

Master's Thesis

# On Knowledge Discovery and Content Analytics from Unstructured Biomedical Textual Data Sets

Christof Stocker

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



Supervisor: Assoc. Prof. Dr. Andreas HOLZINGER, PhD, MSc, MPh, BEng, CEng,  
DipEd, MBCS

Graz, October 14, 2013



Masterarbeit

(Diese Arbeit ist in englischer Sprache verfasst)

# Über Wissensentdeckung und Inhaltsanalyse von unstrukturierten biomedizinischen textbasierten Datensätzen

Christof Stocker

Institut für Informationssysteme und Computer Medien,  
Technische Universität Graz



Betreuer: Assoc. Prof. Dr. Andreas HOLZINGER, PhD, MSc, MPh, BEng, CEng,  
DipEd, MBCS

Graz, 14. Oktober 2013



# Abstract

Electronic patient files contain increasingly large portions of data which has been entered in non-standardized format, often referred to as *free text*. Especially in the German speaking countries, text seems to be the preferred way of documentation and communication when it comes to patient care. Interestingly, a lot of those electronic patient files are in the PDF (or a similar) format. Even though most clinics utilize a database in the background, which does contain some structured information (e.g. patient names, IDs, and addresses), the medical information (i.e. diagnosis, therapies, ...) is still encoded in electronic text.

The idea of this work was to create a prove of concept that tailored solutions to specific biomedical domains can be realized in a realistic time frame and with sufficient quality. Ultimately, our ongoing goal is to improve the efficiency and quality of the patient's care, by helping the medical professional navigate and process the huge amount of electronically stored information at his or her disposal.

To test this idea, we applied information extraction (IE) techniques to electronic, dermatology-specific out-patient cards in order to associate patients with extracted information such as *diagnosis, therapies, medicaments, tumor size*, and so forth. The extracted information are then used to not only navigate the data, but to create chronological sequences to enable the medical professional to see trends and correlations.

The backbone of this project was provided by IBM Content Analytics (ICA), an UIMA-based framework that offers a modular component architecture with natural language processing (NLP) functionality, an *Eclipse*-based development suite, and an end-user interface that is aligned with our vision of including the human in the loop.

In order to evaluate our hypothesis, we calculated the precision and recall of the key annotations. We concluded, that because of the countless possible solutions to an annotation-problem (where most of them are imprecise), the quality of the annotations strongly depend on the skills of the developer formulating them. However, our results and personal experience suggest that ICA is able to provide the necessary flexibility and modularity to efficiently tailor solutions for the purpose of knowledge discovery from unstructured biomedical textual data sets.

## Keywords

text mining, information extraction, natural language processing, medical text, content analytics

## Ö-STAT classification

102020, 102015

## ACM classification

Information Systems



## **Kurzfassung**

Arztbriefe sind eine wertvolle Ressource um die Kontinuität der Behandlung von Patienten sicherzustellen. So geben sie, unter anderen, einen Überblick über den Status des Patienten bei seiner Entlassung, über den Verlauf seiner Erkrankung(en), und auch den veranlassten Therapien.

Eine Kollektion solcher Arztbriefe hat aber weit mehr - heute noch weitgehend ungenutztes - Potential. Nehmen wir als Beispiel die Behandlung von Patienten. Bereits bekannte Fälle mit ähnlichen Verlauf sind hier von großem Interesse für den Mediziner, um ihn mit Vergleichswerten bei seinen Entscheidungen zu unterstützen. Auf der anderen Seite sind auch statistische Informationen von großem Interesse, wie zum Beispiel der Zusammenhang zwischen Therapien und Krankheitsverläufen.

Obwohl die meisten Kliniken sich mittlerweile auf eine elektronische Datenbank stützen, sind ein Großteil der medizinisch relevanten Informationen noch immer in unstrukturierter Form (z.B. elektronischer Text, Bilder) gespeichert. Um maschinell an die im Fließtext umschriebenen, medizinisch relevanten Informationen heranzukommen, müssen Algorithmen aus dem Natural Language Processing (NLP) Bereich eingesetzt werden. Die Entwicklung, und der Einsatz effizienter und präziser Algorithmen weisen jedoch auch so einige Hürden auf.

Wissenschaftliches Ziel dieses Projekts war die experimentelle Erprobung des von IBM zur Verfügung gestellten Softwarepakets IBM Content Analytics (ICA) (einem UIMA-basierten Frameworks für die Extraktion, Navigation, und Interpretation von Informationen in Fließtexten) für den Einsatz im medizinischen Bereich.

Unsere Hypothese, auf die wir uns stützten, war dass zugeschnittene Lösungen für bestimmte medizinische Domänen in realistischer Zeit und mit möglichst wenigen Ressourcen realisierbar sind. Um diese Hypothese zu testen, haben wir Information Extraction (IE) Techniken auf Ambulanzkarten der Dermatologie angewandt, um dadurch Informationen wie Diagnosen, Therapien, Medikamente, Tumordicke, usw. zu extrahieren.

Für die Evaluierung haben wir für die Schlüssel-Annotationen sowohl Präzision als auch Recall ermittelt. Aus den Ergebnissen und unseren persönlichen Erfahrungen schließen wir, dass ICA die notwendige Flexibilität und Modularität aufweist, um effiziente Lösungen für das Ziel von Wissensentdeckung aus biomedizinischen text-basierten Datenquellen zu realisieren.

### **Schlüsselwörter**

Content Analyse, Informationsverarbeitung, Textmining, NLP

### **Ö-STAT Klassifikation**

102020, 102015

### **ACM Klassifikation**

Information Systems





## 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, October 14, 2013

---

Christof Stocker

*“The scientist only imposes two things, namely truth and sincerity, imposes them upon himself and upon other scientists.”*

– Erwin Schrödinger, *Physicist*

*“Knowledge has to be improved, challenged, and increased constantly, or it vanishes.”*

– Peter Drucker, *Management Consultant*

*“Never give up work. Work gives you meaning and purpose and life is empty without it.”*

– Steven Hawking, *Theoretical Physicist*

*“I find that the harder I work, the more luck I seem to have.”*

– Thomas Jefferson, *President of the United States*

*“Whether or not you can never become great at something, you can always become better at it.”*

– Neil deGrasse Tyson, *Astrophysicist*

## Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Holzinger for his continuous support, for always finding the time to answer my comprehensive questions, and for his patience in our seemingly countless discussions. I know that my bold curiosity may sometimes be exhausting, and so I - amongst many others I'm sure - appreciate his dedication to teaching and science.

