that item or respectively the new user has not rated enough items - to find similar ones. (Su & Khoshgoftaar, 2009) In literature the cold-start problem is also referred to as *new user problem* or *new item problem* (Su & Khoshgoftaar, 2009; Devi et al., 2010). The sparsity of ratings of users or ratings on items significantly lowers the accuracy of prediction in collaborative filtering and makes it very difficult to find users similar to the active user (Devi et al., 2010). Section 3.8 provides further information about this topic.

As the approach proposed in this work uses collaborative filtering techniques in cold-start situations, Section 3.8 gives and an overview of state-of-the-art techniques addressing cold-start problems, escpecially in collaborative filtering.

3.6.4 Over-specialization

Content-based recommendation techniques suffer from the over-specialization problem, which occurs when the system can solely provide recommendations on items that are highly similar to items the user liked in the past. Thus the user can never explore surprising items, because he or she is restricted to those he or she already rated in the past. (Iaquinta et al., 2008) Since, in the most cases, content-based recommender systems apply syntactic similarity matching techniques, they lack of knowledge about the user's preferences and semantic understanding. Thus, the quality of personalization is strongly limited. (Pan, Wang, Horng, & Cheng, 2010)

For example, a movie, such as "X-Men Origins: Wolverine", would be recommended to a fan of Science-Fiction action/thriller movies with a high relevance. But since the user is very likely to already know about the recommended item, the recommendation would be less useful. (Abbassi, Amer-Yahia, Lakshmanan, Vassilvitskii, & Yu, 2009)

3.7 Collaborative Filtering Approaches

As mentioned in Chapter 1 this work aims to support the recommendation process of an existing personalization system by applying collaborative recommendations. Therefore the advantages and disadvantages of collaborative filtering techniques will be discussed in more detail in the following sections to obtain more insights about this topic. According to Ghazanfar and Prugel-Bennett (2010) can be divided into two major categories: memory-based collaborative filtering and model-based collaborative filtering. Before these two kinds of collaborative filtering methods are discussed the basic principles of collaborative filtering will be introduced.

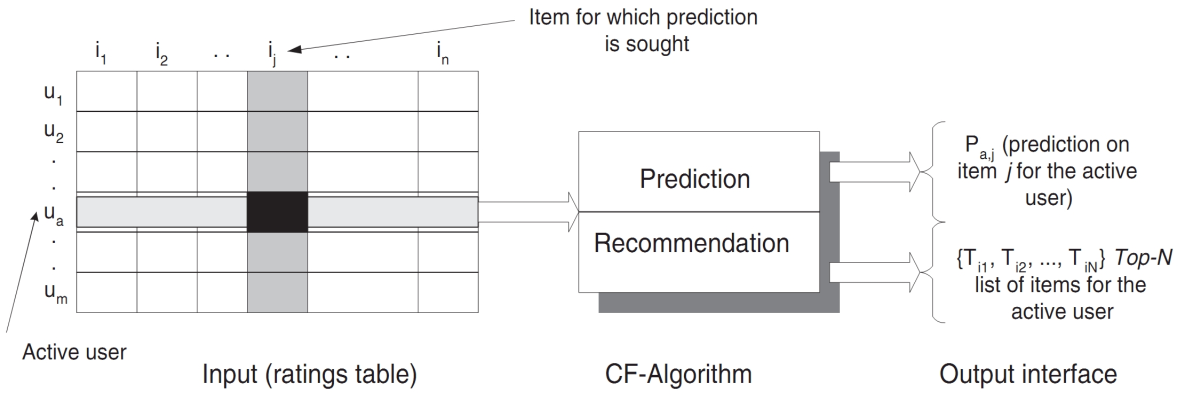
3.7.1 Basic Principles

Over the years more and more websites started to apply collaborative filtering techniques to provide recommendations to their customers in various domains, such as information, entertainment, art, grocery products and books (Good et al., 1999). Collaborative filtering aims to provide recommendations for a certain user based on the opinions and ratings of other users that are like-minded (Sarwar et al., 2001). Cosley et al. (2002) mention that to identify the most highly correlated users (or users having a similarity score that satisfies a certain threshold) and to choose the most similar neighbors, various similarity metrics, such as mean squared difference, vector similarity and Pearson correlation, have been used so far. The item ratings of the identified neighbors that are the most similar to the current user are then used to calculate rating predictions on unseen items. After that the user is presented a list of items, ordered by the predicted item ratings, that he or she has not seen yet. (Cosley et al., 2002) Generally the focus of collaborative filtering is on answering two questions: *"Which items (overall or from a set) should I view?"* and *"How much will I like these particular items?"* (Good et al., 1999).

Schafer, Frankowski, Herlocker, and Sen (2007) state three major tasks of common collaborative filtering systems:

1. Recommend items: The user is shown a list of items, that might be useful. (Schafer et al., 2007) This practice is known as Top-N recommendation (Sarwar et al., 2001). The list is often ordered by the predicted item ratings. Some systems, such as the recommendation engine of the Amazon online store, do not calculate and display personalized predicted ratings. Instead, they display the average customer ratings. (Schafer et al., 2007)
2. Predict for the given item: This task aims to compute the predicted rating for a particular item (Schafer et al., 2007). The predicted rating value for the particular item corresponds to possible opinion values (e.g. within a given rating scale from 1 to 5) (Sarwar et al., 2001). In comparison to rating prediction providing recommendation is less challenging because only some alternatives have to be prepared to be suggested to the user. The computation of personalized predictions becomes even more difficult, if not impossible, if there are too few ratings on that items. (Schafer et al., 2007)
3. Constrained recommendations: This task aims to recommend items from a certain set of items that is given by certain constraints (Schafer et al., 2007).

Figure 3.3 taken from the work of Sarwar et al. (2001) provides a schematic graphical representation of the general collaborative filtering process. Usually collaborative filtering algorithms maintain the entire user-item data (mxn) as a ratings matrix (A), where each entry ($a_{i,j}$) in the matrix expresses the preference score (rating value) of user i on item j. All individual user ratings on the existing items are within a specific numerical range, given by the used rating scale, where the value 0 indicates that item j has not yet been rated by user j. (Sarwar et al., 2001)



Item for which prediction is sought
i1 i2 .. ij .. in
u1 u2 . . ua . . um
Active user
Prediction
Recommendation
Pa,j (prediction on item j for the active user)
{Ti1, Ti2, ..., TiN} Top-N list of items for the active user
Input (ratings table)
CF-Algorithm
Output interface

Figure 3.3: The collaborative filtering process. (Sarwar et al., 2001)

3.7.2 Memory-based Approaches

Memory-based approaches make use of the complete or a selection of the user-item database to calculate predictions (Su & Khoshgoftaar, 2009; Sarwar et al., 2001). Each user in the system is considered to belong to a specific group of people that share similar interests and preferences. To generate predictions about how much a new user or the active user will like items he or she has not seen yet, the system identifies the so-called neighbors of the user. (Su & Khoshgoftaar, 2009) The identification of this set of like-minded users is often implemented by employing statistical techniques (Sarwar et al., 2001).

Although memory-based approaches usually generate quite accurate predictions and benefit from the ability of quickly incorporating the latest information, they are challenged by some problems, such as poor scalability (because searching the entire database for similar customers is very time-consuming in large databases) and data sparsity problems (which may in turn strongly decrease the prediction accuracy) (Xue et al., 2005; Yu, Wen, Xu, & Ester, 2001).

A very well established memory-based collaborative filtering algorithm is the neighborhood-based approach, which works as follows: In a first step similarities or weights, that represent the correlation, distance, or weight ($w_{i,j}$) between to items or users (i and j) are calculated. Afterwards the prediction for the active user is generated by using a simple weighted average or the weighted average of all recorded ratings of items or users on a particular item or user. To produce so-called top-N recommendations (a list of N top-ranked items potentially interesting to the active user), the task is to compute the similarities and to identify the nearest neighbors (the k most similar items or users). The determined nearest neighbors are then aggregated to obtain the top-N most frequent items, which can be recommended to the active user. (Su & Khoshgoftaar, 2009)

Most user-based approaches compute the similarity of two users based on ratings on items that both of them have rated. (Adomavicius & Tuzhilin, 2005) To calculate the similarity between two items item-based approaches at first determine all users that rated both items. Then different similarity calculation techniques can be applied to compute the similarity between the items (Sarwar et al., 2001).

Item-based and user-based collaborative filtering techniques apply similarity calculation methods, such as correlation-based similarity measures that apply the Pearson correlation or other similarity metrics to calculate the similarity between two users or items and Vector Cosine-Based similarity. For user-based approaches, predictions for the active user on a particular item can then be calculated by using the weighted average of all ratings on that certain item made by other users. For item-based approaches rating predictions for a user on a certain item can be calculated by using simple weighted average based on the summation of all other rated items, where the weightings between the items and the particular item are taken into account. (Su & Khoshgoftaar, 2009)

User based Top-N Recommendation

Karypis (2001) describe the functionality of user-based top-N recommendation algorithms as follows: At first the nearest neighbors (k most similar users) of the active user are determined, which is often performed by using the vector-space model. Each user inside this model is represented as vector within the entire m-dimensional item-space. To measure the similarity between the active user and the other n users, the cosine between the vectors of the active and the other n users is calculated. After that the rows in the mxn user-item matrix that correspond to the identified nearest neighbors are aggregated to obtain the set of items (and the related frequencies) that have been purchased by that nearest neighbors. Given the resulting list of candidate items the N most frequent items in that list the user has not seen yet are recommended. (Karypis, 2001)

Item based Top-N Recommendation

To calculate the top-N list of recommendations item-based approaches focus on identifying relations between different items by analyzing the user-item matrix. As these approaches do not need to identify the nearest neighbors of users they perform faster than user-based approaches. Item-based top-N algorithms that calculate relations between items using item-to-item similarity work as follows: At first, during a model building phase, the k most similar items for all existing items are calculated and all corresponding similarities are stored. To provide the top-N recommendations for the active user, the algorithm builds the union set of the k most similar items for all items the active user has purchased so far. Then all items in the union set that have already been purchased by the active user are removed. For each remaining item in the union set the similarity between that item and the set of items already purchased by the active user is calculated. This results in a list of items sorted in descending order of which the first N items are selected as the top-N items to be recommended to the active user. Item-based recommendation algorithms have evolved to address scalability problems user-based recommendation algorithms have to face. (Karypis, 2001)

3.7.3 Model-based Approaches

Model-based collaborative filtering approaches use the recorded user preferences to learn a model which can then be used to calculate predictions (Adomavicius & Tuzhilin, 2005; Yu et al., 2001). The building process of the model can be done off-line which can take several hours or days, but the generated model is usually very fast, small and basically as accurate as memory-based collaborative filtering approaches (Yu et al., 2001). With the developed model the system is able to detect complex patterns based on training data. Afterwards smart predictions for real-word or test-data can be calculated using the learned model (Su & Khoshgoftaar, 2009).

Su and Khoshgoftaar (2009) describes the general functionality of model-based approaches as follows: In a first step the different users of the training database are clustered into small groups or classes by regarding their rating patterns. In a next step the system assigns the active user to one or more of the previously defined user classes to calculate predictions on a particular item by using ratings of the identified classes on that particular item. Model-based approaches apply different methods, such as Bayesian models, clustering techniques, regression based collaborative filtering algorithms and latent semantic collaborative filtering models. Bayesian belief networks, such as Simple Bayesian algorithms and baseline Bayesian model can be used for classification tasks in collaborative filtering. (Su & Khoshgoftaar, 2009)

Clustering approaches are grouping users with similar preferences into clusters. Using the given clusters the system is able to calculate predictions for the active user by averaging the ratings of the other users that are assigned to that cluster. Clustering approaches that enable the assignment of users to several clusters calculate predictions as an average over the clusters in which the user participates which is weighted with consideration of the level of participation in each cluster. (Xue et al., 2005) As clustering models make predictions within the relatively small clusters instead of the complete user dataset, they have a good scalability. Nevertheless systems applying clustering methods have to make trade-offs between good scalability and prediction performance. (Su & Khoshgoftaar, 2009)

Regression based methods calculate predictions based on a regression model by approximating user ratings (Su & Khoshgoftaar, 2009). Latent semantic collaborative filtering techniques aim to determine prototypical interest profiles and user communities by applying a statistical modeling approach that uses latent class variables in a mixture model setting. This approach analyzes user preferences by the means of overlapping user preferences. Latent semantic collaborative filtering techniques usually outperform common model-based approaches in scalability and accuracy. (Su & Khoshgoftaar, 2009) The aspect model which is a probabilistic latent-space model views individual user ratings as a convex combination of rating factors (Xue et al., 2005; Su & Khoshgoftaar, 2009). The latent class variable is associated with all observed user-item pairs assuming that item and user are not dependent from each other considering the latent class variable (Xue et al., 2005).

For systems in which the user preferences do not change very frequently model-based approaches show good performance, but they are not very applicable if the user preference models have to be updated very often or rapidly (Yu et al., 2001).

3.7.4 Advantages and Drawbacks of Collaborative Filtering Recommender Systems

According to Martinez, Perez, and Barranco (2009) there are several advantages collaborative filtering systems do benefit from:

- Knowledge domain is not needed: The used databases of collaborative filtering systems do not need any information or knowledge about the existing items or products.
- Explicit feedback is not required: Although, most collaborative filtering approaches prefer explicit ratings, because more trustworthy and accurate information is made available, implicit feedback obtained from user actions is sufficient to make recommendations.
- Adaptive system: The quality of collaborative filtering systems grows according to the number of users and item ratings.
- Recommendations "outside the box": Items, that are very unlike to other items the user already was interested in, will be recommended too.

Despite the mentioned benefits, collaborative filtering is also challenged by various problems, such as scalability, sparsity and cold-start problems (which are special types of sparsity problems), that have already been discussed in Section 3.6. The cold-start problem and state-of-the-art approaches addressing this problem are discussed in Section 3.8 and Section 3.9 respectively.

Martinez et al. (2009) name two additional drawbacks of collaborative filtering recommender systems:

- The "gray sheep" problem: The system can't provide solid ratings for users that can't be assigned to a specific group.
- Historical data set: The learning aptitude of collaborative filtering approaches can be advantageous, but also disadvantageous at the system startup when there is only a small historical data set.

Synonymy is another potential problem for collaborative filtering approaches. Very similar items that are termed differently (different entries or names), for instance *"children film"* and *"children movie"* could actually represent the same item. However, memory-based collaborative filtering approaches can not recognize them to be the same. Other problems are *shilling attacks*, which occur when users make huge amounts of good ratings to promote specific items and huge amounts of negative ratings to discredit competing items. (Su & Khoshgoftaar, 2009)

3.8 Cold-start Problems

One major goal of this work is to support an existing recommender system with additional personalized recommendations when the system is in a cold-start situation. To obtain more insights about this topic cold-start problems are discussed in this Section in more detail. As stated before in Chapter 1, cold-start problems occur when a new user or a new item enters the recommender system. The next two sections describe the new user cold-start problem and the new item cold-start problem.

3.8.1 New Item Problem

In most recommender systems items are added with a certain regularity. So, again and again, there exist new items that initially are completely unrated. As collaborative filtering recommender systems have to rely on information about user preferences and with that on information about what items have been rated by the users, an item can not be recommended until there is a sufficient number of user ratings on that item. This kind of cold-start problem is called new item problem. (Adomavicius & Tuzhilin, 2005)

Because users rather tend to rate items they are recommended, collecting ratings on new items is even more difficult. Although there are possibilities to get ratings on most good items, for example using non-collaborative filtering techniques, in some domains that have a high item turnover, for instance in systems providing news articles, this problem can be notably tedious. Even if users are often tolerant to systems that do not suggest obscure items, in some domains there may be many very good items that remain hidden, because they are not rated with an adequate amount. (Schafer et al., 2007) Schafer et al. (2007) claim that various techniques can be used to overcome the above mentioned problems, for instance to randomly select items that are not rated or not rated often enough and to ask users to provide ratings on them, or using techniques different from collaborative filtering, such as using metadata or content analysis, to recommend items.

3.8.2 New User Problem

The new user problem occurs when the recommender system has already collected a certain amount of user profiles and recorded a certain amount of ratings, but the system has no information about the new user. Given that situation, the majority of recommender systems show poor performance. (Middleton, Shadbolt, & De Roure, 2004) If a new user has just entered the system and no ratings of that user have been recorded yet, no personalized predictions can be provided. To calculate a neighborhood of similar users, the system needs rating information about the active user. (Schafer et al., 2007) Schafer et al. (2007) exemplify various solutions to overcome this problem:

- Rating of initial items: Before a new user is able to use the system he or she is asked to rate some initial items.
- Provide non-personalized recommendations: Until the user has rated enough items, the system generates non-personalized recommendations based on population averages.
- Summarization of taste: The system asks the user to describe his or her preferences in aggregate, for example by giving an overall description of what kind of movies he or she likes.
- Demographic information: The user is asked for some demographic information.
- Similar demographics: The recommender system utilizes ratings of different users, that have a similar demographic profile, to generate recommendations.

3.9 Addressing Cold-start Problems

Various recommender systems are challenged by cold-start problems. Content-based recommendation techniques do not suffer from new item cold-start problems but still have to face the new user cold-start problem. As mentioned before (see Section 3.5.4), knowledge-based recommendation techniques are not affected by cold-start problems because the generation of recommendations is not based on user ratings, but on deep knowledge about a specific product domain.

The approach proposed in this work tries to support the recommendation process of an existing personalization system by applying collaborative filtering techniques even if the system is in a cold-start siuation. Recommender systems applying collaborative filtering techniques have to consider both new-user and new-item cold-start problems. Hereinafter, some current approaches addressing cold-start problems in escpecially in collaborative filtering recommender systems will be introduced to obtain more insights about this topic.