I would also like to thank my colleague Bernhard Ofner, who worked with me on this project.

Christof Stocker  
Graz, October 14, 2013



# Table of Contents

<b>1</b>	<b>Introduction and Motivation for Research</b>	<b>15</b>
1.1	Storing, Retrieving and Communicating Information . . . . .	15
1.2	The Challenges of Text . . . . .	16
1.3	On the Importance of Text in the Biomedical Domain . . . . .	16
1.4	On the Importance of Electronic Patient Files as Information Source	17
1.5	The Intention of this Work . . . . .	18
1.6	The Organization of this Work . . . . .	19
<b>2</b>	<b>Theoretical Background</b>	<b>21</b>
2.1	Mathematical Notions and Terminology . . . . .	21
2.1.1	Basic Terms and Definitions . . . . .	21
2.1.2	Formal Language Theory . . . . .	23
2.2	Logic and Reasoning . . . . .	26
2.2.1	Boolean Logic . . . . .	26
2.2.2	First-Order Logic . . . . .	28
2.3	Data, Information, and Knowledge . . . . .	36
2.3.1	Unstructured Information . . . . .	37
2.3.2	Knowledge Discovery . . . . .	37
2.3.3	Human-Computer Interaction (HCI) . . . . .	38
2.3.4	HCI-KDD . . . . .	39
2.4	Computational Linguistics (CL) . . . . .	39
2.4.1	Morphology . . . . .	39
2.4.2	Lexicography . . . . .	40
2.4.3	Finite-State Technology . . . . .	40
2.4.4	Text Segmentation . . . . .	40
2.4.5	Part-of-Speech Tagging . . . . .	40
2.4.6	Information Extraction . . . . .	40
2.5	Unstructured Information Management Architecture (UIMA) . . . . .	41
2.5.1	History and Motivation . . . . .	41
2.5.2	Specification . . . . .	42

<b>3</b>	<b>Related Work</b>	<b>45</b>
<b>4</b>	<b>Materials</b>	<b>47</b>
4.1	Electronic Patient Files . . . . .	47
4.2	Apache UIMA Framework . . . . .	48
4.3	IBM Content Analytics with Enterprise Search (ICAwES) . . . . .	50
4.3.1	Terms and Definitions . . . . .	50
4.3.2	Architecture . . . . .	50
4.4	IBM Content Analytics Studio . . . . .	53
<b>5</b>	<b>Methods</b>	<b>55</b>
5.1	Information of Interest to the Physician . . . . .	55
5.2	Naming Convention . . . . .	55
5.3	Designing an Annotation Pipeline . . . . .	56
5.4	Custom Normalizer . . . . .	56
5.4.1	FormulaSolver . . . . .	56
5.4.2	DistanceNormalizer . . . . .	58
5.4.3	DateDifferenceCalculator . . . . .	58
5.5	Extract Basic Annotations to Build on . . . . .	58
5.5.1	Annotation for (Reel) Numbers . . . . .	59
5.5.2	Annotation for Length Measurements . . . . .	60
5.5.3	Annotation for Enumerations . . . . .	60
5.5.4	Annotation for Dates . . . . .	61
5.6	Extract Semi-Structured Information . . . . .	62
5.7	Extract Medical Information . . . . .	63
5.7.1	Annotation for Breslow’s depth . . . . .	63
5.7.2	Annotation for the Stadium of a Malign Melanoma . . . . .	64
5.7.3	Annotation for Location Information . . . . .	64
5.7.4	Annotation for the Location of a Malign Melanoma . . . . .	65
5.7.5	Annotation for Diagnostic Methods . . . . .	65
5.7.6	Annotation for Therapeutic Methods . . . . .	66
5.7.7	Annotation for Symptoms and Medication . . . . .	66
5.7.8	Annotation for Diagnosis . . . . .	66
<b>6</b>	<b>Results</b>	<b>67</b>
6.1	User Interface . . . . .	67
6.2	Annotation Quality . . . . .	73
<b>7</b>	<b>Discussion and Lessons Learned</b>	<b>75</b>

<b>8</b>	<b>Conclusions</b>	<b>77</b>
<b>9</b>	<b>Future Work</b>	<b>79</b>
	<b>List of Figures</b>	<b>81</b>
	<b>List of Tables</b>	<b>83</b>
	<b>References</b>	<b>85</b>





# 1. Introduction and Motivation for Research

The topic of this thesis is the computational extraction of information out of biomedical textual data. This section will serve as a brief introduction and overview of the associated problem domain of this topic. We will discuss why text - and in particular biomedical text - is such an interesting branch of data, and why solutions to the associated problems of information extraction are desirable. Note that parts of this work has been presented during the *SouthCHI* conference at the University of Maribor (Holzinger, Stocker, Ofner, Prohaska, Brabenetz and Hofmann-Wellenhof, 2013).

## 1.1 Storing, Retrieving and Communicating Information

Communication is a vital part of our every day life. Without communication and memorization, there can be no progress. A very prominent medium of both is text. Beside speech, digital text can nowadays be considered as one of the most natural ways of communication. This is probably due to the prominence of handwriting, as its origin and tradition goes back to the early cave painters (Balter, 2000).

But what is communication without memorization? Today, memory is not an issue. Storing information feels as simple as it is cheap. However, retrieving information is not as easy, since most of the information is currently treated as *unstructured*. This ambiguous term is often used for "*everything that is not in a formal database*". Since the 90s, and partially because of this seemingly boundless collection of "unstructured" information in the form of something referred to as "unstructured data", a large portion of statistic analysis shifted from data analysis to data mining (Tufféry, 2011).

But what do the terms *data*, *information* and *knowledge* really mean? Definitions throughout literature vary greatly (Zins, 2007). Recently, the definitions of the terms were revisited for the context of natural language (Holzinger, Stocker, Ofner, Prohaska, Brabenetz and Hofmann-Wellenhof, 2013), as it usually just means that the structure is un-modeled, difficult to model, or might be - in a way - self-describing. Implicitly though, this hidden structure was already treated as such by researchers all over the world and knowledge discovery methods were employed to statistically approximate parts of this un-modeled structure. Be it by exploiting domain knowledge, or without any.

## 1.2 The Challenges of Text

A huge portion of this digital “unstructured” data is text (Stuckenschmidt and van Harmelen, 2005). Naturally, information extraction and knowledge discovery out of text is a hot scientific topic, since useful solutions would open a lot of business opportunities.

Text, seen as transcription of natural language, poses a lot of challenges for computational analysis. Natural language *understanding* is regarded as an AI-complete problem (Waldrop, 1984). In analogy to NP-completeness from complexity theory this means that the difficulty of the computational problem is equivalent to designing a computer which is as intelligent as a human being (Weizenbaum, 1966), and which brings us back to the very roots of the computational sciences (Turing, 1950).

It became evident over the past decades that the understanding of human language requires extensive knowledge, not only about the language itself, but also about the surrounding real world, because language is more than words, and meaning depends on context, and “understanding” requires a vast body of knowledge about this real world context (Waldrop, 1984) - we call this context-awareness (Yndurain et al., 2012). Consequently, natural language processing (NLP) is a term that does not necessarily target a total understanding of language per se (Erhardt et al., 2006).

However, even the extraction of information out of text is a challenging task. The underlying structure of text is fairly complex and not easily understood by computers. In return, the approaches to conquer the medium text are either quite extensive, or specialized for the task at hand. In many applications, most of the textual structure gets discarded and the structural focus depends on the context. Furthermore, and beyond modeling difficulties, one has to deal with incorrect and incomplete data, as the syntactical correctness of text can not be trusted in general.

## 1.3 On the Importance of Text in the Biomedical Domain

Whereas some popular futurists deny the importance of text in the future, most professionals regard text as fundamental for knowledge discovery in life science: in basic research in Bioinformatics as well as in practical clinical research (e.g. Omics). Hence, if we seriously do research in medical knowledge discovery, we simply cannot ignore textual information; although we are aware of the huge difficulties of processing text with our classical Von-Neumann machines. We do not have evidence, whether text is a natural representation of human language, following evolution in the sense of Darwin (Pinker and Bloom, 1990; Nowak et al., 2002) or if text is an abstract, artificial and purely symbolic product of mankind (Hauser et al., 2002).

An interesting but difficult subcategory of biomedical text are in the form of electronic patient records.

## 1.4 On the Importance of Electronic Patient Files as Information Source

Electronic patient files contain increasingly large portions of data which has been entered in non-standardized format, which is often and not quite correctly called *free text* (Holzinger et al., 2008; Kreuzthaler et al., 2011). Interestingly, a lot of those electronic patient files are in the PDF (or a similar) format. Even though most clinics utilize a database in the background, which does contain some structured information (such as the patient names, addresses, and so forth), the medical information (i.e. diagnosis, therapies, ...) is still encoded in electronic text.

Consequently, the introduction of a nationwide or international *Electronic Health Record* (EHR) in the DACH<sup>1</sup> area could prove very beneficial from a computational point of view. An EHR is a systematic digital collection of health information about individual patients or populations (Gunter and Terry, 2005). All three DACH nations have their own approach towards an such a collection (Krüger-Brand, 2011). Austria, for example, introduced the so-called *ELGA* law in 2013. ELGA stands for the German term “*Elektronische Gesundheitsakte*” and is Austria’s version of an EHR (Krüger-Brand, 2010). Its realization will be obligatory for medical professionals starting 2015. Note that ELGA itself has absolutely nothing to do with computational analysis of those medical files, it will however eventually enforce a standard file format that might be beneficial to that cause.

ELGA will slowly introduce a standard format for electronic patient records called *Clinical Document Architecture* (CDA) (Dolin et al., 2006). CDA is XML based and supports the SNOMED (Cornet and de Keizer, 2008) standard for clinical terms. This, in turn, could result in much-needed standardized structure for the medical information in the patient files. However, CDA supports three different levels of “structure”, with the lowest essentially resulting in free text again. This could mean, that the actual level of structure in the information might still depend on the acceptance in the medical community and could go either way; free text, or fully structured information. Given the additional constraint of the rather passive step-by-step introduction of CDA into actual practice, it seems that in the near future, at least, NLP might be the only realistic approach to computationally access the medical information in this data.

Consequently for many years, text mining was and is an essential area of medical informatics, where researchers worked on statistical and linguistic procedures in order to dig out (mine) information from plain text, with the primary aim of gaining information from data. Although text can easily be *created* by medical professionals, the support of (semi-) automatic analyses is extremely difficult and has challenged researchers for many years (Gregory et al., 1995; Holzinger et al., 2000; Lovis et al., 2000). The next big challenge is in Knowledge Discovery from this data. Contrary to the classical text mining, or information retrieval approach, where the goal is to find information, hence the medical professional knows what he wants, in

---

<sup>1</sup>DACH is a German abbreviation for the German speaking nations *Germany, Austria, and Switzerland*

knowledge discovery we want to discover novel insights, get new knowledge which was previously unknown. To reach this goal, approaches from pure computer science alone are insufficient, due to the fact that the “real” intelligence is in the brains of the professionals; the next step consists of making the information both usable and useful. Interaction, communication and sensemaking are still missing within the pure computational approaches (Blandford and Attfield, 2010).

## 1.5 The Intention of this Work

The idea of this work is **not** the development of the most generic solution to medical text processing, but a prove of concept that tailored solutions to specific domains can be realized in a realistic time frame and with sufficient quality. Ultimately, our ongoing goal is to improve the efficiency and quality of patient’s care, by helping the medical professional navigate and process the huge amount of electronically stored information at his or her disposal.