3.9.1 Effective Association Clusters Filtering to Cold-start Recommendations

Huang and Yin (2010) propose an efficiently association clusters filtering (ACF) algorithm that establishes cluster models based on the rating matrix to address the cold-start problem. The proposed methods aim to enlarge the prediction range by calculating predictions based on user groups (or user clusters) instead of relying on the relation between a very small amount of users. The approach may be classified as a metalevel hybrid that is based on the taxonomy given by random clusters. Recommendations can be generated collaboratively by applying the clusters filtering. An example (see Figure 3.4) provided in the work of Huang and Yin (2010) sketches the idea behind the introduced concept. To provide a prediction of the item represented by the rhombos-shaped icon to user x the correlations between user x and cluster C_i and cluster C_i and user 2 are calculated. The approach presented by Huang and Yin (2010) consists of several steps:

- The user set is randomly divided into N clusters, whereas N can be a experience value.

- The user-cluster correlations are calculated to determine which user belongs to what cluster.
- In a training step, the clusters training algorithm selects the minimum correlation user in each cluster, to find the new cluster with the maximum correlation between the chosen user and that cluster. The user is moved to the new cluster if it differs from the original cluster. The algorithm proceeds until there are no further changes in the loop.

Huang and Yin (2010) mention that predictions for a certain user can cover much more unknown ratings because the cluster associated to that user effects more users. To provide experimental results Huang and Yin (2010) used a real dataset with more than 1 million ratings from the MovieLens recommender system. 90% of the dataset were eliminated by random to make the dataset more sparse. Users with less than 5 ratings are considered to be cold-start users. The resulting test set contained 6038 user (containing 1279 cold-start users) with at least one rating and 3408 items and an overall of 100K ratings on these items. Huang and Yin (2010) state that their proposed ACF algorithms improve the coverage of existing collaborative filtering approaches and achieve the same or slightly better precision by considering ratings for the exact target item and also ratings for similar clusters. Additionally Huang and Yin (2010) mention that the coverage improvement through the proposed ACF algorithms is much higher for cold-start users (because they have just few ratings) than for all users compared to existing collaborative filtering approaches. (Huang & Yin, 2010)

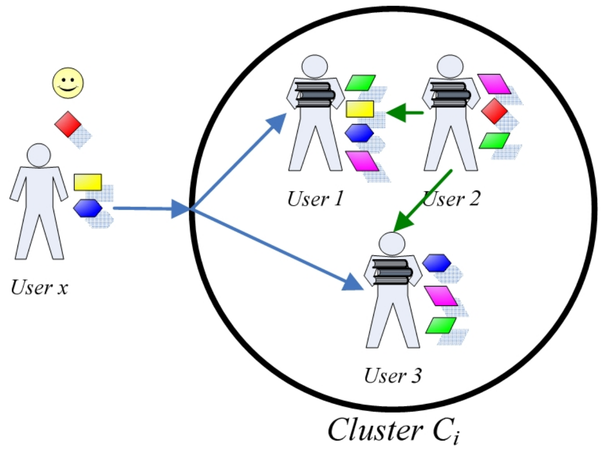


Figure 3.4: Motivation of the clusters filtering algorithm. (Huang & Yin, 2010)

3.9.2 Reducing the Cold-Start Problem in Content Recommendation through Opinion Classification

Poirier, Fessant, and Tellier (2010) propose an approach to reduce the cold-start problem by extracting user ratings from textual user comments via opinion mining to a recommender system. Poirier et al. (2010) divide the entire process chain in two main tasks: The analysis of textual data to obtain a user-item-rating matrix and providing recommendations by applying collaborative filtering using the resulting user-item-rating matrix. Poirier et al. (2010) tested the process chain on movie recommendation by using three different corpora to conduct their experiments:

1. Recommendation Corpus: This is a corpus of ratings by 400,000 users leading to about 100,000,000 user-item-rating triplets that were recorded on Netflix and were made available for the Netflix Prize (a challenge aiming to enhance the collaborative filtering methods for an online rental-service for DVDs). The corpus is used to measure the quality of recommendations in the final step of the experiments.
2. Text Classification Corpus: This is a corpus originating from the website Flixster recording, among other things, reviews and ratings about films. The corpus is resulting in user-item-rate-review quadruples and contains about 3,300,300 reviews of about 10,500 different movies composed by almost 100,000 users.
3. Learning Classification Corpus: This corpus which contains about 175,000 user-item-rate-review quadruples also originates from Flixster but has no intersection with the Test Classification Corpus.

In order to prepare the two textual corpora Poirier et al. (2010) applied some preprocessing such as deleting uncommon words and putting each letter in lowercase.

Task 1 - Opinion Classification (from Texts to Ratings)

In this first step Poirier et al. (2010) chose texts where the author is known and an opinion about an identified item is provided to obtain user-item-review triplets that can be used for the opinion classification task. The goal of this task was to finally receive user-item-rating triplets. To perform the opinion classification Poirier et al. (2010) used a Selective Naive Bayes (SNB) method by applying a tool called KHIOPS. This data mining tool enables automatic creation of successful classification models on very large-scale data. By using this tool the most important and informative variables from the textual data that could be used for opinion classification could be determined. The described classification task allows the classification of the user reviews into positive and negative reviews.

Task 2 - Collaborative filtering

Based on the user-item-rating matrix resulting from task one an item-based collaborative filtering method can be performed. Huang and Yin (2010) mention that the recommender system used can either work with content-based or collaborative filtering techniques, whereby Pearson correlation is used for collaborative filtering and Jaccard similarity is used for content-based filtering. Recommendations are predicted by using similarity tables that are built during the learning phase. To measure the error rate between predicted and real items Poirier et al. (2010) used Root Mean Squared Error (RMSE) measures.

Summary and Results

The experimental tests for this work were performed with ratings obtained through opinion classification and also with real ratings to examine the impact of the opinion classification task. According to Poirier et al. (2010) the results with ratings predicted from opinion classification with data provided from Netflix were as good as results obtained using real ratings. Poirier et al. (2010) mention that their experimental results show that recommendations generated using the external textual data from blogs and forums are reasonably good, but not as good as results obtained by learning tasks on the Netflix corpus. Poirier et al. (2010) assume that this difference originates from the different amount of information available in each corpora (an average Netflix user made about 200 ratings on DVDs whereas an average Flixster user only made about 30 ratings on films). Moreover Poirier et al. (2010) claim that an achieved F_{score} of 0.71 for the opinion classification task is sufficient to generate recommendations that are as good as ones provided by using real ratings.

3.9.3 Alleviating the Cold-start Problem of Recommender Systems using a New Hybrid Approach

To overcome the new user cold-start problem Basiri, Shakery, Moshiri, and Zi Hayat (2010) propose the "*optimistic exponential type of ordered weighted averaging (OWA) operator*" as a hybrid recommender system approach that aims to provide accurate information by using all existing information that is available for every user. The proposed approach uses five different classification strategies and the results returned by the underlying classifiers are combined applying the optimistic exponential OWA operator to finally generate recommendations.

Figure 3.5 shows a graphical overview of the approach and the cooperation of the five strategies that are explained hereinafter (Basiri et al., 2010):

1. Collaborative filtering method: The first strategy applies a collaborative filtering technique where the distance between two customers is calculated considering the number of common rated items.
2. Content-based filtering method: The second strategy applies content-based filtering and calculates predictions based on the similarity between the current item and items the active user rated in the past.
3. Collaborative filtering and demographic-based filtering: The third strategy works similar to the first strategy, nevertheless to compute the similarity between two users demographic characteristics of both users are additionally incorporated with the calculation.
4. Demographic-based filtering: The fourth strategy uses demographic information about users to find like-minded users.
5. Collaborative filtering and content-based filtering: The fifth strategy combines strategy 1 and strategy 3 as a hybrid approach where the result of this strategy is calculated by dividing the summation of the results from strategy 1 and strategy 3 by 2.

In their work Basiri et al. (2010) refer recommendation as a binary classification problem and distinguish between preferred and non-preferred items for a particular user, where accepted items are labeled with 1 and non-accepted items are labeled with 0. Basiri et al. (2010) used k-nearest neighbor algorithm to identify instance labels and Euclidean distance similarity measure to detect the similarity between items or users in all five strategies. The results of all strategies are used as input for the OWA algorithm and the fused result of the OWA algorithm is used as final result for each instance. An item is only recommended to a user if the resulting predicted label is 1.

To evaluate their approach Basiri et al. (2010) used the MovieLens dataset and claim that their experimental results show the superiority of the proposed approach in (new user) cold-start conditions compared to other well-known approaches.

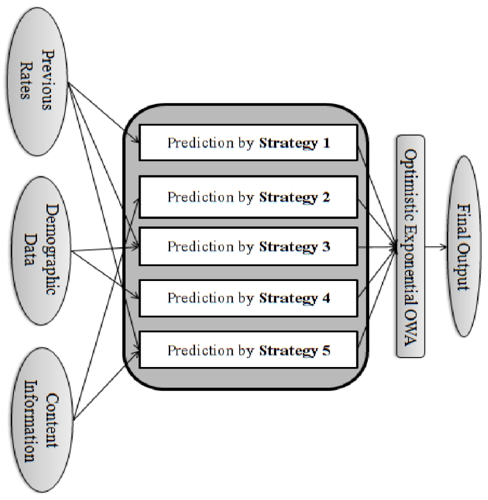


Figure 3.5: The proposed hybrid recommender system approach. (Basiri et al., 2010)

3.10 Conclusion

This Chapter discussed the various types of recommendation techniques as well as their benefits and drawbacks. Collaborative filtering has shown to be the most adequate method to fulfill the tasks that are needed to achieve the goals of this work that are briefed in Chapter 1. As mentioned in Section 3.7.4 collaborative filtering does not need knowledge and information about the existing items and as well implicit feedback given by unary ratings can be sufficient to make valuable recommendations. These insights allow the prototypical recommendation engine to apply collaborative filtering because the ratings about the items (or features) that are extracted from the positively rated customer reviews can be considered to be unary ratings.

The major tasks (see Chapter 3.7.1) of collaborative filtering recommender systems include the recommendation of items the user has not seen yet and he or she might be interested in and to make rating predictions about certain items. The approach proposed in this work aims to provide a list of recommended items to support the target system Xohana instead of making predictions for certain items. User-based Top-N approaches introduced in Section 3.7.2 which generally works by considering ratings made by the nearest neighbors of the active user also provides knowledge that is very valuable for the prototypical implementation of the recommendation engine. The informations about the benefits of clustering approaches provided in Section 3.7.3 are also very useful because the data source used by the prototype that is built upon customer reviews can easily be divided into user clusters. This clustering has the advantage of better scalability. Additionally the clusters filtering approach proposed by Huang and Yin (2010) (see Section 3.9.1) has shown to be useful to enlarge the prediction range by calculating predictions based on groups of users rather than relying on relations between a small amount of users which is also useful to overcome cold-start problems.

The approach proposed by Poirier et al. (2010) (see Section 3.9.2) uses opinion classification to infer ratings about certain items by processing user comments. Although this approach differs from the approach proposed in this work which does not try to extract opinions about certain items or products but aims to extract and use preferred preferences of users within a certain domain, it has shown that data extracted from user reviews or user comments can be useful to address cold-start problems.

The next chapter introduces a conceptual design that covers the requirements and goals to this work. Thereby it will be described in more detail how the proposed ideas and insights obtained from this and the previous chapter are applied.

4. The Supportive Recommender System Prototype - Conceptual Design

In the last two chapters in depth research about the topics that are related to this work has been conducted to identify established technologies and methods that are adequate to develop a concept that is able to fulfill the goals of this thesis. As stated in Chapter 1 this work aims to provide an effective approach that allows to support the recommendation task of an existing personalization system, escpecially in cold-start situations. To the achieve this objective, the knowledge base of the existing system shell be extended by extracting relevant data and information from customer reviews. With that, personalized collaborative recommendations shell be provided as a supplement to those generated by the target system. The insights gained from previous research about information extraction technologies and recommender systems are used to design this approach.

For easier comprehension of the basic ideas and major goals of the proposed approach, the existing target system Xohana and its general functionality is shortly briefed at first. After that, the major goals of the proposed approach and the strategies that are used to implement the belonging sub-tasks as well as the correlations and connections between the recommender system prototype and the target system are explained.

4.1 The Target System - Xohana

Xohana[1] is an innovative market place that helps people to better articulate what they really want. The way in which the system works to support the articulation

[1]http://www.xohana.com Xohana e.U. Website, last access 03/2011

of customer requirements is a very crucial feature of Xohana: structured, but (compared to other current approaches) without limiting the liberty of expression. In its current implementation, Xohana helps people to articulate and describe the kind of vacation - especially in the domain of health tourism - they desire. This can be done by defining several requirements that consist of a criterion with a certain demand. Xohana facilitates this process of requirement articulation by making intelligent, personalized suggestions of that criteria and demands on a much finer grained level instead of suggesting products. That is, the items suggested by Xohana are word sequences or terms that can be used to describe the customers wishes and requirements.

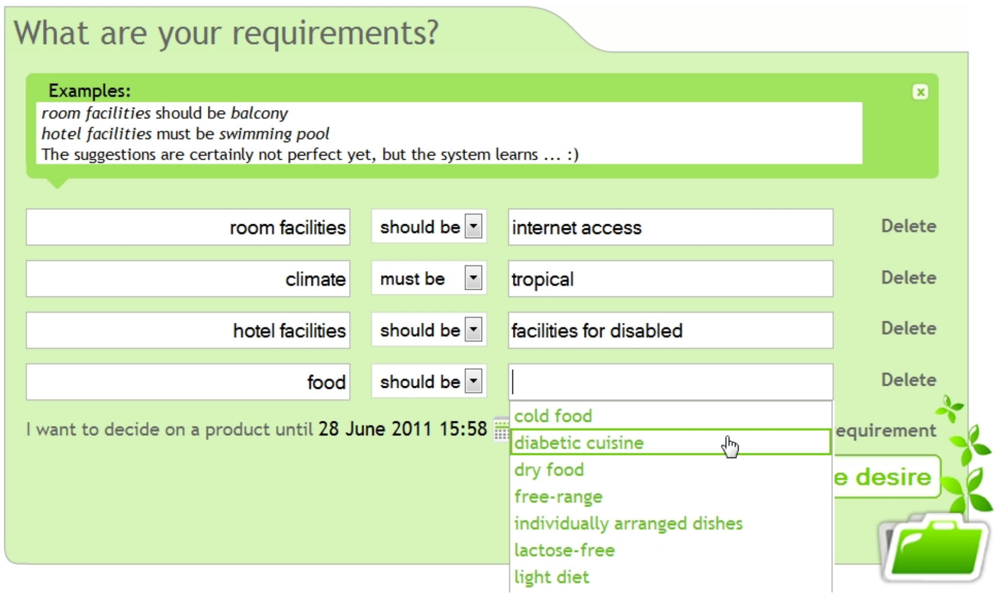
The suggestions are personalized and always fit to the current inquiry which is succeeded by using both pre-populated data (for instance sector knowledge modeled in the system and geographical relations) and behavior patterns of previous customers (self-learning system). The suggestions can either be adapted or ignored to articulate the own new needs of the customer (which helps vendors to recognize what requirements have not been considered yet).

Figure 4.1 shows the user-interface of Xohana. Customers can define several requirements consisting of a criterion with a certain demand, such as "*climate* must be *tropical*" or "*food* should be *diabetic cuisine*". After all requirements are entered via the user-interface the customer can save his or her desire to finally achieve offers that best fit to their needs. (Rollett, 2008; Semantic Web Company, 2009)

4.2 System Goals and Requirements

The prototype implementation aims to determine and demonstrate the potential for supporting and optimizing the recommendation process of the existing personalization system Xohana rather than creating a standalone recommender system. The three major goals of the approach proposed in this work are the following:

- **Goal 1 - Providing additional product features**: Xohana contains a data source of high quality that was created with expert knowledge of the (health) tourism domain to produce suggestions for terms or word sequences used to formulate the customer requirements. This goal consists of providing additional


What are your requirements?

Examples:
room facilities should be *balcony*
hotel facilities must be *swimming pool*
The suggestions are certainly not perfect yet, but the system learns ... :)

room facilities	should be	internet access	Delete
climate	must be	tropical	Delete
hotel facilities	should be	facilities for disabled	Delete
food	should be		Delete

I want to decide on a product until 28 June 2011 15:58

Figure 4.1: The Xohana User-Interface - Screenshot taken from Xohana Website (Xohana.com, 2011)

product features - in that case word sequences or terms used to articulate the customer requirements about the desired vacation - as a supplementation to those that are already provided in the target system Xohana. This supplementary data should be obtainable without much domain knowledge from accessible and already existing data.

- **Goal 2 - Providing personalized recommendations**: To make intelligent and useful recommendations the prototypical recommender system should be able to provide personalized recommendations to the current system user based on rating behaviour of other users. This should be done by exploiting the structure and nature of the additionally used data sources.
- **Goal 3 - Providing another alternative solution to address cold-start problems**: The generation of additional supporting personalized recommendations via the recommender system prototype using user-based collaborative filtering should be possible for the target system in cold-start situations where no rating history is available and with that, every item is unrated and every customer is considered to be a cold-start user. The idea behind this is, to provide a possibility to support the target system with another alternative to address cold-start problems in situations where user histories (or rating histories) are missing by using alternative rating histories.

The next section describes the conceptual design that was elaborated to reach these three goals.

4.3 Concept Description

To fulfill the goals of the proposed approach an adequate data source that can be used to supply the recommender system prototype has to be prepared. As already mentioned in Chapter 2 there is a huge amount of natural language texts, such as user comments and user reviews, that are produced and available via the Internet, containing much useful information that can be used for different purposes. Considering the insights that emerge from this part of the work, 3s about products that are written in natural language have shown to provide valuable information about product characteristics, user opinions and preferences of customers.

A small but sufficient amount of customer reviews from the famous web portal TripAdvisor.com[2] is used to provide the needed data source. The collected information about the customer reviews consists of data such as the name and geographical region of the review objects (that is accommodations such as hotels, apartments, holiday clubs or similar), the vacation category (for instance Adventure, Beaches & Sun, Family Fun, etc.), the customer review text and the description about the review object either collected via the belonging hotel website or an online holiday booking website. The descriptions and reviews are all written in English natural language. To make the data that is residing within these texts useable for computational tasks some pre-processing has to be done beforehand. This preprocessing includes the extraction of the product features describing the review objects. To enable this, part-of-speech tagging is used to assign each word in the texts the belonging part-of-speech. With that the feature extraction methods of the prototype are able to identify word groups of interest that can either consist of one or more nouns or on or more adjectives followed by one or more nouns. The extracted word groups can then be used to supply the prototypical recommender system.