The big challenges - when confronted with text written by (German speaking) medical professionals - are (a) the weak grammatical structure, (b) the mixture of languages, and (c) the large portion of (non-standardized) abbreviations.

The following excerpt highlights typical examples of abbreviation-heavy text:

St.p.	MM,	TD	1,5 mm,	OS	li.	innen,	11/2010
St.p.	NE	und sentinel node Biopsie ing links (tumorfrei) 01/2011					

- ... Abbreviation for “*status post*” (state that follows an intervention)
- ... Abbreviations for medical terms, such as *malign melanoma*
- ... Abbreviations for location and direction specification

Furthermore, medical professionals from the German speaking nations tend to use both German, as well as Latin words for medical terms. This means that in order to understand the text, the “reader” has to (at least partially) have knowledge of the vocabulary of both languages.

The following string is a typical example for Latin terms within medical text:

Fil hep. kut, intraabdom und iliaca1 (ED 11/2012)
---

To test our idea, we applied IE techniques to electronic, dermatology-specific out-patient cards in order to associate patients with extracted information such as *diagnosis, therapies, medicaments, tumor size*, and so fourth. The extracted information are then used to not only navigate the data, but to create chronological sequences to enable the medical professional to see trends and correlations; or in other words: apply *knowledge discovery*.

The backbone of this project was provided by IBM Content Analytics (ICA), as it offers a stable framework, basic NLP functionality, and a user interface that is aligned with our vision of including the human in the loop.

## 1.6 The Organization of this Work

The rest of this work is organized as follows:

We will start in Sec. 2 with the basic terms, definitions, and background necessary to understand this work; reader familiar with the theory of computation and computational linguistics should have no problem by skipping that section.

In Sec. 3 we will discuss how other scientists approached the topic of knowledge discovery in medical text.

The sections 4 and 5 will describe the underlying data sets and tool, as well as how we utilized those to reach our goals.

We will then present you with our results in Sec. 6 and end with a discussion (Sec. 7), our conclusion (Sec. 8), and an outlook to our future work (Sec. 9).



## 2. Theoretical Background

This chapter will provide a clear description of the terms and definitions needed to understand the content of this document. Note that, since this is not a textbook, most proofs for already established propositions, theorems and corollaries will be omitted. However, I will explicitly cite to sources that do list the corresponding proofs.

### 2.1 Mathematical Notions and Terminology

We will start by laying down the mathematical foundation by introducing *sets*, *sequences*, *functions* and *relations*. We will then describe the notions of *formal language theory* by defining terms such as *alphabet* and *language*.

#### 2.1.1 Basic Terms and Definitions

We begin with discussing basic mathematical concepts, that we will need to be able to understand more complicated subjects. First, let us introduce the notion of a *set*. We will base our definitions in this section on the way they are depicted in the work of Sipser (1996).

**Definition 1** (Set). *A set is a group of objects represented as a single unit. It may contain any type of object, such as numbers, symbols, and other sets.*

A set is usually denote by an upper case Latin letter, such as  $A$ , or as upper case Greek letters for special sets. The symbol  $\in$  denotes set membership while  $\notin$  denotes nonmembership of the set. We refer to the objects of a set as its *elements*. The order of the elements within a set doesn't matter, nor does repetition.

There are a couple of ways to formally describe a set. A set with no members is called the **empty set** and is denoted as  $\emptyset$ . If we want to describe the content of a set with a rule, we write  $A = \{n \mid \text{rule about } n\}$ . Another way is to simply list its elements inside curled brackets.

*Example 1.* The set  $\{n \mid n = m^2 \text{ for some } m \in \mathbb{N}\}$  describes the set of perfect squares. The set  $A = \{4, 7, 23\}$  contains the elements 4, 7, and 23. We write  $7 \in A$  since 7 is a member of  $A$ , but we write  $8 \notin A$ , because 8 is not a member of  $A$ .

For two sets  $A$  and  $B$ , we say that  $A$  is a **subset** of  $B$ , denoted as  $A \subseteq B$ , if each element  $x \in A$  is also an element of  $B$  ( $x \in B$ ). We say that  $A$  is **equal** to  $B$

if  $A$  is a subset of  $B$  and  $B$  is also a subset of  $A$ .  $A$  is called a **proper subset** of  $B$ , denoted  $A \subset B$ , if  $A$  is a subset of  $B$ , but  $A$  is not equal to  $B$ .

A special set, that we are going to use a lot after introducing logic, is the set of truth values.

**Definition 2** (Truth values). *There are two truth values, “TRUE” and “FALSE”, denoting truth and falsity (Simpson, 2011).*

A **countable set** is a set, that has the same cardinality as some subset of the set of natural numbers. An **infinite set** contains infinity many objects, and renders us unable to list all its elements. To describe its contents we will often use the three dots ... notations to say “continue forever”.

*Example 2.* The infinite set of natural numbers is defined as  $\mathbb{N} = \{1, 2, 3, \dots\}$ .

If we take two sets  $A$  and  $B$ , the **union** of those, written  $A \cup B$ , is the set we gain by combining all elements of  $A$  and all elements of  $B$  into a single set. On the other hand, the **intersection** of  $A$  and  $B$ , written  $A \cap B$ , is the set that contains only those elements that are contained in both sets. In the case of  $A \cap B = \emptyset$ , we call the sets disjoint.

**Definition 3** (Sequence). *A sequence of objects is a list of those objects in a specific order.*

As we know, the order of the elements doesn't matter in a set. In a sequence, however, it does. On the other hand, while *repetition* doesn't matter in a set, it does in a sequence. We denote a sequence using round brackets. Similar to sets, sequences can be *finite* or *infinite*. Finite sequences are usually called **tuples**. A sequence with  $k$  elements is thus a  $k$ -tuple. For convenience a 2-tuple is often referred to as **pair**, and a 3-tuple as **triple**.

The next term we are going to introduce is a *function*. Functions, also commonly referred to as *mappings*, are objects that set up an input-output relationship and are central to mathematics.

**Definition 4** (Function). *A (unary) function  $f$  is an objects that maps elements of some input set  $D$ , called **domain**, to an output set  $R$ , called its **range**. We denote such a function as*

$$f : D \longrightarrow R.$$

*If  $b \in R$  is the output value that  $f$  associates with the input value  $a \in D$ , we write  $f(a) = b$ .*

If the input of a function  $f$  is a  $k$ -tuple  $(a_1, a_2, \dots, a_k)$  for  $a_i \in A_i$  ( $1 \leq i \leq k$ ), we call  $f$  a  $k$ -ary function. The  $a_i$  are then called the arguments of  $f$ . Its domain is denoted as  $A_1 \times \dots \times A_k$ .

For  $k = 2$ , we will call a function *binary*. Certain binary functions are usually written in an infix notation (i.e. instead of  $+(a_1, a_2)$  we write  $a_1 + a_2$ ).

**Definition 5** (Predicate). *A predicate is a function whose range is  $\{\text{TRUE}, \text{FALSE}\}$ . A unary predicate is called a property.*



**Definition 6** (Relation). *A  $k$ -ary relation is a predicate whose domain is a  $k$ -tuple  $A \times \dots \times A$  of one set  $A$ .*

Let  $R$  be a  $k$ -ary relation, then  $R(a_1, \dots, a_k)$  means that  $R(a_1, \dots, a_k) = \text{TRUE}$ . A 2-ary relation is called a *binary* relation. As with binary function, binary relations also commonly use the infix notation.

A special type of binary relation is the *equivalence relation* which denotes that two objects are equal in some feature.

**Definition 7** (Equivalence relation). *Let  $R$  be a binary relation. We say that  $R$  is an equivalence relation, iff*

1.  *$R$  is reflexive (i.e. for every  $x$ ,  $xRx$ ),*
2.  *$R$  is symmetric (i.e. for every  $x$  and  $y$ ,  $xRy$  implies  $yRx$ ), and*
3.  *$R$  is transitive (i.e. for every  $x$ ,  $y$ , and  $z$ ,  $xRy$  and  $yRz$  implies  $xRz$ ).*

## 2.1.2 Formal Language Theory

Next we are going to jump into formal language theory. We will base our definitions in this section on the way they are depicted in the work of Clark et al. (2010), Mitkov (2003) and Sipser (1996).

A language is defined in respect to some *alphabet* usually denoted as  $\Sigma$  or  $\Gamma$ .

**Definition 8** (Alphabet). *An alphabet is a non-empty finite string of symbols.*

*Remark 1.* Some authors (see Mitkov, 2003) also refer to an alphabet as vocabulary. In this document, however, we do not define the term vocabulary to avoid confusion later on (see also 24).

The symbols of an alphabet are also called its *letters*. A *string over an alphabet*, also called *word*, is a finite sequence of symbols from that alphabet. Given an alphabet  $\Sigma$ , the set of all strings over  $\Sigma$  is denoted as  $\Sigma^*$ . Let  $\Sigma$  be some alphabet and  $w$  be a string over  $\Sigma$ , the *length* of  $w$  is denoted as  $|w|$  and is the number of symbols in the sequence  $w$ . A string with the length 0 is referred to as an *empty string*, denoted as  $\varepsilon$ .

Let  $\Sigma$  be some alphabet and  $w_1, w_2$  be strings over  $\Sigma$ . We say that  $w_1$  is a substring of  $w_2$  if  $w_1$  appears consecutively within  $w_2$ .

This brings us to the definition of a formal language.

**Definition 9** (Formal language). *Let  $\Sigma$  be an alphabet. A formal language  $L$  over  $\Sigma$  is any subset of  $\Sigma^*$ ,  $L \subseteq \Sigma^*$ .*

Next, we are going to introduce the formalization of a generative *grammar* first proposed by Chomsky (1956), which can be used to generate languages. Furthermore, grammars allow us to classify languages into specific classes.

**Definition 10** (Formal grammar). A formal grammar is a tuple  $G = (\Gamma, \Sigma, S, P)$ , where  $\Sigma$  and  $\Gamma$  are disjoint alphabets,  $S \in \Gamma$ , and  $P$  is a set of pairs  $(w, v)$ , such that each  $w, v \in (\Gamma \cup \Sigma)^*$  and  $w$  contains at least one letter from  $\Gamma$ .

$\Gamma$  is called the **non-terminal** alphabet,  $\Sigma$  the **terminal** alphabet,  $S$  the initial letter, and  $P$  is called the set of **productions**.

*Remark 2.* In the context of formal grammars a pair  $(w, v)$  in  $P$  is usually written as  $w \rightarrow v$ .

Given a grammar  $G$  and  $w, v \in (\Gamma \cup \Sigma)^*$ , we say that a **direct derivation**, written  $w \Rightarrow_G v$ , holds iff: (1)  $u_1, u_2 \in (\Gamma \cup \Sigma)^*$  exist such that  $w = u_1 \alpha u_2$  and  $v = u_1 \beta u_2$ , as well as (2)  $\alpha \rightarrow \beta \in P$  exists.

More generally, given a grammar  $G$  and  $w, v \in (\Gamma \cup \Sigma)^*$ , we say that a **derivation**, written  $w \Rightarrow_G^* v$ , holds iff: (1) either  $w = v$ , or (2)  $z \in (\Gamma \cup \Sigma)^*$  exists such that  $w \Rightarrow_G^* z$  and  $z \Rightarrow_G^* v$ .

With this notation we can now describe the *unique* language a grammar generates

**Definition 11.** The unique language generated by a grammar  $G$  is defined by

$$L(G) = \{w \mid S \Rightarrow_G^* w \text{ with } w \in \Sigma\} \quad (2.1)$$

The interesting property of grammars is, that they can be classified into different types according to their productions. A grammar is said to be of type:

- **0 or Phrase-Structure Grammar** (RE) iff there are no restrictions on the form of the productions in  $P$ .
- **1 or Context-Sensitive Grammar** (CS) iff every production in  $P$  is of the form:

$$u_1 N u_2 \rightarrow u_1 w u_2,$$

where  $u_1, u_2, w \in (\Gamma \cup \Sigma)^*$ ,  $N \in \Gamma$ , and  $w \neq \varepsilon$  (except for the rule  $S \rightarrow \varepsilon$ , iff  $S$  does not occur on any right-hand side of a rule).