In general, the prototype covers two major tasks: Automatic product feature extraction from customer reviews which can be seen as the major part of the data processing step and provision of personalized recommendations using the extracted

[2]http://www.tripadvisor.com TripAdvisor LLC Homepage, last access 03/2011

features. The next sections describe how these tasks can be fulfilled and how the recommender system prototype and the target system Xohana are collaborating.

4.3.1 Data Processing - Product Feature Extraction from Customer Reviews

Catalog descriptions provide general information about a certain accommodation and its facilities and services. Customer reviews give useful information about certain features (positive or negative) of the reviewed object that are often not covered by the catalog description. In addition customer reviews allow to recognize what the customer liked or disliked in particular an with that the customer's subjective preferences. Certain terms and phrases customers use to describe their experience can be seen as items he or she liked in particular about the accommodation. These phrases or groups of words can be used to build the bridge between the word sequences or terms that are suggested by Xohana to help the user to articulate his or her requirements and needs for his desired kind of holiday.

Xohana enables users to formulate personal requirements using criteria and the belonging demands, for instance the criterion "food" that is demanded to be "vegetarian" resulting in the defined requirement "*food* should be *vegetarian*" (see Figure 4.1). In order to correspond to this schema this feature extraction task aims to extract features as word groups that consist of adjective-noun pairs. The adjective-noun pairs can again be seen as pairs of criterion and demand. As word groups consisting of one or several nouns are also likely to be product features they are extracted too. Although noun sequences that do not contain adjectives can not be directly modeled as pairs of criterion and demand they can be very useful for the query completion task (see Section 4.3.2) and for recommending single criteria or demands instead of recommending compelte requirements.

Figure 4.2 shows an extract of a customer review composed at TripAdvisor.com. The example shows the features consisting of noun sequences or adjective-noun pairs that have to be extracted from the review texts. Depending on writing style and grammar the texts can be differently structured which complicates the feature extraction process. One can see that the extraction of feature F_1 ("*clean hotel*") and feature F_6 ("*private balcony*") has to be extracted with different strategies. Features like feature F_6 that occur as consecutive sequences of adjectives and nouns can be identified and extracted easier than features like feature F_1 where the words that build the feature are present in a different order and additionally divided by an unpredictable number of other words. Chapter 5 describes the developed product feature extraction algorithm that is used to enable the extraction of features that are structured in either ways.

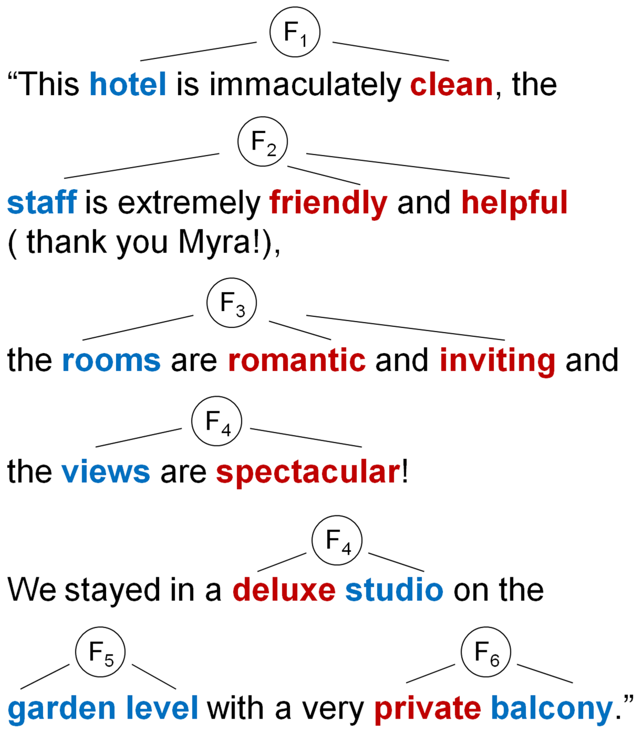

F1
"This hotel is immaculately clean, the
F2
staff is extremely friendly and helpful
(thank you Myra!),
F3
the rooms are romantic and inviting and
F4
the views are spectacular!
F4
We stayed in a deluxe studio on the
F5
F6
garden level with a very private balcony."

Figure 4.2: Product Feature Extraction Example - Customer Review Extract, taken from (TripAdvisor.com, 2011a)

In avoidance of the very complex tasks that have to be performed for opinion mining or sentiment analysis, only positively rated reviews with an overall rating scare of 5 out of 5, resulting in an user recommendation of almost 100%„ are used to build the data source needed by the recommender system prototype. This strategy also fits very well to the requirement articulation process of Xohana as customers do not define things they dislike, but only things they prefer and need. The entire data processing step that has to be performed to automatically generate the needed data source from the customer reviews for the recommender system prototype includes, in general, the following tasks: Data pre-processing, product feature extraction, statistical analysis and some data post-processing (see Figure 4.3).

In the data pre-processing step the collected review and accommodation description texts have to be prepared for the next step, the product feature extraction task. Therefore at first, the texts have to be cleaned from possible HTML tags. As the feature extraction process relies on knowing the word classes of each sentence in the review or accommodation description text, the words in the natural language texts have to be tagged with their parts-of-speech. With knowledge about the parts-of-speech the feature extraction can be done using adequate algorithms to identify the word groups of interest that represent the product features.

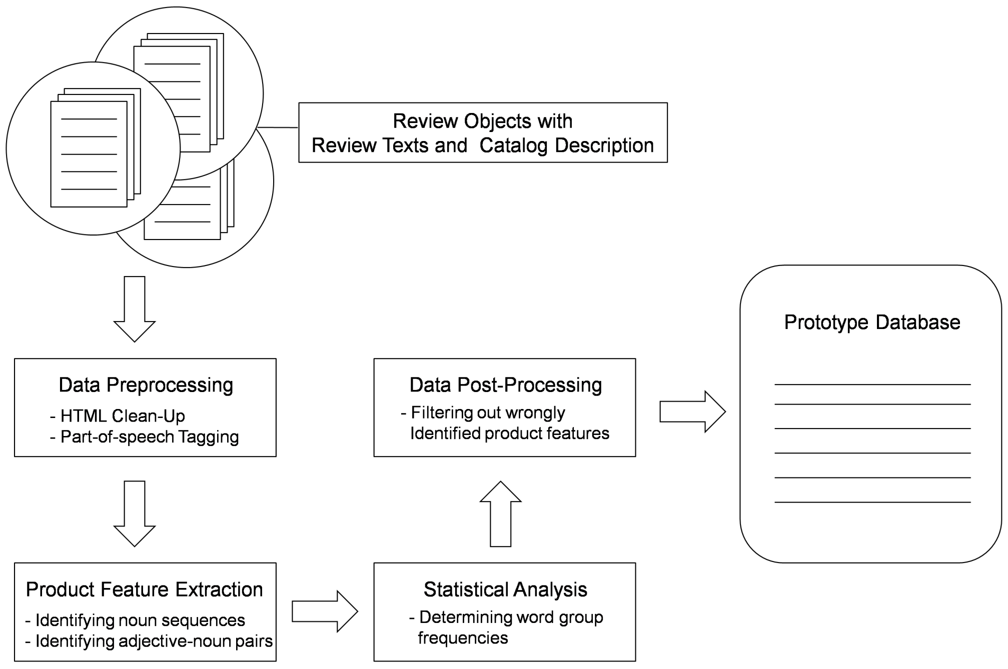
After the features have been identified and extracted some statistical analysis has to be done to obtain useful information about frequencies of the extracted features. To know how frequently the features occur within the entire data set and also within the review and description texts of a certain review object is very useful considering the recommendation strategies of the prototype.

The automatic feature extraction process does not need specific domain knowledge and does not use any ontology or manually built data set. Some of the extracted word groups are identified correctly from the technical point of view, but they are no product features. For instance word groups like "next day", "first night" or "previous reviews" are correctly identified as adjective-noun pairs, but are definitely no product features of the reviewed accommodation. To get rid of most of these false positives some semi-automatic post-processing has to be done. This is not an easy task, but fortunately these kind of word groups occur very frequent within the entire data set which makes it easier to identify and eliminate them.

The result of this data processing step is a database of groups of words that represent the product features extracted from the customer reviews and accommodation descriptions. The database is built in a way that makes it useable for the recommendation process of the prototype. The next section describes the concept of the recommender system prototype and elucidates some strategic decisions.

4.3.2 Providing Personalized Recommendations

The supportive recommender system prototype utilizes the database mentioned in the previous section. The database is designed in a way that makes it possible to infer ratings about the existing items (here word groups that represent the extracted product features). Each customer that provided a review is considered to be a sin-



Review Objects with
Review Texts and Catalog Description
Data Preprocessing
- HTML Clean-Up
- Part-of-speech Tagging
Product Feature Extraction
- Identifying noun sequences
- Identifying adjective-noun pairs
Statistical Analysis
- Determining word group frequencies
Data Post-Processing
- Filtering out wrongly Identified product features
Prototype Database

Figure 4.3: Data Processing including Product Feature Extraction

gle user an each extracted product feature is considered to be an item. Each item that originates from a certain customer review is considered to be unary rated by the user who wrote the review. Additionally all items that were extracted from the review object description (catalog description of the accommodation) are also considered to be unary rated by the user who wrote the review about that certain review object. The rating information enables the recommender system prototype to apply user-based collaborative filtering to generate supplementary personalized recommendations for the target system. To follow this strategy two basic assumptions are met:

1. All items that were extracted from a certain review were preferred by the user because only positively rated review objects with almost 100% recommendation are used. In positive reviews written by a very satisfied good features of the review object and subjective preferences of the customer are usually emphasized.
2. All items that were extracted from the catalog description have been known beforehand (before the holiday was booked) by the customer. With that one can assume that the items (or product features) mentioned in the description text have been recognized by the customer. As the user was very satisfied considering only notably positively rated review objects are chosen one can

even assume that he or she liked the items (or product features) mentioned in the catalog description. This corresponds to a unary rating which indicates that the user has seen or liked the items mentioned in the catalog description.

The set of unary rated items that correspond to the extracted product features by each user can be regarded as the user's preferences. In order to exploit the given structure of the review data source from TripAdvisor.com, the reviewed objects with the articulated preferences of all reviewers are viewed as naturally given clusters of user preferences (and with that as user clusters). This follows the basic assumption that all users that were satisfied by the same review object have similar taste about vacations (or a certain type of vacation) that is based on the product features known beforehand from the catalog descriptions. The user's taste within these clusters varies by the subjective preferences articulated in the customer reviews. This situation facilitates the recommendation process of the prototype in identifying users that have similar preferences as the active system user of Xohana. As these users can be easier identified by only calculating the similarity to the given user clusters instead of each user in the prototype database, this method additionally provides better scalability. Each cluster represents a collaborative preference space. Based on that clusters personalized recommendations can be made to the active user.

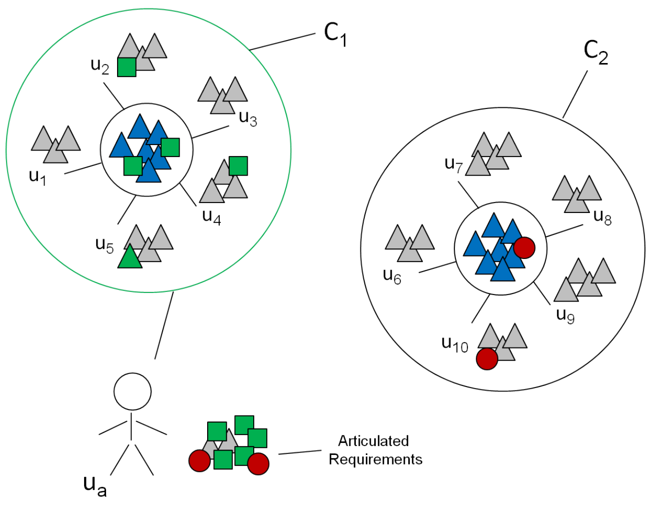


Figure 4.4: Clusters of Collaborative User Preferences

Figure 4.4 sketches the basic idea behind this approach. The triangles right to the active system user u_a symbol represent the requirements he or she has already articulated via the Xohana user interface. The symbols C_1 and C_2 represent user clusters that are built upon the reviewers of the same review object where the blue triangles represent the items (product features) that were extracted from the catalog description of the belonging review object. The gray triangles next to the users (u_1 - u_{10}) represent the items that were extracted from the customer reviews of the belonging users. The entire collaborative preference space of each cluster consists of the item ratings extracted by all reviewers and the catalog description of the review object. The squares and circles symbolize the co-rated items of the active user u_a and the other users in the prototype database. As can be seen from Figure 4.4, the active user u_a has a higher correlation (considering the co-rated items represented by the squares) to cluster C_1. Thus, regarding the articulated requirements of the active user u_a the user's preferences are more similar those of cluster C_1.

4.3.3 Collaboration between Xohana and Prototype

To enable a collaboration between the prototype and Xohana the extracted items (product features) are regarded as terms that are used in Xohana. Each reviewer is regarded as a user that already provided unary ratings about the items (or terms) that were extracted from the belonging review. By using the extracted items (or terms) the recommender system prototype can produce supplementary personalized recommendations of terms for Xohana users and provide additional terms to that already covered by the knowledge source of Xohana.

This Section describes how the prototype collaborates with the target system Xohana to provide supplementary personalized recommendations based on the user preference clusters described in the previous section. Figure 4.5 outlines the connection between Xohana and the prototypical recommender system. Whenever the user articulates a criterion or a demand via the user interface of Xohana the back-end system of Xohana generates the appropriate suggestions. Additionally Xohana requests supplementary recommendations from the recommender system prototype. The prototype has access to all words or strings the user entered via the Xohana user interface. To identify the preference cluster containing the users that are most like-minded as the active user, the prototype at first tries to match the requirements already articulated by the user with the items in the database of the prototype. The criteria with the belonging demands the active user already articulated are considered to be the unary rated items of the active user. To choose one of the preference clusters the cluster containing the most co-rated items to the active user is determined. The personalized recommendations are than made based on the chosen cluster with kind of a user-based collaborative filtering method.

To match the requirements articulated by the active user with the items in prototype database a query suggestion method is used. Whenever a new requirement is entered, the prototype can help to autocomplete partially entered requirements by making some suggestions. For example, if the user has entered a criterion the prototype can suggests corresponding demands via query completion. For example if the user enters the criterion "*food*" the prototype can suggest items like "*vegetarian*" or "*Italian*" for the belonging demand, based on the complete items (product features) "*vegetarian food*" and "*Italian food*" that are stored in the prototype database. Likewise query completion can be done for criteria and demands only, for example if the user has just entered some letters. For instance, if the user has already entered the criterion "*staff*" and he or she wants to define the requirement "*staff* should be *english speaking*", the prototype can recommend the term "*english speaking*" if the user has just entered the first three letters "*eng*". The query completion method can also be combined with the collaborative filtering approach based on the chosen user preference cluster where the query completion is limited on the preferences of the like-minded users within that cluster instead of using the entire item dataset of the prototype database.

Figure 4.5 provides a simplified view of the collaboration between Xohana and the prototype. The user communicates his requirements via the user interface of Xohana. The data entered by the user is than sent to the Xohana system and forwarded to the prototype. The prototype uses the transferred input data, such es entered criteria, demands and incomplete words typed by the user, to apply query completion and to generate supplementary personalized recommendations utilizing the prototype database. Additionally the already articulated requirements are sent to the prototype to determine the user preference cluster that best fits to the preferences of the active user every time a new requirement has been stored. The suggestions and recommendations generated by both Xohana and the recommender system prototype are then aggregated and submitted to the user via the input interface.

All articulated requirements that can be matched to the items in the prototype database are already rated by one or more users in the the user dataset of the prototype database. By using the built rating dataset consisting of unary item ratings a rating history can be provided for cold-start situations where no rating history is recorded. This allows to find users that are similar to the active user and to apply collaborative filtering to support the recommendation process of Xohana even if there are no collected user histories. With that Goal 3, providing another solution to address cold-start problems, can be fulfilled.

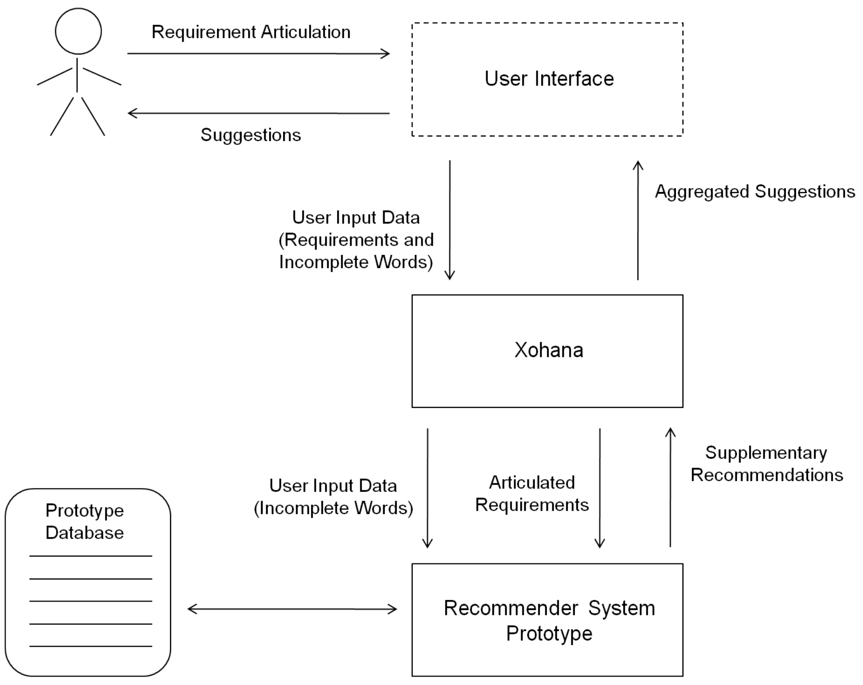

Requirement Articulation
User Interface
Suggestions
Aggregated Suggestions
User Input Data (Requirements and Incomplete Words)
Xohana
Supplementary Recommendations
Articulated Requirements
User Input Data (Incomplete Words)
Prototype Database
Recommender System Prototype

Figure 4.5: Collaboration between Xohana and Prototype

4.4 Conclusion

This Chapter provided an overview of the basic ideas behind the proposed approach. The insights obtained from the previous chapters have been used to design a concept that uses various established techniques, such as natural language processing and collaborative filtering, to achieve the goals defined in this Chapter. Considering the system requirements to achieve the goals and to perform the needed tasks especially the product feature extraction process has show to be quite challenging due to the different writing style and used grammar of the used customer reviews. Nevertheless the utilization of the customer reviews provides a good possibility to develop a cluster based collaborative filtering approach with naturally given clusters (that do not need to be computationally determined) that facilitates the preference elicitation of the active user that shell be supported during the requirement articulation process. The query completion method offers a good solution to the goal of providing additional unknown items. With the database built that contains the unary rated items a rating history can be provided which provides another solution to support the target system in cold-start situations.

The recommender system prototype was implemented based on the concept proposed in this Chapter. The next chapter elucidates how the developed ideas and strategies have been programmatically implemented to create the resulting application.

5. Detailed Design and Prototype Implementation

This Chapter describes how the elaborated concept proposed in Chapter 4 was implemented. At first the applied methods and techniques and the used software tools and programming languages as well as why they have been chosen will be explained. Afterwards the detailed design of the implemented software components, their tasks and collaboration will be described. In a next step the structure of the database that was created from the used customer reviews will be shown to better understand how the developed methods and algorithms work. Finally the last section explains more precisely how the data processing tasks, especially the product feature extraction task, and the recommendation tasks have been implemented in the prototype components.

5.1 Methods and Materials

This Section outlines the techniques and methods, tools, programming languages and the implementation framework that have been used for the implementation of the prototype and why they have been chosen.

5.1.1 Methods and Techniques

The following approaches and technologies adapted from previous research have been used to fulfill the tasks of the prototype:

- Information Extraction with Part-of-speech Tagging
- String Matching with Wildcards

- Query Suggestion
- Collaborative Filtering

Part-of-speech tagging was chosen because it gives very much valuable information about the classes of words in natural language texts. Knowing the word classes or parts-of-speech of the words in the product reviews is a very good basis for automatic extraction of certain information of interest. Moreover, various state-of-the-art approaches that were introduced in this work (see Section 2.3) successfully used part-of-speech tagging as a preprocessing step for product feature extraction.

String matching with wildcards allows a looser matching of strings that are syntactically not totally equal. For instance, using a wildcard pattern like "*% bathroom*" would match the string "*outdoor bathroom*" as well as the string "*indoor bathroom*" (the "%" symbol represents the wildcard character). This technology is very crucial to fulfill the recommendation process of the protype. As the users of Xohana can define requirements by using terms without limitation of expression these terms have to be, if possible, matched to the items (or groups of words) in the prototype database at first to determine like-minded users, to perform the query suggestion task and to finally being able to generate collaborative recommendations.

As mentioned in Chapter 1 query suggestion technologies generally suggest to users search terms previously made by other users. This process can be adapted by the recommendation process of the prototype. When Xohana users articulate their requirements via the user interface they define criteria and demands. If the user has already typed, for instance, the criterion (or has chosen one of the suggested criteria) this criterion can be regarded as an incomplete search term. The recommender system prototype can try to find full queries that are syntactically similar to the original query. The full queries correspond the items (or word groups) in the prototype database that have already been defined by other users. For example if the user defines the criterion "*cuisine*" the recommender system prototype can try to complete this incomplete requirement articulation with complete terms (items) in the prototype database. For instance, the prototype could suggest terms such as "Italian cuisine" or "vegetarian cuisine". To personalize this process the query suggestion can be combined with collaborative filtering, where only complete terms of like-minded users are suggested.

Collaborative filtering is used to provide personalized recommendations based on the preferences of other like-minded users. The goal of the prototype is to suggest to the user terms that might be of interest in addition to that suggested by Xohana. As there is no content information available about the used items in the prototype database that was built from the product reviews, because the items are simple strings, content-based filtering would not have been possible to fulfill the recommendation task using the prototype database. Also there is no demographic information about the active user (not least because each user is considered as a cold-start user) to provide demographic based recommendations. But the nature of the used product reviews provides knowledge about the preferences of the customers that reviewed the various accommodation objects. With that it is possible to identify users of similar interest and to apply collaborative filtering for providing personalized collaborative recommendations.

5.1.2 Software Tools and Programming Languages

The following programming languages, software tools and frameworks have been used for the implementation of the prototype:

- Ruby On Rails[1] (version 2.3.5)
- Ruby[2] (version 1.8.7)
- PHP[3] (verion 5.1.3)
- MySQL[4] (version 5.1.41)
- Rubygem EngTagger[5] (verion 0.1.1)
- Rubygem ActiveRecord[6] (version 2.3.5)
- Rubygem Sanitize[7] (version 1.2.1)
- REXML[8] (version 3.1.7.2) XML toolkit for Ruby

[1]http://rubyonrails.org Ruby On Rails project homepage, last access 04/2011
[2]http://www.ruby-lang.org/en Ruby project homepage, last access 04/2011
[3]http://www.php.net PHP project homepage, last access 04/2011
[4]http://www.mysql.com MySQL project homepage, last access 04/2011
[5]http://engtagger.rubyforge.org EngTagger Library project homepage, last access 03/2011
[6]http://ar.rubyonrails.org ActiveRecord rubygem, last access 04/2011
[7]http://rubygems.org/gems/sanitize Sanitze rubygem, last access 04/2011
[8]http://www.ruby-doc.org/stdlib/libdoc/rexml/rdoc/index.html REXML XML toolkit for Ruby, last access 04/2011

The software components of the prototype have been implemented in the *Ruby On Rails* (for short *Rails*) open-source web-framework using the programming language *Ruby*. All prototype components that include the datapreprocessing, product feature extraction, statistical analysis and recommendation tasks have been implemented as part of Rails components. Only the postprocessing tasks for automatically and semi-automatically filtering out as much wrongly extracted product features as possible, have been implemented with some simple scripts using the programming language *PHP* and *MySQL*. The database built from the product reviews that is used to fulfill the recommendation tasks is implemented as a MySQL database.

MySQL was chosen, among other things, for two reasons: on the one hand Ruby On Rails provides a very good integration of MySQL databases. And on the other hand MySQL allows to use wildcards in SQL-Queries. This is very important to fulfill the recommendation tasks of the prototype to allow a looser string matching between the items (the extracted product features) of the prototype database and the terms articulated by Xohana users (see Section 5.4).

Because some of the collected product reviews contain HTML-tags that trouble the part-of-speech tagger they had to be removed before the tagging process was performed. This was done with the rubygem *Sanitize* which can be used to clean written text from HTML-tags. For the part-of-speech tagging which is crucial for the product feature extraction task the rubygem *EngTagger* was used. EngTagger, that was already shortly briefed in this work (see Section 2.2.3), takes as input any string and tries to assign each word in a sentence the proper part-of-speech. The results when using this tagger for tagging the product review texts were not always perfect but one has to keep in mind that part-of-speech tagging is not a trivial task. Additionally the different writing styles and accents used by the reviewers hamper the tagging process. Nevertheless, the results of EngTagger showed to be more than sufficiently for the product feature extraction task during the implementation phase of the prototype. Also EngTagger performed better in tagging the product review texts than other part-of-speech taggers that were available as rubygems, such as the rule-based part-of-speech tagger *Ruletagger*.

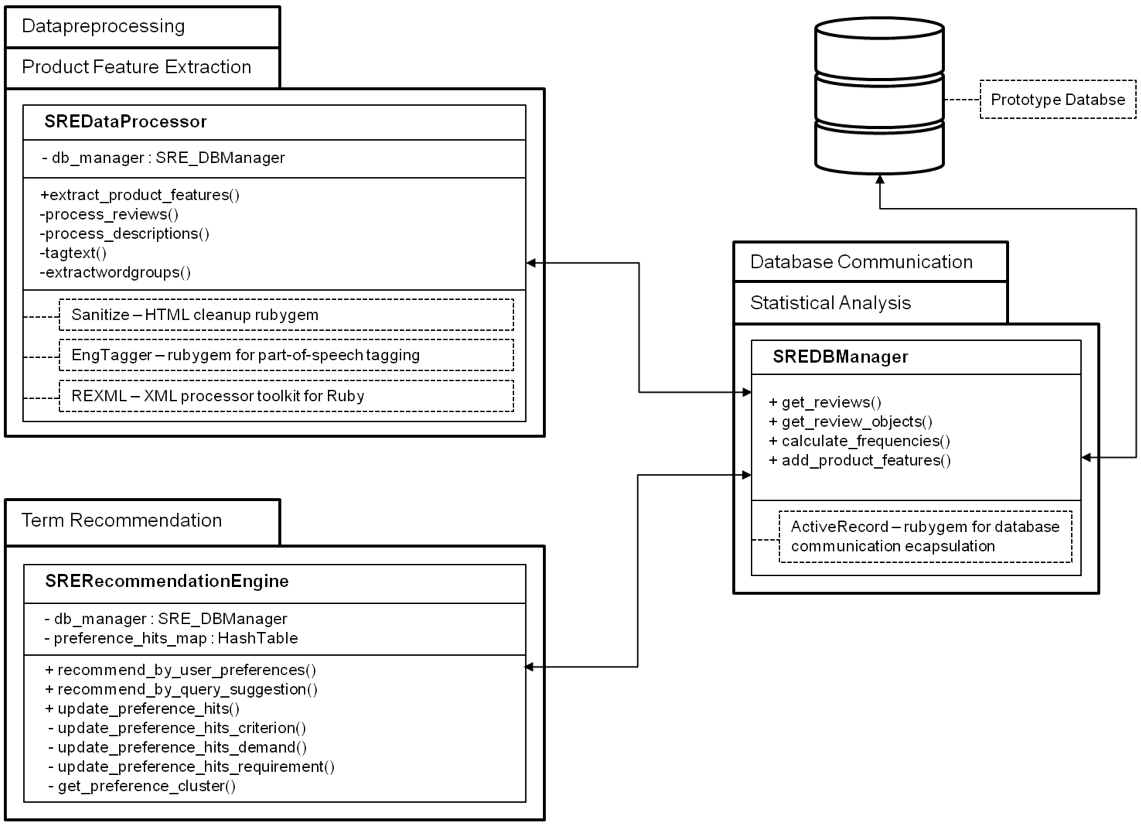
EngTagger returns as result an XML string whereby the XML-tags represent the parts-of-speech with the different tag names and the value of the XML-tags represent the tagged words (or tokens). To make the output of EngTagger useable for further processing *REXML* (an XML processor toolkit for the Ruby programming language) was used to transform the XML string with the tagged words into an array datastructure that is more suitable for the product feature extraction algorithms.

5.2 Prototype Components - Detailed Design

As mentioned in the previous section all components of the prototype have been implemented as Rails modules with the Ruby programming language. Figure 5.1 presents a graphical view of the components and how they are associated. The most important methods are listed in the class diagrams. All components are named with the prefix "*SRE*" which stands for "*Supportive Recommendation Engine*". The *SRE-DataProcessor* class is responsible for all data preprocessing and product feature extraction tasks. The SREDBManager class is used by both the SREDataProcessor and *SRERecommendationEngine* class to communicate with the prototype database. Additionally the SREDBManager is responsible for the statistical analysis that is to record how frequent the extracted terms occured within the entire dataset and the frequencies of the terms that occured only in the reviews (and as well in the corresponding catalog description) of a certain review object. The SRERecommendationEngine covers the recommendation tasks including, query suggestion, the generation of collaborative recommendations and the combination of both techniques. The components and their tasks are described in more detail in the following sections. In order to better understand how the implemented algorithms and methods of the components work, the structure of the prototype database is described first.

5.2.1 The Prototype Database

Figure 5.2 provides a graphical representation of the prototype database. The associations between the database tables are represented in a simplified way to show which tables are connected with each other. Several data about the the collected product reviews from Tripadvisor.com have been stored in the prototype database. The information about the different holiday types, such as "*Adventure*" or "Family and Fun", the global regions, such as "*Africa & the Middle East* or "*Europe*", hotel names, etc., have only been collected for completeness and for usage in



Datapreprocessing
Product Feature Extraction
SREDataProcessor
- db_manager : SRE_DBManager
+extract_product_features()
-process_reviews()
-process_descriptions()
-tagtext()
-extractwordgroups()
Sanitize – HTML cleanup rubygem
EngTagger – rubygem for part-of-speech tagging
REXML – XML processor toolkit for Ruby
Prototype Databse
Database Communication
Statistical Analysis
SREDBManager
+ get_reviews()
+ get_review_objects()
+ calculate_frequencies()
+ add_product_features()
ActiveRecord – rubygem for database communication ecapsulation
Term Recommendation
SRERecommendationEngine
- db_manager : SRE_DBManager
- preference_hits_map : HashTable
+ recommend_by_user_preferences()
+ recommend_by_query_suggestion()
+ update_preference_hits()
- update_preference_hits_criterion()
- update_preference_hits_demand()
- update_preference_hits_requirement()
- get_preference_cluster()

Figure 5.1: Prototype Components

possible future work. These data are not used in the current implementation of the prototype. The currently used tables are the following:

- review_objects: This database table contains the collected review objects, accommodations such as hotels, apartments, clubs or similar. Each review object entry has a unique ID and contains the catalog description that was collected from the website of the reviewed accommodation. All catalog descriptions are written in natural English language.
- reviews: The records in this table contain the product reviews and each record is associated to the corresponding review object. Each record has a unique ID which represents the unique ID of a reviewer which is finally considered as a unique user during the recommendation tasks of the prototype.
- review_token_groups: This table is used to store the product features (or items) that are extracted from the customer reviews. The field *token_string* represents the extracted product features which here are terms that consist of on ore more tokens. Being more precise each extracted term can consist of one or more nouns or one or more adjectives followed by on or more nouns.

Every record in this table is associated to the belonging review object and as well to the belonging review. The field *origin* is used to determine whether the term was extracted from a product review or from the catalog description of a certain review object.

- review_tokens: As the extracted terms that are recorded in the table *review_token_groups* can consist of several tokens, it is necessary to know if these tokens are nouns or adjectives (this is crucial for the recommendation task which will be explained in Section 5.4). Therefore each record in this table stores a single token that belongs to a complete term from the *review_token_groups* table whereas the part-of-speech tag of each single token is recorded. Additionally the position of the token within the belonging *review_token_group* record is stored.
- token_groups: This table records the unique terms (token groups) that exist in the entire prototype database. Terms that are recorded in the *review_token_groups* table can occur several times. This table is used to record how often these terms have been articulated by all reviewers in total which is represented by the field *frequency*.
- review_object_token_group_frequencies: This table is used to record how often a unique term has been articulated in total by reviewers of a certain review object.

In order to improve the database performance all tables used *Indexing*, which is a feature of MySQL that can help to speed up database queries.

5.2.2 SREDataProcessor

This component is responsible for data preprocessing and product feature extraction. The public method *extract_product_features()* can be applied to start the entire product feature extraction process, including the data preprocessing task. To communicate with the prototype database the SREDataProcessor uses the SREDBManager class. The method *extract_product_features()* includes the following steps:

- Requesting catalog description texts and product reviews using SREDBManager: The review texts and catalog description for each review object are requested using the methods *get_review_objects()* and *get_reviews()*.