- **2 or Context-Free Grammar** (CF) iff every production in  $P$  is of the form:

$$N \rightarrow w,$$

where  $N \in \Gamma$  and  $w \in (\Gamma \cup \Sigma)^*$ .

- **3 or Regular Grammar** (REG) iff every production in  $P$  is in the form:

$$A \rightarrow wB, \text{ or}$$

$$A \rightarrow w,$$

where  $A, B \in \Gamma$  and  $w \in \Sigma^*$

A language is said to be of type  $i$ ,  $i = 0, 1, 2, 3$  if its generated by a grammar of type  $i$ . The family of type  $i$  languages is denoted as  $\mathcal{L}_i$ .

With this notation introduced, we can now state one of the most important early results in formal language theory, which is called the **Chomsky hierarchy** of languages (Chomsky, 1956):

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0 \quad (2.2)$$

One of the most important linguistic formalisms that we need are **regular expressions**. Every well-formed regular expression denotes a set of strings, or a language (Clark et al., 2010).

**Definition 12** (Regular expression). *Let  $\Sigma$  be an alphabet. Given  $\Sigma$ , the set of regular expressions over  $\Sigma$  is defined as follows (Clark et al., 2010):*

- $\emptyset$  is a regular expression;
- $\varepsilon$  is a regular expression;
- if  $a \in \Sigma$  is a letter, then  $a$  is a regular expression;
- if  $e_1$  and  $e_2$  are regular expressions, then so are  $(e_1 + e_2)$  and  $(e_1 \cdot e_2)$ ;
- if  $e$  is a regular expression, then so is  $(e)^*$ ;
- nothing else is a regular expression over  $\Sigma$ .

As the next step, we need to define a mapping, commonly referred to as **denotation**, from regular expressions to sets of strings over  $\Sigma$ .

**Definition 13** (Denotation). *Let  $e$  be a regular expression. Given  $e$ , its denotation, written as  $[[e]]$ , is a set of strings defined as follows (Clark et al., 2010):*

- $[[\emptyset]] = \{\}$ , the empty set;
- $[[\varepsilon]] = \{\varepsilon\}$ , the singleton set containing the empty string;
- if  $a \in \Sigma$  is a letter, then  $[[a]] = \{a\}$ , the singleton set containing  $a$  only;
- if  $e_1$  and  $e_2$  are two regular expressions whose denotations are  $[[e_1]]$  and  $[[e_2]]$ , respectively, then  $[[e_1 + e_2]] = [[e_1]] \cup [[e_2]]$  and  $[[e_1 \cdot e_2]] = [[e_1]] \cdot [[e_2]]$ ;
- if  $e$  is a regular expression whose denotation is  $[[e]]$  then  $[[e]^*] = [[e]]^*$

It is important to note, that the class of languages that can be expressed as the denotation of regular expressions is called the class of regular languages (Clark et al., 2010), (Mitkov, 2003).

## 2.2 Logic and Reasoning

We will base our definitions in this section on the way they are depicted in the work of Papadimitriou (1994), Simpson (2011), and Lifschitz (2009).

The reason, that logic is explained in such detail is two-fold. First of all, logic is a very common tool in knowledge discovery in databases (KDD) and should not be ignored in this work. Secondly, the definitions are important to be able to state its limitations.

### 2.2.1 Boolean Logic

Let  $X$  be a countable infinite alphabet of Boolean variables  $X = \{x_1, x_2, \dots\}$ . Such a Boolean variable can take either one of the two truth values. We can combine Boolean variables using connectives.

**Definition 14** (Boolean connectives). *The Boolean connectives consists out of two binary connectives **conjunction** ( $\wedge$ ), **disjunction** ( $\vee$ ), as well as one unary connective called **negation** ( $\neg$ ).*

**Definition 15** (Boolean expression). *A Boolean expression can be any one of the following:*

1. A Boolean variable  $x_i \in X$ ,
2. An expression of the form  $\neg\phi$ , where  $\phi$  is a Boolean expression,
3. An expression of the form  $(\phi_1 \odot \phi_2)$ , where  $\phi_1$  and  $\phi_2$  are Boolean expressions and  $\odot$  is a binary Boolean connective.

Now that we have defined the syntax of Boolean expressions, we can think about giving them meaning by stating the semantics. We start by defining the set of Boolean variables  $X(\phi)$  given a Boolean expression  $\phi$ .

**Definition 16.** *Let  $X$  be a countable infinite alphabet of Boolean variables,  $\phi$ ,  $\psi_1$  and  $\psi_2$  be Boolean expressions, and  $\odot$  be a binary Boolean connective. We define the set  $X(\phi) \subset X$  of the boolean variables appearing in  $\phi$  inductively as follows:*

1. If  $\phi$  is a Boolean variable  $x_i$ , then  $X(\phi) = \{x_i\}$ ,
2. If  $\phi = \neg\psi_1$ , then  $X(\phi) = X(\psi_1)$ ,
3. If  $\phi = (\psi_1 \odot \psi_2)$ , then  $X(\phi) = X(\psi_1) \cup X(\psi_2)$ .

**Definition 17** (Truth assignment). *A truth assignment  $T$  is a mapping from a finite set  $X'$  of Boolean variables,  $X' \subset X$  to the set of truth values. Let  $\phi$  be a Boolean expression. If  $X(\phi) \subset X'$ , we call a truth assignment appropriate to  $\phi$ .*

**Definition 18.** *Let  $\phi$ ,  $\psi_1$  and  $\psi_2$  be Boolean expressions, and  $T$  be a truth assignment appropriate to those. We say that  $T$  satisfies  $\phi$  (written  $T \models \phi$ ) for the following situations:*

1. If  $\phi$  is a Boolean variable  $x_i \in X(\phi)$  where  $T(x_i) = \text{true}$ ,
2. If  $\phi = \neg\psi_1$  where  $T \not\models \psi_1$  (meaning  $T \models \psi_1$  does not apply),
3. If  $\phi = (\psi_1 \vee \psi_2)$  where either  $T \models \psi_1$  or  $T \models \psi_2$ ,
4. If  $\phi = (\psi_1 \wedge \psi_2)$  where both  $T \models \psi_1$ , as well as  $T \models \psi_2$  hold.

**Definition 19** (Equivalence). *Let  $\phi_1, \phi_2$  be Boolean expressions. We call  $\phi_1, \phi_2$  equivalent, written  $\phi_1 \equiv \phi_2$ , if for any truth assignment  $T$  appropriate to them,  $T \models \phi_1$ , iff  $T \models \phi_2$ , or in short  $T \models (\phi_1 \Leftrightarrow \phi_2)$ .*

Two equivalent Boolean expressions can be used interchangeably, as they can be considered as different representation of the same object.

**Proposition 1.** *Let  $\phi_1, \phi_2$  and  $\phi_3$  be arbitrary Boolean expressions. Then:*

1.  $(\phi_1 \vee \phi_2) \equiv (\phi_2 \vee \phi_1)$ ,
2.  $(\phi_1 \wedge \phi_2) \equiv (\phi_2 \wedge \phi_1)$ ,
3.  $\neg\neg\phi_1 \equiv \phi_1$ ,
4.  $((\phi_1 \vee \phi_2) \vee \phi_3) \equiv (\phi_1 \vee (\phi_2 \vee \phi_3))$ ,
5.  $((\phi_1 \wedge \phi_2) \wedge \phi_3) \equiv (\phi_1 \wedge (\phi_2 \wedge \phi_3))$ ,
6.  $((\phi_1 \wedge \phi_2) \vee \phi_3) \equiv ((\phi_1 \vee \phi_3) \wedge (\phi_2 \vee \phi_3))$ ,
7.  $((\phi_1 \vee \phi_2) \wedge \phi_3) \equiv ((\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3))$ ,
8.  $\neg(\phi_1 \vee \phi_2) \equiv (\neg\phi_1 \wedge \neg\phi_2)$ ,
9.  $\neg(\phi_1 \wedge \phi_2) \equiv (\neg\phi_1 \vee \neg\phi_2)$ ,
10.  $(\phi_1 \vee \phi_1) \equiv \phi_1$ ,
11.  $(\phi_1 \wedge \phi_1) \equiv \phi_1$ .

*Proof.* Trivial. Properties (1) to (5) are immediate consequences of the connective's definitions, while (6) to (11) can easily be shown through the use of truth tables.  $\square$

**Definition 20** (Satisfiable). *Let  $\phi$  be a Boolean expression. If there exists a truth assignment  $T$  appropriate to  $\phi$  such that  $T \models \phi$ , we call  $\phi$  satisfiable.*

**Definition 21** (Valid). *Let  $\phi$  be a Boolean expression. If  $T \models \phi$  for all truth assignment  $T$  appropriate to  $\phi$ , we call  $\phi$  valid, or a tautology and denote it  $\models \phi$ .*

**Proposition 2.** *A Boolean expression is unsatisfiable, iff its negation is valid.*

*Proof.* Let  $\phi$  be an unsatisfiable Boolean expression and consider  $\neg\phi$ . Obviously for any truth assignment  $T$  appropriate to  $\phi$ ,  $T \not\models \phi$ , and thus  $T \models \neg\phi$ .  $\square$

**Definition 22** (Disjunctive normal form). *An expression is said to be in disjunctive normal form if it is of the form  $C_1 \vee \dots \vee C_m$ , where each clause  $C_i$ ,  $i = 1, \dots, m$ , is of the form  $A_1 \wedge \dots \wedge A_n$ , and each  $A_j$ ,  $j = 1, \dots, n$  is either a Boolean variable or the negation of a Boolean variable.*

**Definition 23** (Conjunctive normal form). *An expression is said to be in conjunctive normal form if it is of the form  $C_1 \wedge \dots \wedge C_m$ , where each clause  $C_i$ ,  $i = 1, \dots, m$ , is of the form  $A_1 \vee \dots \vee A_n$ , and each  $A_j$ ,  $j = 1, \dots, n$  is either a Boolean variable or the negation of a Boolean variable.*

## 2.2.2 First-Order Logic

First-order logic is distinguished from propositional logic by the use of predicates and quantifiers. It provides a syntax capable of expressing detailed mathematical statements, as well as semantics to determine the meaning behind its expressions.

The approach to define FOL varies from author to author. Two of them are more common than others: (1) A more simplistic approach starting with the signature  $\sigma$  (Papadimitriou, 1994), and (2) a more traditional approach starting with a language  $L$  and a predicate logic without operations (Simpson, 2011).

### Syntax

We will describe the more simplistic approach using the *signature* and mention parallels to other definitions.

**Definition 24** (Signature). *A signature, sometimes referred to as vocabulary,  $\sigma = (\Phi, \Pi, r)$  is a 3-tuple containing two disjoint countable sets  $\Phi$  and  $\Pi$ , and the arity function  $r$ .*

1. *A set  $\Phi$  of function symbols.*
2. *A set  $\Pi$  of relation symbols.*
3. *An arity function  $r$  mapping  $\Phi \cup \Pi$  to the non-negative integers.*

The notation for a signature varies from author to author, but since we use  $\Sigma$  to denote an alphabet, we will use  $\sigma$  for a signature.

The arity function tells us how many arguments each function- and relation symbol takes.

*Example 3.* Let  $\Phi$  be a set of function symbols and  $r$  the arity function. A function symbol  $f \in \Phi$  with  $r(f) = n$  is called a  $n$ -ary function symbol. The same principle applies to relation symbols.

Concerning the arity of the relation symbols, it might be interesting to note, that while some author say that a relation symbol can never be 0-ary (Papadimitriou, 1994), other say that 0-ary relation symbol behave as propositional atoms (Simpson, 2011). If 0-ary relations symbols are allowed, then every formula of propositional

logic is also a formula of first-order logic and thus predicate calculus could be seen an extension of the propositional calculus (Simpson, 2011), which is the approach we will follow.