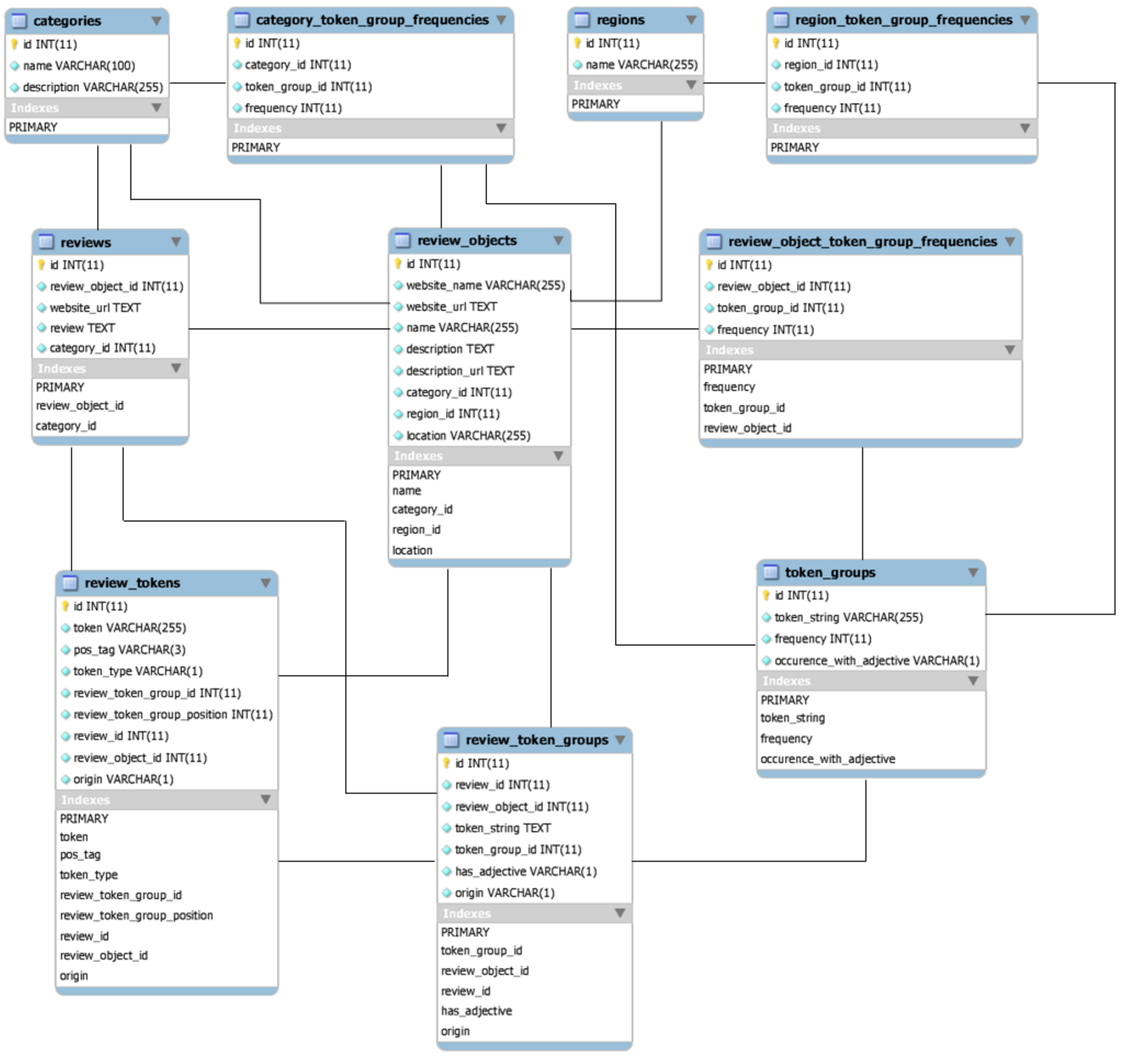


Figure 5.2: Prototype Database

- Preparation of the catalog descriptions and review texts: Each text is cleaned from possible HTML-tags and disturbing special characters using the rubygem Sanitize.
- Part-of-speech tagging: The texts are tagged with its proper parts-of-speech using the rubygem EngTagger.
- Preparation of the tagged text: As EngTagger returns an XML string containing the result of the part-of-speech tagging process, the XML string is transformed into an array of hashes containing tag-token pairs for further processing. Each entry in the array contains a hash where the key of the hash represents the token (or word) and the value represents the belonging part-of-speech tag. The last three steps (including this one) are performed by the method *SREDataProcessor:tagtext()*.
- Product feature extraction: After the result of the part-of-speech tagger has been transformed into a format that is easier to handle, the product features are being extracted from the catalog descriptions and customer reviews. This is performed by the method *SREDataProcessor:extractwordgroups()*. The implemented algorithms for the product feature extraction task are explained in more detail in Section 5.3.1. After the product feature extraction was done for a certain catalog description or review text, the extracted features are inserted into the prototype database calling the method *add_product_features()* of the SREDBManager class.

5.2.3 SREDBManager

The SREDBManager class is responsible for communicating with the prototype database and is used by the other major components. In addition, the SREDB-Manager is responsible for the statistical analysis that is to record in the prototype database how frequently the extracted product features (the extracted terms) occur in the entire dataset and within the reviews and catalog descriptions of a certain review object. This is performed by the method *calculate_frequencies()* of the SREDBManager class that is called by SREDataProcessor during the product feature extraction process.

5.2.4 SRERecommendationEngine

The SRERecommendationEngine provides supplementary term recommendations that can be requested by calling the two methods *recommend_by_user_preferences()* and *recommend_by_query_suggestion()*. The latter method provides query suggestions (or completion) considering the entire dataset, whereby the method *recommend_by_user_preferences()* only concentrates on terms originating from the preference cluster the active user belongs to. This method also combines query suggestion with the collaborative filtering approach in certain situations. To identify the user cluster the active user belongs to the hashmap *preference_hits_map* is used. Whenever the active user articulates a new requirements his preference history is extended and with that the preference cluster the user belongs to could change. The hashmap *preference_hits_map* contains the level of the relation of the active user to each existing preference cluster (user cluster). The method *update_preference_hits()* is called whenever a new requirement has been articulated by the user to update the relations of the active user to each preference cluster. To do so, the SRERecommendationEngine implements the three different methods *update_preference_hits_criterion()*, *update_preference_hits_demand()* and *update_preference_hits_requirement()* considering different factors that influence the calculation of the preference relation. The preference cluster estimation and the generation of term recommendations is discussed in more detail in Section 5.4.

5.3 Product Feature Extraction

This Section describes the product feature extraction algorithm which is performed by the SREDataProcessor component. As mentioned in Section 5.2.2 the review texts and catalog descriptions are tagged with their parts-of-speech. Starting from situation where array of hashes that contains key-value pairs of tokens (words) and part-of-speech tags is available, the product feature extraction process will now be explained in more detail.

There are two kinds of terms that are extracted as product features:

1. Noun sequences: Such terms can consist of one or more subsequent nouns. Noun sequences that are extracted as product features can include, for in-

stance, terms like "*bathroom*", "*room service*", "*roof top terrace*", etc.

2. Adjective-noun pairs: Such terms can consist of one or more adjectives followed by one or more nouns. Adjective-noun pairs can include, for instance, terms like "*local cuisine*", "*excellent service*", "*personal service*", "*private airport transfer*", "*indoor swimming pool*" or similar.

The product feature extraction algorithm takes as input the array of hashes containing the word-tag pairs obtained from the part-of-speech tagging preprocessing step. The algorithm iterates through the array starting at the left-most word of the tagged text (position 0 in the array). At each position in the array the algorithm at first examines if the current token is a noun or a word. Depending on the part-of-speech of the current token, two different sub-algorithms are performed that are explained in the following section.

5.3.1 The Product Feature Extraction Algorithms

The different possibilities in writing style and used grammar had to be considered during the implementation of the product feature extraction task. As mentioned above the main algorithm iterates through the array and performs two different sub-algorithms depending on the part-of-speech tag of the current token.

Sub-algorithm 1

Whenever an adjective was found, this sub-algorithm (that is much easier to perform) is used. This sub-algorithm is able to extract product features that consist of one or more adjectives, followed by one or more nouns. To do so, the algorithm takes as input the array of tag-token pairs and the current position within the array. Then the sub-algorithm iterates through the array until a noun was found. Between the occurrence of the first adjective (the start position of this sub-algorithm) and the first noun that was found, there must only occur further adjectives or adverbs. Adverbs are ignored and not recorded as they are not informative. For instance, suppose a review sentence like "*We had an absolutely quiet, private and well-maintained beach.*". The interesting product feature would be "*quiet private well-maintained beach*". The adverbs "*absolutely*" can be ignored as it is not informative and would usually not be used to articulate a requirement.

In addition to adverbs commas (",") and conjunctions ("and", "or") may also occur between the first adjective (starting position of the sub-algorithm) and the first found noun. Reconsidering the example sentence "*We had an absolutely quiet, private and well-maintained beach.*" one can see, that several adjectives are used to describe the noun "*beach*". As commas and conjunctions are used to enumerate descriptive adjectives in English language, these parts-of-speech have to be permitted. Another implementation decision was made regarding numbers. In order that terms like "*24 hour room service*", "*2 persons*", "*2 weeks*", "*2 rooms*", etc. are not missed, numbers are considered as adjectives as well during the product feature extraction process. Although this leads to more incorrectly identified features, there are several features that would have been missed otherwise.

After the first noun was found, the sub-algorithm moves on as long as further nouns are found that directly succeed the last noun. Once the succeeding word is not a noun the sub-algorithm stops and a new product feature has been extracted successfully. The recorded adjectives and nouns are than concatenated in the correct order to build a complete token string that represents a product feature and with that a term that can be suggested to the user. If a sentence is finished and no noun was found after the occurring adjective (for instance, due to grammar or writing errors) the recorded adjectives are discarded and no product feature is extracted. The sub-algorithm returns the current position in the tag-token pair array to the caller to tell the main algorithm at what position to proceed. If the product feature was successfully extracted the sub-algorithm returns the very last position after the last noun that was found. If feature extraction process was interrupted the sub-algorithm returns the position in the array where the main algorithm should move next to continue the iteration. Figure 5.3 shows an example of a product review extract. The main algorithm would use sub-algorithm 2 to handle the words "*surroundings*" and "*service*" (as sub-algorithm 2 is used when a noun occurs) and sub-algorithm 1 as soon as the adjective "*friendly*" occurs which would successfully extract the feature "*friendly helpful staff*".

Sub-algorithm 2

This sub-algorithm is called by the main algorithm, implemented in the method *extract_product_features()* of the SREDataProcessor class, if the current word is

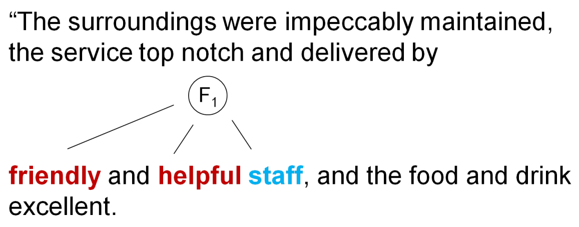

“The surroundings were impeccably maintained, the service top notch and delivered by
F_1
friendly and helpful staff, and the food and drink excellent.

Figure 5.3: Product Feature Extraction Example - Customer Review Extract, taken from (TripAdvisor.com, 2011b)

a noun. This sub-algorithm iterates through the array that was passed by the main algorithm and tries at first to find the longest continuous noun sequence. Additionally, this sub-algorithm tries to find adjectives that succeed a noun sequence and are linked to that noun sequence via a certain verb. In English language (and for sure in other languages) it is common to name a certain noun in a sentence and to use a subsequent verb in combination with adjectives to describe this noun. For instance, suppose sentences like "*The room was very spacious and comfortable.*", "*We had a beach that was totally clean and quiet.*", "*The staff was english-speaking and very professional.*", "*The Spa looked amazing and very relaxing.*". The linking verbs that were used in the current implementation of the prototype are: "*to be*", "*look*", "*taste*" and "*seem*". The proper forms of "*to be*" (was, were, is, etc.) are used the most frequent within the review texts.

To perform its task, sub-algorithm 2 uses some rules to decide whether the product feature extraction has to be terminated after an identified noun sequence was interrupted or not. The iteration proceeds until a linking verb has been found whereby some parts-of-speech (such as conjunctions like "*that*" or determiners like "*which*") are allowed between the detected noun sequence and the linking verb. This is necessary to allow the algorithm to correctly extract product features from sentences like "*We really liked our room that/which was very clean and modern.*". If, for instance, a new noun or a non-linking verb, is found the iteration is stopped and the sub-algorithm terminates. In that case the detected noun sequence is recorded and sub-algorithm 2 returns the very next position after the detected noun sequence to the main algorithm to tell it at what position to proceed with the iteration.

If sub-algorithm 2 correctly detects a linking verb after a noun sequence, he proceeds the iteration until one or more adjectives have been found. As in sub-algorithm 1 adverbs may occur in combination with adjectives, but they are ignored the same way. Adjectives that occur after a linking verb was detected must not be succeeded by a noun, because the recorded adjectives do not belong to the previously detected noun sequence. In that case the recorded adjectives are discarded and the detected noun sequence is recorded as a new product feature. After that sub-algorithm 2 terminates and returns the very first position after the recorded noun sequence to tell the main algorithm at what index to proceed the iteration. To correctly extract a product feature consisting of a noun sequence that is connected with one more adjectives via a linking verb, the next word after the last adjective must not be a noun. If so, a term is built from the noun sequence and the connected adjectives and extracted as new product feature. After that again sub-algorithm 2 returns the next position in the array where the main algorithm has to proceed.

5.3.2 Postprocessing

Although the product feature extraction algorithm has the advantage that many product features can be extracted from the customer reviews without using a pre-built taxonomy or ontology and without needed expert knowledge about the domain, it has the drawback, that all noun sequences and adjective-noun pairs are extracted. With that undesired terms that are no product features are extracted as well. To react to this problem, some postprocessing is performed to remove most of the false positives. To do so two postprocessing steps are performed:

1. Automatic postprocessing: This is done with a simple PHP script that performs some MySQL queries in the prototype database to automatically erase obviously wrongly identified terms that contain special characters such as ".", "?", "!" or similar. Such terms are for instance "*amazing staff.they*" or "*day+The*" that occured due to writing or typing errors in the product reviews. The used part-of-speech tagger tends to identify unknown words as nouns, therefore such terms are tagged as nouns and extracted by the product feature extraction algorithm
2. Semi-automatic postprocessing: In addition to terms that are extracted due to writing or typing errors, there are other terms that are real words but no product features. Because the prototype database is a quite big number of terms, a manual clean-up would be very impracticable. Fortunately undesired

terms (that are primarily noun sequences) like "*everything*", "*anything*", "*my husband*", "*next day*" occur very frequent. In addition, they belong to the most frequent terms in the entire dataset. Thus, these terms can be erased by viewing the most frequent terms and the entire dataset and selecting the obviously wrongly identified ones by hand. Using simple MySQL queries the selected terms can be removed from the prototype dataset.

5.4 Implementation of the Recommendation Process

In general, the prototype generates a list of suggested terms during the recommendation process by using the different methods *recommend_by_query_suggestion()* and *recommend_by_user_preferences()* of the SRERecommendationEngine class (see Section 5.2.4). These two methods can be called to request additional term suggestions during the requirement formulation process. The tasks performed in these methods are explained in the following sections.

5.4.1 Query Suggestion

This approach adapts the general functionality of query suggestion (or query completion) that is done by search engines. Thereby the criteria and demands as well as the entire requirements that are articulated by the user in the Xohana user interface are considered as search queries. The query suggestion compares the user input entered during the requirement articulation process to all complete terms (or items) that are available in the entire prototype database to make query suggestions - or in that case to make term suggestions. There are two major application cases where the query suggestion method is performed using the entire prototype dataset:

1. The user articulates a new criterion: Whenever the user articulates a new criterion, the query suggestion method is called as soon as the user typed at least three characters. Three characters have shown to be a good number during the implementation and testing phases of the prototype. The entered characters are then used to search the prototype database for complete terms containing the incomplete term.

2. The user articulates a demand for the entered criterion: Again, two different cases have to be distinguished. Either the user has just focused the demand field in the user interface, or the user has already type three characters in the demand field. If the user just focused the demand field, the prototype database is searched for terms that match the already entered criterion. If the user already started typing in the demand field, the prototype database is searched for terms that match a combination of the criterion and the incomplete demand consisting of the characters already typed into the demand field.

In both application cases wildcards are used to search the MySQL database. MySQL supports pattern matching with wildcards using the *LIKE* command. The *%* symbol represents the wildcard character that allows to match any character within a certain pattern. For instance, if the user defines the criterion "*beach*" the search patterns *beach% %* and *% beach%* are used. The second pattern can be used to find and suggest terms like *clean beach*, *romantic beach*, *private beach* or *quiet beach* whereas these terms consist of adjective-noun pairs. The first pattern on the other hand does not find adjective-noun patterns, as the first word in the terms that are matched by that pattern always has to be the word *beach*. Nevertheless, there are terms that consist of noun sequences that are related to the defined criterion such as *beach access*, *beach transportation service*, *beach chairs* or *beach parties*.

Suppose another example where the user is articulating the criterion of the requirement and has already typed the three letters *roo*. Using the flexible wildcard search patterns mentioned above, the user can be suggested terms like *room service*, *hotel room*, *spacious room*, *separate sitting room*, *non-smoking room*, *oceanview room* or similar.

Before the suggested terms are displayed to the user, they are prepared to only show the currently relevant part of the complete term. That means, if the user is articulating a criterion, only the noun sequence of the complete term is shown (if the term consists of adjectives and nouns). On the other hand, if the user articulates the demand, only the adjectives are shown (if the term consists of adjectives and nouns). If the term only consists of nouns, the entire noun sequence is shown. For instance, if the term *wireless internet* is suggested from the item dataset of the prototype database, the user is shown the noun *internet* in the criterion field and likewise the adjective *wireless* in the demand field.

To handle criteria like *room facilities*, *hotel facilities*, *sports and leisure activities*, *medical services* or similar criteria that indicate certain amenities of the accommodation, the term matching is performed differently. In that case terms (product features) that have been extracted from catalog descriptions are preferred over those extracted from customer reviews as it is almost impossible to find terms that contain such criteria.

The term recommendation of the prototype uses some simple weighting strategies to sort the list of suggested terms. Terms that contain adjectives and nouns are weighted a little higher than terms that only consist of noun sequences. Additionally, the recommendation methods consider how frequent a term occurs within in the entire prototype database in the weighting calculation in order to weight frequent terms higher than infrequent. The result of the query suggestion (here term suggestion) method *recommend_by_query_suggestion()* of the SRERecommendationEngine class is a list of up to 50 suggested terms ordered by the calculated weighting.

5.4.2 Collaborative Recommendations

As mentioned in Chapter 4 the collaborative filtering approach of the recommender system prototype assigns the current user to one of the obtained user preference clusters considering the articulated requirements. The tasks performed by the method *recommend_by_user_preferences()* of the SRERecommendationEngine class restricts the suggested items to terms that have been preferred by like-minded users. As mentioned in Section 4.3.2 term recommendations are generated based on the preference cluster to what the active user has the highest relation considering the co-rated items (represented by the articulated requirements). This section explains how the user preference estimation was implemented and how the collaborative recommendations are generated.

User Preference Estimation

To estimate the preferences of the active user, the relation to each preference cluster is calculated. This done by calling the method *update_preference_hits()* of the SRERecommendationEngine class whenever the active user has articulated a new requirement. To do so, a hashmap is used that records the current relation value to the active user for each cluster. For instance, if there are 5 user preference clusters (obtained from 5 different review objects), the hashmap has 5 entries of key-value pairs. The keys represent the ID of the review object (that corresponds to a user preference cluster). The value represents the strength of the relation of the active user to the cluster. The articulated requirements are considered as unary rated items of the active user whereas the terms that were extracted from the product reviews are considered as unary rated items of the corresponding reviewer. The unity of all unary ratings on the different items (terms) performed by the users in a certain cluster are regarded as pooled ratings. Each articulated requirement of the active user that can be matched to a term (item) in the prototype database is considered as a co-rated item. All terms that can be matched to a requirement belong to a certain review object (and with that to a certain cluster). As the preferences of the users in the clusters are very similar, the calculation of the relation between the active user and the clusters is simplified. The more co-rated items the active user has with the collectivity of the users in a certain preference cluster, the higher becomes the relation of that user to that particular cluster.

For instance, if the active user articulated the requirement "*bathroom* should be *outdoor*" he provided a unary rating on the term (or item) *outdoor bathroom*. For each cluster, that contains on or more users that also preferred an *outdoor bathroom* the relation to the active user becomes stronger by increasing the corresponding value in the hashmap.

As the articulated requirements consist of criteria with a certain demand it is useful to attempt to match them with terms in the prototype database as well instead of matching only a complete requirement. Criteria and demands are also dealt as complete requirements during the preference elicitation task. However, the relation to a preference cluster is increased less in that case as matching complete requirements as co-rated items indicates a much higher relation. To consider that, the SRERecommendationEngine distinguishes between three cases:

1. Calculation by complete requirement: If a complete requirement consisting of criterion and demand is found as co-rated item, the relation value is increased by 4.0, which is an experience value.
2. Calculation by criterion: If only the criterion was identified as a co-rated item,

the relation value of the corresponding cluster is increased by the value 0.8, which is an experience value.

3. Calculation by demand: If only the demand was identified as a co-rated item, the relation value of the corresponding cluster is increased by the value 1.0, which is an experience value. If the criterion is a term like *room facilities*, *hotel facilities*, *sports and leisure activities*, *medical services* or similar terms that are used to list various accommodation amenities, the demand is usually a complete term itself. Therefore the value indicating the strength of the relation is increased the same way as for complete requirements consisting of criterion and demand.

With every articulated requirement the preference elicitation becomes more and more accurate, because there are usually more hits in a certain cluster. The current preference cluster the active user has the highest relation to can change whenever a new requirement was articulated. As the calculation of the relation is iterative the values do not have to be recalculated with every new articulated requirement. If the relation value of several clusters is even, one is chosen by random. From experience during the test phases of the implementation this usually only happens at the initial phase if only a few requirements have been articulated.

Generating collaborative recommendations

As soon as the hashmap that calculates the relations between the active user and existing user preference clusters was built, the SRERecommendationEngine class is ready to provide additional collaborative term recommendations that can be required using the public method *recommend_by_user_preferences()* of the SRERecommendationEngine class. To query for the current preference cluster the method *get_preference_cluster()* of the SRERecommendationEngine class is called. The collaborative recommendations are generated from terms that have been rated by users within that certain preference cluster. To generate a list of recommended terms, two different application cases are distinguished:

1. The user articulates a new criterion: Whenever the user starts to articulate a new criterion by focusing the empty criterion field, collaborative recommendations are requested from the SRERecommendationEngine class. The list

of recommended terms is then created from different sublists. As the terms extracted from the catalog description represent the preferences all users in the current cluster have in common, a selection of these terms is added to the entire recommendation list whereby terms containing adjectives and terms consisting of at least two words are preferred. In a next step terms that have been rated the most frequent within the current preference cluster are chosen whereby again terms containing adjectives and terms consisting of at least two words are preferred. If the user has typed at least three characters in the criterion field, the collaborative recommendation process is combined with the query suggestion approach in the same way as described in Section 5.4.1. That means, the generated recommendations are limited by the current preference cluster and the incomplete criterion string.

2. The user articulates a demand for the entered criterion: If the user has just focused the demand field in the user interface the prototype database is searched for terms that match the already entered criterion, restricted to the current preference cluster whereby terms that have been rated the most frequent within the current cluster are chosen. Terms containing adjectives are preferred over terms consisting only of noun sequences. In addition, terms extracted from the catalog description that represent the preferences that all users in the current cluster have in common are also selected whereby terms containing adjectives and terms consisting of at least two words are again preferred. Terms extracted from the catalog description are more preferred if the entered criterion indicates the description of accommodation amenities.
3. The user starts typing the demand string for the entered criterion: If the user has typed at least three characters in the demand field simply the most frequent terms (matching the already defined criterion in combination with the incomplete demand entered by the user) within the current cluster are chosen.

5.5 Conclusion

This Chapter explained in more detail how the recommender system prototype was implemented to achieve the goals of this work. Various established technologies and techniques have been used to develop the prototype. The ruby framework helped to facilitate the prototype implementation especially considering the communication with the MySQL database of the prototype by using the ActiveRecord library. The part-of-speech tagger library EngTagger has shown to perform good enough for the preprocessing task to the product feature extraction process. The product feature extraction approach relies on certain grammatic rules and can therefore only perform adequate as long the used product reviews show correct grammar and writing style which was the case in nearly all reviews. One weakness of the product feature extraction algorithm is that there are many terms that are indeed correctly identified from the technical point of view, although they are no product features. Nevertheless the algorithm does not miss many of the potential product features. Additionally most of the incorrectly identified terms could be erased with the post-processing tasks. Moreover, with the application of the query suggestion approach the incorrectly identified terms are usually not suggested because the possible terms are constrained by the search string. MySQL has shown to be the right choice to implement the prototype database especially considering the possibility of using wildcards in the query search patterns which was essential to allow a more flexible matching between the articulated requirements and the terms in the prototype database. In addition, the indexing feature of MySQL is very useful to speed up the queries of the quite large prototype dataset which is very important for the rather complex queries that are performed especially during the generation of collaborative recommendations.

In order to estimate the potential of the implemented prototype to optimize the recommendation process of the existing personalization system, an evaluation with different use-cases was performed. The setup and process of the evaluation as well as the results are presented in the next chapter.

6. Evaluation and Results

To evaluate the potential of the recommender system prototype in assisting the recommendation process of the target system, especially in cold-start situations, 5 use cases have been performed. The goal of the evaluation was to show how good the recommendation process of the target system could be improved by applying the recommendations provided by the recommender system prototype in addition to those already provided by the target system. Therefor, it was determined how many of the suggested terms were useful to the user and with that, successfully recommended. In order to provide an evaluation concept with a practical orientation, 5 different testruns of the recommender system prototype in collaboration with Xohana have been performed that represent the entire process of articulating a customer's desired type of vacation. Because the process of determining how many terms were successfully recommended includes inspecting the complete list of suggested terms, it would have been too complicated to make the evaluation with real users. Therefore the evaluation was performed with simulated use-cases. The next sections describe the character of the used training and test data and how the evaluation was processed. The last sections of this Chapter present and discuss the results of the evaluation.

6.1 Training Data

The prototype database that serves as training data was built from 1.717 customer reviews collected from TripAdvisor.com that were written to rate 8 different accommodation objects. To avoid the necessity of opinion mining, only outstanding positively rated accommodation objects with the highest possible rating (5 points out of 5 on the rating scale) have been chosen. All used reviews are written in

English language. As mentioned before (see Section 4.3.2) the different accommodation objects represent the different user preference clusters. All vacations the different reviewers have been on, were different kinds of summer holidays. The different preference clusters vary in their characteristics and provide different preference stereotypes corresponding to the different types of summer vacations and features of the reviewed accommodation objects:

- Preference Cluster 1: Holiday in a luxury hotel in the center of Marrakech, providing a traditional authentic Moroccan experience. The vacation fits for people who prefer upscale accommodations and do not travel with young children.
- Preference Cluster 2: Holiday on an Asian tropical island. This vacation fits for people that prefer to be located next to beautiful sand beaches and to reside in an accommodation that provides enough space and outdoor areas as well as leisure facilities and possibilities for trips and excursions.
- Preference Cluster 3: Holiday on Bali, for people that prefer various relaxing facilities, such as spas and healing packages, and much privacy in a natural environment.
- Preference Cluster 4: Holiday in the Dominican Republic, for people that prefer all-inclusive packages with many entertainment and sports facilities and access to a private beach.
- Preference Cluster 5: Holiday on the Seychelles, for people that prefer marine facilities such as fishing and diving. This kind of vacation escpecially fits for honeymooners that prefer idyllic and romantic settings.
- Preference Cluster 6: Holiday in Phuket, for people that prefer outstanding customer service, very professional staff and family friendliness.
- Preference Cluster 7: Holiday in Nice, for people that prefer a very good location, a tastefully decorated accommodation style, stunning sea views and appreciate gourmet cuisines.
- Preference Cluster 8: Holiday in South Beach Miamy, for people that prefer fashionable accommodations with very personal customer service and modern business facilities.

During the product feature extraction task 40.773 unique terms have been extracted from the customer reviews and catalog descriptions. To erase as many wrongly identified terms as possible, some post-processing tasks were performed (as described in Section 5.3.2). With that 5.507 (13.52 %) of the extracted terms could be filtered out as wrongly identified terms resulting in about 35.266 remaining terms that were used as training data for the evaluation. Of that remaining terms 2182 were extracted from the 8 catalog descriptions and 33.084 terms were extracted from the 1.717 customer reviews.

6.2 Test Data

To generate the use case simulations with adequate test data, 5 outstanding positively rated customer reviews about accommodation objects for summer holidays and the 5 belonging catalog descriptions have been used. Thereby the articulated requirements for each testrun were built from features of the reviewed object that the customer (the reviewer) considered as positive and noteworthy and from the general features that are described in the catalog description. To avoid biasing of the results by making sure that the used test data differes from the training data, the customer reviews and the belonging catalog descriptions of the reviewed objects have been chosen from a different Webportal, HolidayCheck.com[1], in order to build the requirements for the use-case simulations. During the use case simulations the requirements that were built from the product review and the belonging catalog description were articulated in that order in which they occured in the review text and the catalog description. The features described in the catalog description have been used based on the quite obvious assumption that a customer that was satisfied with a certain review object also considered the features listed in the belonging catalog description as important, because these features have been known beforehand very likely.

The extracted features served as terms to articulate the requirements in the use-case simulations. The terms that were extracted from the catalog description were selected with consideration to the kind of reviewer. For instance, for a single vacationist hotel facilities like "*babysitting*" or "*child care*" would not be interesting. As both review texts and catalog description are usually very extensive, it was attempted to use the most relevant terms in order to obtain a reasonable number

[1]http://www.holidaycheck.com HolidayCheck.com homepage, last access 04/2011

of requirements. For each testrun overall 26 requirements (with a median of 26 and a standard deviation of 1) were manually extracted from both the review text and the belonging catalog description. The extracted requirements for each use-case simulation are shown in the Appendix (see A).

6.3 Evaluation Process

All testruns were performed on a machine with Windows 7 64-Bit operating system, a Intel Core i5 CPU 2.67 dual core processor and 4GB physical memory. The target system Xohana did not use any user histories during the evaluation and was therefore in a cold-start situation. To simulate the use-cases, all requirements (that means all criteria and demands) were manually entered via the user interface of Xohana. Requirements extracted from the customer review were articulated at first followed by those extracted from the catalog description.

To evaluate the results the number of terms that were successfully recommended to the user were recorded in every testrun. That means, whenever a term that was useable to articulate a criterion or demand could be identified in the list of all suggested terms, a term was considered as successfully recommended. Synonymous terms were also considered to create realistic conditions. That means if the user wants to define the requirement "*food* should be *vegetarian*" and he is suggested the word *meal* for the formulation of the criterion, this term is also considered as successfully recommended, as human users can easily recognize synonymous words.

The overall success rate results from the number of hits (successfully recommended terms) that could be obtained by Xohana in collaboration with the recommender system prototype. As the main objective of this work is to support the recommendation process of an existing personalization system the potential of improvement through the application of the prototype had to be measured. Whenever one of the suggested terms was chosen, the origin of the suggested term was recorded in order to know if the term was recommended by Xohana or the recommender system prototype. With that, the increase of successfully recommended terms through the application of the recommender system prototype could be measured. If no term could be successfully suggested by both Xohana and the prototype during the articulation of the current requirement, the first three characters of the requirement (criterion or demand) were typed to obtain new term suggestions from Xohana and the prototype.

The number of term suggestions shown to the user via the Xohana user interface is limited to a reasonable amount due to usability issues. If there are more candidates for the term suggestions as the maximum number of terms that are shown to the user, a selection is chosen by random whereby terms with a higher relevance (that is terms that are higher weighted) are preferred. In order to get deterministic results, all recommendations made by both Xohana and the recommender system prototype (the query suggestion returns up to 50 recommended terms and collaborative filtering method of the recommender system prototype returns up to 100 recommended terms) were considered during the evaluation phase. To view all suggested terms in every recommendation step, the lists of suggested terms obtained by both Xohana and the prototype were logged to a textfile using different debugging methods. Whenever new terms are suggested, the logfile can be viewed to check whether a term of the requirements that were extracted from the customer review and correspoding catalog description test data was successfully recommended by Xohana and/or the prototype. For instance, if the current requirement is *room facilities* should be *wireless internet*" and the user has already articulated the criterion "*room facilities*", the focus is set on the demand field in the input interface. With that new term recommendations are produced by Xohana and the prototype. To check for possible hits, the current suggestion lists in the logfile are searched for the term "*wireless internet*". If, for instance, the term was found in the recommendation list of the prototype, a hit is recorded for the prototype. As synonymous terms are considered as well the identification of terms like "*wifi*" in the recommendation list would also result in a new hit.

6.4 Results

Table A.1 (see A) shows the results of the first test run. The requirements were built from a customer review (HolidayCheck.com, 2011e) at HolidayCheck.com and the corresponding catalog description (HolidayCheck.com, 2011j). The table headings indicate what criterion and what demand has been successfully recommended by Xohana and the prototype. The used search-terms that were typed in the criterion or demand input fields of the user interface, if no terms could be successfully recommended, are shown in the rightmost fields of the table. The used abbreviations are explained in the Appendix (see A). To calculate the optimization potential the total number of possible hits (successfully recommended terms for criteria and demands) is calculated at first. The optimization potential is then calculated as the difference of the hit ratio performed by Xohana in collaboration with the prototype and the hit ratio of Xohana only. The hits that were made on the same term suggestion by both the prototype and by Xohana do not affect the result, as only the increase of hits made by the prototype is measured. For instance, in the case of use-case simulation 1 there are 50 possible hits (as there are 25 articulated requirements and with that 25 criteria and 25 demands). As Xohana achieved in total 33 hits, it had a hit ratio of 66.00 % during the first test run. The recommender system prototype achieved in total 27 hits, whereby 14 hits for the same term suggestion were already achieved by Xohana. This means a growth of 14 hits which results in a total of 47 hits that could be obtained by Xohana in collaboration with the recommender system prototype which again results in a hit ratio of 94.00 %. This yields in a potential improvement of 28.00 % through the application of the recommender system prototype.

The other 4 testruns have been performed in the same way. The result tables of all 5 use-case simulations containing the detailed calculation of the optimization potentials measured in each testrun and the list of all articulated requirements are shown in the Appendix (see A). Table 6.1 summarizes the results of the 5 use-case simulations. The abbreviations used in the table are listed above:

- ***TR***: number of test run
- ***PH***: total of possible hits
- ***C X***: hits for criteria recommended by Xohana
- ***D X***: hits for demands recommended by Xohana
- ***H X***: total hits of Xohana
- ***HR X***: hit ratio of Xohana in %
- ***C XP***: hits for criteria recommended by Xohana in collaboration with the prototype
- ***D XP***: hits for demands recommended by Xohana in collaboration with prototype

Table 6.1: Use-case Simulation Results

TR	PH	C X	D X	H X	HR X	C XP	D XP	H XP	HR XP	OP
1	50	21	12	33	66.00	24	23	47	94.00	28.00
2	52	18	14	32	61.54	26	26	52	100.00	38.46
3	54	20	13	33	61.11	27	25	52	96.30	35.19
4	54	23	16	39	72.22	27	23	50	92.59	20.37
5	50	18	12	30	60.00	23	22	45	90.00	30.00
AVG	52	20	13.4	33.4	64.17	25.4	23.8	45	94.58	**30.40**

- ***H XP***: total hits of Xohana in collaboration with prototype
- ***HR XP***: hit ratio of Xohana in collaboration with prototype in %
- ***OP***: optimization potential in %
- ***AVG***: Average

Considering all 5 use-case simulations an average hit ratio of about **94.58 %** could be performed by Xohana in collaboration with the recommender system prototype. The improvements of the recommendation process of Xohana in all simulated use-cases, that could be achieved through the application of the recommender system prototype, result in an average optimization potential of about **30.40 %**.

6.5 Conclusion

The average optimization potential obtained from all 5 testruns leads to a very satisfying total of about 30 %. From the use case simulations it is obvious that with the application of search-strings (first 3 characters of a searched term) terms could be recommended in the clear majority of all cases. During the manually performed test phases it could be observed that also the collaborative filtering approach performed quite well without using query suggestion with search-terms. It is particularly noticeable that the recommender system prototype showed very good results during the articulation of special user needs such as "*balcony* should be *private*", "*bathroom* should be *spacious*" or "*staff* should be *english speaking*". The major improvements of the hit ratio could be achieved in the suggestion of demands. During the criterion articulation the support via the recommender system prototype was not that needed, but showed to be very useful for handling rather special, personal and rare criteria. Considering this and the outcome of the evaluation with an optimaziation potential of about 30 %, the recommendations provided by the prototype show to be a very good supplement to those provided by the existing system, particularly in cold-start situations.

7. Discussion and Lessons learned

Over the years, various types of recommender systems have emerged that all have their advantages and drawbacks. From research accomplished in this work, the insight was gained that there is no perfect recommendation technique for all application cases. The right choice depends on the goals and tasks of the system and as well on the available knowledge and structure of the data that is used. Content-based filtering approaches have the advantage that as soon as the user preferences have been estimated, items that have similar properties as those the user preferred in the past can be recommended (Shih & Liu, 2005). This can be done without collecting preference profiles of other users in the system. Moreover content-based approaches do not suffer from new item cold-start problems as they have access to item features. Nevertheless, most content-based approaches suffer from over-specialization that is they are not able to recommend new items that are not similar to those the user liked in the past (Iaquinta et al., 2008). In addition an adequate amount of ratings by the user has to be available to make valuable recommendations (Blanco-Fernandez et al., 2008; Felfernig & Burke, 2008).

Collaborative filtering approaches have the advantage that they do not need to understand the content or structure of an item. Recommendations can be generated based on the preferences of other like-minded users (Sarwar et al., 2001). However, collaborative filtering approaches show poor performance in cold-start situations when rating data tends to be sparse (Devi et al., 2010).

Demographic approaches use demographic information to learn relationships between items and the type of users who preferred it. The implementation of demographic-based approaches can be done quick and easy, but the success of such systems depends on the completeness of the demographic data. (Nageswara Rao & Talwar, 2008)

Knowledge-based approaches aim to draw conclusions about how a particular item conforms to a particular user requirement by making inferences and using functional knowledge (Fürnkranz & Hüllermeier, 2010, p. 395). Although knowledge-based approaches do not suffer from cold-start problems, they have the drawback that the required knowledge is occupied by domain-experts and difficult to obtain and transform into an adequate representation. (Ricci et al., 2010)

In this work, existing data from heterogeneous sources has been used to support the recommendation process of the existing personalization system Xohana in cold-start situations. With that the existing knowledge source of the target system could be supplemented to provide optimization potentials for the recommendation process. Based on the insight, that the right choice of an adequate recommendation technique depends also on the consistency of the available data, collaborative filtering has shown to be the right choice. The suggested items are terms that are used to help the user to articulate personal requirements and needs. As such terms do not provide certain features that allow to calculate item similarities, content-based filtering techniques are not appropriate. Because the recommender system prototype aims to provide additional recommendations in cold-start situations, every user has to be considered as a new user. Therefore no demographic information about the user is available that is needed by demographic approaches. However, the approach proposed in this work uses an adequate amount of product reviews that provide, if processed appropriately, enough information about user preferences that can be used to supply the data source of a collaborative filtering system.

Regarding the product feature extraction method an interesting insight about the character of the used customer reviews could be obtained. The first attempt of this work was to use both negatively and positively rated reviews. During the inspection of the review data positively rated reviews have shown to provide more different extractable features. Very unsatisfied customers tend to complain about a few things they disliked without articulating many facts about the review object. Additionally, opinion mining would be needed to distinguish preferred from unpreferred items which unnecessarily complicates the product feature extraction task. Moreover, as users in the target system only articulate things they prefer, it is not relevant to extract disliked items.

To extract the relevant features from the product reviews that are used to build the database for the recommender system prototype, part-of-speech tagging has shown to be a powerful information extraction tool to identify the information of interest. Although, the used part-of-speech tagger was not trained to work on customer reviews in the tourist domain and had therefore some problems with word-ambiguities, it performed more than satisfying for the product feature extraction task. The strength of the product feature extraction approach of the prototype is that a very high fraction of features could be extracted from the review texts and catalog descriptions without using any kind of previously constructed taxonomy or ontology and without special domain knowledge. The drawback of that approach is that also noun sequences or noun-adjective pairs are correctly identified from the technical point of view, although they are no product features. Nevertheless, this problem was mitigated quite well with adequate postprocessing methods that filtered out most of the incorrectly identified features. Moreover, the wrongly extracted terms that could not be removed does not diminish the success of the recommender system prototype.

The cluster-based approach applied by the recommender system prototype allows a quite fast and easy identification of customers that share the same preferences as the active system user. With the combination of the collaborative recommendation process and the query suggestion approach, the number of relevant items the active user might potentially prefer can be limited more effectively. In future work, additional neighborhood selection within the given preference cluster would be conceivable to provide even more finer-grained recommendations. To improve the implemented product feature extraction process the application of a part-of-speech tagger that is trained on texts in the tourism domain would also be recommendable.

Considering the results of the performed use-case simulations the approach proposed in this work shows a very good optimization potential for the recommendation process of the target system Xohana, especially in cold-start situations. With that, the main objective of this goal and the three associated goals could be successfully achieved. During the evaluation it was observeable that the prototype showed a good supportive functionality especially for articulating special user requirements that are not easy to cover with an initially built knowledge base. Given that, the first associated goal, which was to supplement the knowledge base of the existing system with additional valuable terms, could be successfully reached. By the combination of collaborative filtering and query suggestion many useful personalized term recommendations could be provided by the recommender system prototype. Especially during the articulation of very personal criterions, collaborative recommendations based on the user histories built from the customer reviews could be successfully provided even without using search terms. With that, the second associated goal, which was to provide additional personalized recommendations, could be achieved. As the target system was in a cold-start situation during the entire evaluation process, the very promising optimization potential of about 30 % shows that the third associated goal, to provide another solution to address cold-start problems, could be reached as well.

8. Summary and Future Work

Many recommender systems still have to face data sparsity and especially cold-start problems in initial phases where no or not enough information about system users has been collected. In the course of this thesis a recommender system prototype was implemented that aims to assist the recommendation process of the existing personalization system Xohana, in the domain of tourism, with additional, collaborative recommendations, especially in cold-start situations.

In the first part of this work in depth research about established methods and current approaches in the fields of natural language processing, with focus on information extraction from product reviews, and recommender systems, with focus on cold-start problems and collaborative filtering, has been conducted. In the second part of this thesis, based on the obtained insights from research, a concept was developed that meets the requirements and goals of this work. In the third part, the prototype was implemented based on the proposed concept followed by an evaluation of the resulting application. The implemented prototype has more or less two major tasks: Product feature extraction from customer reviews and provision of additional recommendations by applying query suggestion and collaborative filtering based on user preference clusters.

To provide an alternative knowledge base, product features and information about user preferences have been extracted from a small but for this work sufficient amount of customer reviews at TripAdvisor.com. As in other current approaches dealing with product feature extraction from customer reviews, part-of-speech tagging also showed to be a very valuable preprocessing tool for product feature extraction. The product feature extraction approach has the advantage that a high fraction of the potential product features can be extracted automatically from the customer reviews that are written in natural language without using any previously defined data (such as taxonomies or ontologies) and without the need of special domain knowledge. Nevertheless, the product feature extraction approach has a drawback. As all noun-sequences and adjective-noun pairs are automatically extracted as product features, there are also incorrectly identified product features. However, this problem could be mitigated with adequate postprocessing tasks that helped to filter out most of the false positives.

The recommendation task of the prototype performed very satisfying. With the user histories and rating data inferred from the customer reviews, a collaborative recommendation approach that is based on naturally given user preference clusters, could be implemented. Also query suggestion could be implemented successfully by using the extracted product features (or items) from the customer reviews. The generated recommendations showed to be a very valuable supplement to those generated by the existing personalization system Xohana. The considerable advantage of this recommender system approach is, that recommendations can be made based on user histories extracted from the used customer reviews even if the target system is in a cold-start situation where those user histories are missing.

Considering the very promising results from the evaluation, the concept of applying query suggestion and a collaborative filtering approach based on user preference clusters extracted from customer reviews, to support the recommendation process of an existing personalization system in cold-start situations, has proven to be quite efficient. The estimated optimization potential of about 30 % opens possibilities for further research and future work on this topic.

There are some concrete recommendations to improve the performance of the prototype. As the efficiency of the product feature extraction approach is limited to the capabilities of the used part-of-speech tagger, applying a part-of-speech tagger that is trained on review texts in the tourist domain would help to weaken problems with word ambiguities. To enhance the accuracy of the recommender system prototype, a finer-grained identification of like-minded users could be conducted again within the selected preference cluster. This should be done with consideration of the resulting computation effort. Additionally the weighting strategies of the recommender system prototype could be improved, to make more accurate predictions of the recommended items within the sorted list of suggested terms. Another idea is to consider synonymous terms during the recommendation process of the prototype.

As the prototype is syntactically matching the terms that are entered by the user to that in the prototype database, different terms that have the same meaning are not detected. This problem could be mitigated by using a synonym database.

To take advantage of the proposed approach, the developed ideas could also be adapted to other types of recommender systems. The approach would best fit to be applied in domains, dealing with very complex products. Reviews about those products need to provide enough data and information about subjective user preferences. For instance, customer reviews about educational institutions like colleges or universities that are available on the Internet, would meet these requirements. The reviewers are students that have or had personal experience with the institution and articulate what they prefer and dislike about the reviewed object. This is very similar to the way customer reviews about accommodation objects are arranged that, considering the results obtained in this work, already proved to provide valuable information for recommender systems.

A. Results of the Use-case Simulations

A.1 Used Abbreviations

- ***C X***: criterion suggested by Xohana
- ***C P***: criterion suggested by prototype
- ***D X***: demand suggested by Xohana
- ***D P***: demand suggested by prototype
- ***C S***: used search-term for criterion
- ***D S***: used search-term for demand

A.2 Use-case Simulation 1

- Short description of the vacation: Safari-Trip with mobile Camping, Region Kenya, Africa

- Accommodation description at HolidayCheck.com (HolidayCheck.com, 2011e)

- Customer review at holidaycheck.com (HolidayCheck.com, 2011j)

Table A.1: Use-case Simulation 1

Articulated requirements, total: 25	**C X**	**C P**	**D X**	**D P**	**C S**	**D S**
climate should be tropical	1	0	1	1	-	-
interests should be photo shoot	0	0	0	1	"int"	"pho"
service should be exceptional	0	1	0	1	"ser"	-
meal/food should be breakfast	1	0	1	1	-	"bre"
meal/food should be lunch	1	0	1	1	-	"lun"
holiday/campsite facilities should be safari	1	0	0	1	-	"saf"
meal/food should be dinner	1	0	1	1	-	-
staff/service/personal should be tour guide	0	1	1	1	-	"tou"
drinks should be cool/cold	1	1	0	1	-	-
transportation should be comfortable	1	1	0	1	"tra"	-
meal/food should be picnic lunches	1	0	0	1	-	"pic"
meal/food should be fresh	1	1	0	1	-	-
meal/food should be local	1	0	0	1	-	-
room facilities should be shower	1	0	1	0	-	-
room facilities should be toilet	1	0	1	0	-	-
room type/lodging should be camp/camping	1	0	0	1	-	"cam"
room/room type should be double bed/room	1	1	1	1	-	-
bed/beds should be kingsize/queensize	1	0	1	1	-	-
hotel facilities should be bureau/writing table	1	0	1	0	-	"wri"
hotel facilities should be lounge	1	0	1	0	-	-
drinks should be complimentary/free	1	0	0	1	-	-
dinner should be a la carte/dinner menus	0	1	0	1	"din"	-
entertainment/activities should be game drives	1	1	0	0	-	"gam"
entertainment/activities should be balloon ride	1	0	0	0	-	"bal"
hotel facilities should be pool/swimming pool	1	0	1	0	-	-
Number of hits	**21**	**8**	**12**	**18**		

Based on the hits the following values can be calculated:

- Possible hits (criteria + demands): 50
- Hits criteria Xohana: 21
- Hits demands Xohana: 12
- Total hits Xohana: 33
- Hit ratio Xohana: 66.00%
- Hits criteria Xohana + prototype: 24
- Hits demands Xohana + prototype: 23
- Total hits Xohana + prototype: 47
- Hit ratio Xohana + prototype: 94.00 %
- **Optimization potential: 28.00 %**

A.3 Use-case Simulation 2

- Short description of the vacation: Honeymoon, Region Thailand, Asia
- Hotel description at HolidayCheck.com (HolidayCheck.com, 2011c)
- Customer review at HolidayCheck.com (HolidayCheck.com, 2011h)

Table A.2: Use-case Simulation 2

Articulated requirements, total: 26	**C X**	**C P**	**D X**	**D P**	**C S**	**D S**
holiday should be honeymoon/honeymooning	0	1	0	1	"hol"	"hon"
location should be near/close to beach	1	1	1	0	-	-
transfer/transportation should be taxi	0	1	0	1	"tra"	-
staff should be attentive	0	1	0	1	-	-
staff/people should be english speaking	0	1	0	1	-	"eng"
food should be asian/thai	1	1	0	1	-	-
meal/food should be breakfast	1	0	1	1	-	-
dinner should be romantic	0	1	0	1	"din"	-
pool should be lounge chair/pool chair	0	1	0	1	"swi"	"cha"
pool should be umbrella	0	1	0	1	-	"umb"
room facilities should be dvd player	1	0	1	1	-	-
hotel facilities should be library	1	1	0	1	-	"lib"
room location should be ocean/beach (view)	1	0	1	0	-	-
tv should be flat-screen	1	0	0	1	-	"fla"
hotel facilities should be free parking	1	0	1	1	-	-
room facilities should be air condition	1	0	1	0	-	-
room facilities should be safe	1	0	1	0	-	"saf"
balconies/balcony should be private/own	0	1	0	1	"bal"	-
room facilities should be minibar	1	0	1	0	-	-
hotel facilities should be massage service	1	0	1	1	-	"mas"
hotel facilities should be laundry (service)	1	0	1	1	-	-
hotel facilities should be car rental	1	0	0	1	-	"car"
room facilities should be bath tub	1	0	1	0	-	-
room facilities should be shower	1	0	1	0	-	-
activities/sports should be bicycle hire	1	0	1	0	-	"bic"
hotel facilities should be whirlpool	1	0	1	0	-	"whi"
Number of hits	**18**	**11**	**14**	**17**		

Based on the hits the following values can be calculated:

- Possible hits (criteria + demands): 52
- Hits criteria Xohana: 18
- Hits demands Xohana: 14
- Total hits Xohana: 32
- Hit ratio Xohana: 61.54 %
- Hits criteria Xohana + prototype: 26
- Hits demands Xohana + prototype: 26
- Total hits Xohana + prototype: 52
- Hit ratio Xohana + prototype: 100.00 %
- **Optimization potential: 38.46 %**

A.4 Use-case Simulation 3

- Short description of the vacation: Beach holiday, Turkish Riviera
- Hotel description at HolidayCheck.com (HolidayCheck.com, 2011d)
- Customer review at HolidayCheck.com (HolidayCheck.com, 2011i)

Table A.3: Use-case Simulation 3

Articulated requirements, total: 27	**C X**	**C P**	**D X**	**D P**	**C S**	**D S**
hotel facilities should be all inclusive	1	0	1	1	-	"all"
season should be low/off season	1	0	0	1	-	-
hotel facilities should be shopping/shops	1	0	0	1	-	"sho"
distance to beach should be less than 500 m	1	0	1	0	-	-
beach should be disco	0	1	1	1	"bea"	"dis"
beach should be beach bar	0	1	0	1	-	-
sports should be volleyball	1	0	0	1	-	"vol"
hotel facilities should be animation	1	0	0	1	-	"ani"
staff should be various/different languages	0	1	0	1	-	"lan"
food should be big variety/choice	1	1	0	1	-	"var"
hotel facilities should be fitness trainer	1	1	0	0	-	"gym"
beach should be clean	0	1	0	1	-	-
hotel facilities should be water-park	1	0	0	0	-	"wat"
room should be spacious	0	1	0	1	-	-
room facilities should be balcony	1	0	1	0	-	-
bed should be kingsize	1	0	1	1	-	-
bathroom should be spacious	0	1	0	1	-	-
mini bar should be free/complimentary	0	1	0	1	"min"	"fre"
hotel facilities should air condition	1	0	1	0	-	-
room facilities should be TV	1	0	1	0	-	-
room facilities should be safe	1	0	1	0	-	-
hotel facilities should be room service	1	0	1	0	-	-
sports should be tennis	1	0	1	0	-	-
hotel facilities should be swimming pool	1	0	1	1	-	-
hotel facilities should be massage	1	0	1	1	-	"mas"
hotel facilities should be spa centre/service	1	0	0	1	-	"spa"
hotel facilities should be sauna	1	0	1	0	-	-
Number of hits	**20**	**9**	**13**	**17**		

Based on the hits the following values can be calculated:

- Possible hits (criteria + demands): 54
- Hits criteria Xohana: 20
- Hits demands Xohana: 13
- Total hits Xohana: 33
- Hit ratio Xohana: 61.11 %
- Hits criteria Xohana + prototype: 27
- Hits demands Xohana + prototype: 25
- Total hits Xohana + prototype: 52
- Hit ratio Xohana + prototype: 96.30 %
- **Optimization potential: 35.19 %**

A.5 Use-case Simulation 4

- Short description of the vacation: Active-Hotel, Mountain-Biking Tours, Torbole, Italy
- Hotel description at HolidayCheck.com (HolidayCheck.com, 2011a)
- Customer review at HolidayCheck.com (HolidayCheck.com, 2011f)

Table A.4: Use-case Simulation 4

Articulated requirements, total: 27	**C X**	**C P**	**D X**	**D P**	**C S**	**D S**
sports should be mountain biking	1	0	1	1	-	"mou"
hotel facilities should be bike service	1	0	0	0	-	"bik"
location should be central	1	1	1	1	-	-
landscape should be mountains	1	0	1	0	-	
food should be veried (fresh) breakfast	1	0	1	1	-	"bre"
distance to town should be less than 1 km	1	0	1	0	-	-
distance to grocery should be less than 1 km	1	0	1	0	-	-
staff should be different/several languages	0	1	0	1	-	"lan"
staff should be friendly helpful	0	1	0	1	-	-
hotel facilities should be bike hire	1	0	1	0	-	-
hotel facilities should be laundry	1	0	1	0	-	-
activities should be guided tours	1	0	0	1	-	"gui"
food should be snacks	1	1	1	1	-	"sna"
hotel facilities should be bike room	1	0	0	0	-	"bik"
hotel facilities should be swimming pool	1	1	1	0	-	-
sports and activities should be excursions	1	0	0	1	-	"exc"
nights should be quiet	0	1	0	1	-	-
bathroom should be spacious	0	1	0	1	-	-
room facilities should be balcony	1	0	1	0	-	-
room facilities should be air condition	1	0	1	0	-	-
room facilities should be flat screen tv	1	0	0	1	-	"fla"
room type should be non-smoking room	1	0	1	1	-	"non"
hotel facilities should room service	1	0	1	0	-	-
sports should be hiking	1	0	0	0	-	"hik"
hotel facilities shold be free parking	1	0	0	0	-	"fre"
room facilities should be safe	1	0	1	0	-	-
hotel facilities should be whirlpool	1	1	1	0	-	-
Number of hits	**23**	**8**	**16**	**12**		

Based on the hits the following values can be calculated:

- Possible hits (criteria + demands): 54
- Hits criteria Xohana: 23
- Hits demands Xohana: 16
- Total hits Xohana: 39
- Hit ratio Xohana: 72.22 %
- Hits criteria Xohana + prototype: 27
- Hits demands Xohana + prototype: 23
- Total hits Xohana + prototype: 50
- Hit ratio Xohana + prototype: 92.59 %
- **Optimization potential: 20.37 %**

A.6 Use-case Simulation 5

- Short description of the vacation: Family Vacation in Feld am See, Austria
- Hotel description at HolidayCheck.com (HolidayCheck.com, 2011b)
- Customer review at HolidayCheck.com (HolidayCheck.com, 2011g)

Table A.5: Use-case Simulation 5

Articulated requirements, total: 25	**C X**	**C P**	**D X**	**D P**	**C S**	**D S**
holiday should be familiy trip/vacation	0	1	0	1	"hol"	"fam"
service should be outstanding	0	1	0	1	-	-
location should be lake side	1	0	1	0	-	-
facilities should be shopping/shops	1	0	1	1	-	"sho"
beach should be private	0	1	0	1	-	"bea"
hotel facilities should be airport transfer	1	0	1	0	-	-
staff should be english speaking	0	1	0	1	-	-
food should be healthy	1	1	0	1	-	"hea"
food should be children menus	1	0	0	0	-	"chi"
soft drinks should be free	0	0	0	1	"sof"	-
food should be fresh fruit	1	1	0	1	-	"fre"
room should be spacious	0	1	0	1	-	-
room facilities should be minibar	1	0	1	0	-	-
room facilities should be balcony	1	0	1	0	-	-
room location should be lake view	1	0	1	0	-	-
safety should be exceptional	0	0	0	1	"saf"	"exc"
hotel facilities should be hotel safe	1	0	1	0	-	-
hotel facilities should be parking	1	0	1	0	-	"fre"
hotel facilities should be familiy room	1	0	0	1	-	"fam"
hotel facilities should be internet	1	0	1	0	-	-
hotel facilities should be restaurant	1	0	1	0	-	-
sports should be tennis court	1	0	1	0	-	-
sports should be minigolf	1	0	0	0	-	"min"
hotel facilities should be swimming pool	1	0	1	0	-	-
hotel facilities should be child care	1	1	0	0	-	"chi"
Number of hits	**18**	**8**	**12**	**11**		

Based on the hits the following values can be calculated:

- Possible hits (criteria + demands): 50
- Hits criteria Xohana: 18
- Hits demands Xohana: 12
- Total hits Xohana: 30
- Hit ratio Xohana: 60.00 %
- Hits criteria Xohana + prototype: 23
- Hits demands Xohana + prototype: 22
- Total hits Xohana + prototype: 45
- Hit ratio Xohana + prototype: 90.00 %
- **Optimization potential: 30.00 %**

List of Figures

List of Tables

References

Abbassi, Z., Amer-Yahia, S., Lakshmanan, L. V., Vassilvitskii, S., & Yu, C. (2009). Getting recommender systems to think outside the box. In *Proceedings of the third acm conference on recommender systems* (pp. 285–288). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1639714.1639769

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*, 734–749.

Basiri, J., Shakery, A., Moshiri, B., & Zi Hayat, M. (2010). Alleviating the cold-start problem of recommender systems using a new hybrid approach. In *Telecommunications (ist), 2010 5th international symposium on* (pp. 962–967). Available from http://khorshid.ut.ac.ir/~m.zihayat/Files/IST%20v3.1.pdf

Bergholz, A. (2003). Coping with sparsity in a recommender system. In *Webkdd 2002 - miningweb data for discovering usage patterns and profiles* (Vol. 2703, pp. 86–99). Springer Berlin / Heidelberg. Available from http://dx.doi.org/10.1007/978-3-540-39663-5_6 (10.1007/978-3-540-39663-5_6)

Blanco-Fernandez, Y., Pazos-arias, J., Gil-Solla, A., Ramos-Cabrer, M., & Lopez-Nores, M. (2008, may). Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *Consumer Electronics, IEEE Transactions on*, *54*(2), 727 -735.

Brill, E. (1994). Some advances in transformation-based part of speech tagging. In *Proceedings of the twelfth national conference on artificial intelligence (vol. 1)* (pp. 722–727). Menlo Park, CA, USA: American Association for Artificial Intelligence. Available from http://portal.acm.org/citation.cfm?id=199288.199378

Burke, R. (2000). Knowledge-based Recommender Systems. In *Encyclopedia of library and information systems* (Vol. 69). Available from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.6029

Burke, R. (2007). Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web* (pp. 377–408). Berlin, Heidelberg: Springer-Verlag. Available from http://portal.acm.org/citation.cfm?id=

1768197.1768211

Cosley, D., Lawrence, S., & Pennock, D. M. (2002). Referee: an open framework for practical testing of recommender systems using researchindex. In *Proceedings of the 28th international conference on very large data bases* (pp. 35–46). VLDB Endowment. Available from http://portal.acm.org/citation.cfm?id=1287369.1287374

Devi, M. K. K., Samy, R. T., Kumar, S. V., & Venkatesh, P. (2010). Probabilistic neural network approach to alleviate sparsity and cold start problems in collaborative recommender systems. In *Computational intelligence and computing research (iccic), 2010 ieee international conference on* (pp. 1–4). Available from http://dx.doi.org/10.1109/ICCIC.2010.5705777

Felfernig, A., & Burke, R. (2008). Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on electronic commerce* (pp. 3:1–3:10). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1409540.1409544

Fürnkranz, J., & Hüllermeier, E. (2010). *Preference Learning* (1st ed.). Berlin: Springer. Hardcover.

Gao, W., Niu, C., Nie, J.-Y., Zhou, M., Hu, J., Wong, K.-F., et al. (2007). Cross-lingual query suggestion using query logs of different languages. In *Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval* (pp. 463–470). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1277741.1277821

Ghazanfar, M., & Prugel-Bennett, A. (2010, January). A scalable, accurate hybrid recommender system. In *Knowledge discovery and data mining, 2010. wkdd '10. third international conference on* (pp. 94–98).

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., et al. (1999). Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the sixteenth national conference on artificial intelligence and the eleventh innovative applications of artificial intelligence conference innovative applications of artificial intelligence* (pp. 439–446). Menlo Park, CA, USA: American Association for Artificial Intelligence. Available from http://portal.acm.org/citation.cfm?id=315149.315352

He, J., & Chu, W. W. (2009). *A Social Network-Based Recommender Sys-*

tem (SNRS) (Tech. Rep.). Computer Science Department, UCLA. Available from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.156.2547

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004, January). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, *22*, 5–53. Available from http://doi.acm.org/10.1145/963770.963772

HolidayCheck.com. (2011a). *Catalog information about aktivhotel santa lucia in torbole (italy).* http://www.holidaycheck.com/hotel-travel+information_Aktivhotel+Santa+Lucia-hid_156219.html. (last access 04/2011)

HolidayCheck.com. (2011b). *Catalog information about hotel brennseehof in feld am see (austria).* http://www.holidaycheck.com/hotel-travel+information_Hotel+Brennseehof-hid_92665.html. (last access 04/2011)

HolidayCheck.com. (2011c). *Catalog information about hotel chong fah beach resort in bang niang beach (thailand.* http://www.holidaycheck.com/hotel-travel+information_Hotel+Chong+Fah+Beach+Resort-hid_124614.html. (last access 04/2011)

HolidayCheck.com. (2011d). *Catalog information about hotel royal dragon in evrenseki (turkey).* http://www.holidaycheck.com/hotel-travel+information_Hotel+Royal+Dragon-hid_158494.html. (last access 04/2011)

HolidayCheck.com. (2011e). *Catalog information about mara bush camp in masai mara (kenya).* http://www.holidaycheck.com/hotel-travel+information_Mara+Bush+Camp-hid_174453.html. (last access 04/2011)

HolidayCheck.com. (2011f). *Customer review about aktivhotel santa lucia in torbole (italy).* http://www.holidaycheck.com/detail-reviews_Aktivhotel+Santa+Lucia+Hotel+perfect+for+mountain+biking-ch_hb-id_1882435.html. (last access 04/2011)

HolidayCheck.com. (2011g). *Customer review about hotel brennseehof in feld am see (austria).* http://www.holidaycheck.com/detail-reviews_Hotel+Brennseehof+A+dream+holiday+for+the+whole+family-ch_hb-id_1976665.html. (last access 04/2011)

HolidayCheck.com. (2011h). *Customer review about hotel hotel chong fah beach resort in bang niang beach (thailand.* http://www.holidaycheck.com/

detail-reviews_Hotel+Chong+Fah+Beach+Resort+Outstanding-ch_hb-id_1565275.html. (last access 04/2011)

HolidayCheck.com. (2011i). *Customer review about hotel royal dragon in evrenseki (turkey), khao lak / phang nga.* http://www.holidaycheck.com/detail-reviews_Hotel+Royal+Dragon+The+best+holiday+I+have+had+yet-ch_hb-id_2349474.html. (last access 04/2011)

HolidayCheck.com. (2011j). *Customer review about mara bush camp in masai mara (kenya.* http://www.holidaycheck.com/detail-reviews_Mara+Bush+Camp+Choose+no+other+this+is+exceptional-ch_hb-id_1720134.html. (last access 04/2011)

Huang, C., & Yin, J. (2010). Effective association clusters filtering to cold-start recommendations. In *Fuzzy systems and knowledge discovery (fskd), 2010 seventh international conference on* (Vol. 5, pp. 2461–2464).

Iaquinta, L., Gemmis, M. de, Lops, P., Semeraro, G., Filannino, M., & Molino, P. (2008). Introducing serendipity in a content-based recommender system. In *Hybrid intelligent systems, 2008. his '08. eighth international conference on* (pp. 168–173).

Jackson, P., & Moulinier, I. (2002). *Natural language processing for online applications: text retrieval, extraction, and categorization.* Amsterdam: John Benjamins Publishing Company.

Jia-li, D., & Ping-fang, Y. (2010). Towards natural language processing: A well-formed substring table approach to understanding garden path sentence. In *Database technology and applications (dbta), 2010 2nd international workshop on* (pp. 1–5).

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition second edition* (2nd ed.). London: Pearson Education.

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on information and knowledge management* (pp. 247–254). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/502585.502627

Kumar, D., & Josan, G. S. (2010, September). Article:part of speech taggers for morphologically rich indian languages: A survey. *International Journal of*

Computer Applications, *6*(5), 1–9. (Published By Foundation of Computer Science)

Li, Z. (2010). Product feature extraction with a combined approach. In *Intelligent information technology and security informatics (iitsi), 2010 third international symposium on* (pp. 686–690).

Loh, S., Lorenzi, F., Saldana, R., & Licthnow, D. (2004). A tourism recommender system based on collaboration and text analysis. *Information Technology and Tourism*, *6*.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993, June). Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, *19*, 313–330. Available from `http://portal.acm.org/citation.cfm?id=972470.972475`

Martinez, L., Perez, L., & Barranco, M. (2009). Incomplete preference relations to smooth out the cold-start in collaborative recommender systems. In *Fuzzy information processing society, 2009. nafips 2009. annual meeting of the north american* (pp. 1–6).

McCallum, A. (2005, November). Information extraction: Distilling structured data from unstructured text. *Queue*, *3*, 48–57. Available from `http://doi.acm.org/10.1145/1105664.1105679`

Middleton, S. E., Alani, H., & Roure, D. D. (2002). Exploiting synergy between ontologies and recommender systems. *CoRR*, *cs.LG/0204012*.

Middleton, S. E., De Roure, D. C., & Shadbolt, N. R. (2001). Capturing knowledge of user preferences: ontologies in recommender systems. In *Proceedings of the 1st international conference on knowledge capture* (pp. 100–107). New York, NY, USA: ACM. Available from `http://doi.acm.org/10.1145/500737.500755`

Middleton, S. E., Shadbolt, N. R., & De Roure, D. C. (2004, January). Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, *22*, 54–88. Available from `http://doi.acm.org/10.1145/963770.963773`

Mitkov, R. (2003). *The oxford handbook of computational linguistics (oxford handbooks in linguistics s.).* Oxford: Oxford University Press.

Moens, M.-F. (2006). *Information Extraction: Algorithms and Prospects in a Retrieval Context (The Information Retrieval Series)* (1st ed.). Dordrecht:

Springer.

Mohammad, S., & Pedersen, T. (2003). Guaranteed pre-tagging for the brill tagger. In A. Gelbukh (Ed.), *Computational linguistics and intelligent text processing* (Vol. 2588, pp. 117–172). Springer Berlin / Heidelberg. Available from `http://dx.doi.org/10.1007/3-540-36456-0_15` (10.1007/3-540-36456-0_15)

Nageswara Rao, K., & Talwar, V. G. (2008). Application domain and functional classification of recommender systems a survey. *Desidoc journal of library and information technology*, *28*(3), 17–36.

Navarro, G. (2001, March). A guided tour to approximate string matching. *ACM Comput. Surv.*, *33*, 31–88. Available from `http://doi.acm.org/10.1145/375360.375365`

Navarro, G., & Raffinot, M. (2002). *Flexible pattern matching in strings: Practical on-line search algorithms for texts and biological sequences.* Cambridge: Cambridge University Press.

Pan, P.-Y., Wang, C.-H., Horng, G.-J., & Cheng, S.-T. (2010). The development of an ontology-based adaptive personalized recommender system. In *Electronics and information engineering (iceie), 2010 international conference on* (Vol. 1, pp. V1-76–V1-80).

Papagelis, M., Rousidis, I., Plexousakis, D., & Theoharopoulos, E. (2005). Incremental collaborative filtering for highly-scalable recommendation algorithms. In *Foundations of intelligent systems* (pp. 553–561). Available from `http://dx.doi.org/10.1007/11425274_57`

Poirier, D., Fessant, F., & Tellier, I. (2010, 31). Reducing the cold-start problem in content recommendation through opinion classification. In *Web intelligence and intelligent agent technology (wi-iat), 2010 ieee/wic/acm international conference on.*

Rafiei, D., & Li, H. (2009). Data extraction from the web using wild card queries. In *Proceeding of the 18th acm conference on information and knowledge management* (pp. 1939–1942). New York, NY, USA: ACM. Available from `http://doi.acm.org/10.1145/1645953.1646270`

Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., et al. (2002). Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on intelligent user* *interfaces* (pp. 127–134). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/502716.502737

Resnick, P., & Varian, H. R. (1997, March). Recommender systems. *Commun. ACM*, *40*, 56–58. Available from http://doi.acm.org/10.1145/245108.245121

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2010). *Recommender Systems Handbook* (1st ed.). New York: Springer. Hardcover.

Rollett, H. (2008). *Xohana company profile.* http://www.unternehmerwerden.at/images/stories/profile/25folder-xohana.pdf. (last access 03/2011)

Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/371920.372071

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web* (pp. 291–324). Berlin, Heidelberg: Springer-Verlag. Available from http://portal.acm.org/citation.cfm?id=1768197.1768208

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001, January). E-commerce recommendation applications. *Data Min. Knowl. Discov.*, *5*, 115–153. Available from http://portal.acm.org/citation.cfm?id=593429.593510

Semantic Web Company. (2009). *Herwig rollett: "xohana helps people better articulate what they really want.".* http://www.semantic-web.at/1.36.resource.288.herwig-rollet-x22-xohana-helps-people-better-articulate-what-they-really-want-x22.htm. (last access 03/2011)

Shahabi, C., Banaei-Kashani, F., Chen, Y.-S., & McLeod, D. (2001, September 3). Yoda: An accurate and scalable web-based recommendation system. In C. Batini, F. Giunchiglia, P. Giorgini, & M. Mecella (Eds.), *Cooperative information systems* (Vol. 2172, pp. 418–432). Springer Berlin / Heidelberg. Available from http://dx.doi.org/10.1007/3-540-44751-2_31

Shih, Y.-Y., & Liu, D.-R. (2005). Hybrid recommendation approaches: Collaborative filtering via valuable content information. In *System sciences, 2005. hicss '05. proceedings of the 38th annual hawaii international conference on* (p. 217b).

Somprasertsri, G., & Lalitrojwong, P. (2008). Automatic product feature extraction from online product reviews using maximum entropy with lexical and syntactic features. In *Information reuse and integration, 2008. iri 2008. ieee international conference on* (pp. 250–255).

Su, X., & Khoshgoftaar, T. M. (2009, January). A survey of collaborative filtering techniques. *Adv. in Artif. Intell., 2009*, 4:2–4:2. Available from `http://dx.doi.org/10.1155/2009/421425`

TripAdvisor.com. (2011a). *Customer review example 1, tripadvisor.com.* `http://www.tripadvisor.com/ShowUserReviews-g147351-d151157-r99911897-Hotel_L_Esplanade-Grand_Case_St_Maarten_St_Martin.html#CHECK_RATES_CONT`. (last access 03/2011)

TripAdvisor.com. (2011b). *Customer review example 2, tripadvisor.com.* `http://www.tripadvisor.de/ShowUserReviews-g297555-d301880-r8205978-Four_Seasons_Resort_Sharm_El_Sheikh-Sharm_El_Sheikh_South_Sinai_Red_Sea_and_Sinai.html#CHECK_RATES_CONT`. (last access 04/2011)

Wu, Y., Wu, X., Min, F., & Li, Y. (2010). A nettree for pattern matching with flexible wildcard constraints. In *Information reuse and integration (iri), 2010 ieee international conference on* (pp. 109–114).

Xohana e.U. (2011). *Xohana website.* `http://www.xohana.com`. (last access 03/2011)

Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., et al. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international acm sigir conference on research and development in information retrieval* (pp. 114–121). New York, NY, USA: ACM. Available from `http://doi.acm.org/10.1145/1076034.1076056`

Yu, K., Wen, Z., Xu, X., & Ester, M. (2001). Feature weighting and instance selection for collaborative filtering. In *Database and expert systems applications, 2001. proceedings. 12th international workshop on.*

Zhan, T., & Li, C. (2010, 31). Product feature mining with nominal semantic structure. In *Web intelligence and intelligent agent technology (wi-iat), 2010 ieee/wic/acm international conference on.*