The alphabet  $\Sigma$  consists of *logical* and *non-logical* symbols. We already know the non-logical symbols, defined as the atoms of our signature. The set of logical symbols contain: (1) The Boolean connectives including implication ( $\Rightarrow$ ) and biimplication ( $\Leftrightarrow$ ), (2) parentheses, brackets and other punctuation symbols, (3) a fixed, countably infinite set of object variables  $\{x, y, z, \dots\}$ , (4) a special<sup>1</sup> binary equality symbol = sometimes known as identity symbol, (5) and the two quantifier.

*Remark 3.* Even though the equality relation is mentioned separately, there is no reason to treat it that way in terms of semantics, as we will see later.

*Remark 4.* It might be interesting to note, that sometimes implication and biimplication are not explicitly stated as connective, since given the Boolean expressions  $\phi_1$  and  $\phi_2$ :  $\phi_1 \Leftrightarrow \phi_2$  is shorthand for  $((\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1))$  and  $\phi_1 \Rightarrow \phi_2$  is shorthand for  $(\neg\phi_1 \vee \phi_2)$  (Papadimitriou, 1994).

**Definition 25** (Quantifier). *We introduce the universal quantifier ( $\forall$ ), read as “for all”, and the existential quantifier ( $\exists$ ), read as “there exists”.*

*Remark 5.* It might be interesting to note, that sometimes the existential quantifier ( $\exists$ ) is not treated extra, but as a special case of the universal quantifier ( $\forall$ ) defined as follows (Lifschitz, 2009):  $\exists x\phi \equiv \neg\forall x\neg\phi$  (We will see what  $\equiv$  means in FOL further down).

Since FOL is a formal language, it can be mechanically determined whether some expression is legal. There are two kinds of legal expressions: **terms** and **formulas**. In short, terms represent objects, while formulas express predicates that can take the two truth values.

**Definition 26** (Term). *Let  $\sigma = (\Phi, \Pi, r)$  be a signature. A term can be any one of the following:*

1. *An object variable,*
2. *An object constant,*
3. *If  $f \in \Phi$  is a  $n$ -ary function symbol with  $n > 0$ , and  $t_1, \dots, t_n$  are terms, then  $f(t_1, \dots, t_n)$  is a term.*

*Remark 6.* An object constant is a function constant of arity 0.

**Definition 27** (Atomic formula). *Let  $\sigma = (\Phi, \Pi, r)$  be a signature and  $R \in \Pi$  an  $n$ -ary relation symbol. If  $t_1, \dots, t_n$  are terms, then the expression  $R(t_1, \dots, t_n)$  is called an atomic formula, or atomic expression.*

**Definition 28** (First-order formula). *A first-order formula, also known as first-order expression, can be anything of the following:*

---

<sup>1</sup>The equality symbol is sometimes just assumed to be contained within  $\Pi$

1. An atomic formula,
2. An expression of the form  $\neg\phi$ , where  $\phi$  is a formula,
3. An expression of the form  $(\phi_1 \odot \phi_2)$ , where  $\phi_1$  and  $\phi_2$  are formulas and  $\odot$  is a binary propositional connective,
4. An expression of the form  $\forall x\phi$  or  $\exists x\phi$ , where  $\phi$  is a formula and  $x$  is a variable,

## Semantics

On to semantics. To determine truth of an expression we introduce the notion of a **model** as an analog of truth assignment in Boolean logic. However, its a far more complicated object, since variables, functions, and relations can now take on much more complex values than just true or false.

**Definition 29** (Model). Let  $\sigma = (\Phi, \Pi, r)$  be a signature. A model  $M$  appropriate to  $\sigma$  is a pair  $M = (U, \mu)$ , where  $U$  is a non-empty set called the universe of  $M$ , and  $\mu$  is a function mapping the symbols of  $\sigma$  and the variables to elements in  $U$ . The special equality symbol  $=$  is always assigned to the relation  $=^M$  defined as  $(u, u) : u \in U$ .

Essentially a model is a mathematical *structure*, where  $\mu$  is the interpretation function,  $\sigma$  is the signature and  $U$  is the domain. Sometimes the interpretation function is called *unique valuation* denoted as  $v_M$ . There are also cases where  $\mu$  is used to denote just a variable assignment additionally to an interpretation function  $I$ .

*Remark 7.* Let  $x$  be a variable. We introduce a convention for convenience that metalanguage expressions of the form  $\mu(x)$  will be written as  $x^M$ .

Concerning the interpretation function  $\mu$ , this means that each variable  $x$  will be mapped to an element  $x^M \in U$ , each  $n$ -ary function symbol  $f \in \Phi$  will be mapped to an actual function  $f^M : U^n \rightarrow U$ , and each  $n$ -ary relation symbol  $R \in \Pi$  will be mapped to an actual relation  $R^M \subseteq U^n$ . So, concerning the semantics of terms, if  $t$  is a variable or a constant, then  $t^M$  is explicitly defined by  $\mu$ . On the other hand, if  $t = f(t_1, \dots, t_n)$ , where  $f$  is an  $n$ -ary function symbol and  $t_1, \dots, t_n$  are terms, then  $t^M = f^M(t_1^M, \dots, t_n^M)$ .

Now to the notion of satisfiability, starting with atomic expressions.

**Definition 30.** Let  $M$  be a model,  $\phi$  be an atomic expression,  $R$  be an  $n$ -ary relation symbol, and  $t_1, \dots, t_n$  be terms. We say that  $M$  satisfies  $\phi$ , if  $(t_1^M, \dots, t_n^M) \in R^M$ .

**Definition 31** (Satisfaction). Let  $M$  be a model,  $x$  a variable,  $U$  be the universe of  $M$ ,  $\phi$ ,  $\psi_1$ , and  $\psi_2$  be a first-order expression. We say that  $M$  satisfies  $\phi$ ,  $M \models \phi$  inductively as follows:

1. If  $\phi = \neg\psi_1$  where  $T \not\models \psi_1$  (meaning  $T \models \psi_1$  does not apply),
2. If  $\phi = (\psi_1 \vee \psi_2)$  where either  $T \models \psi_1$  or  $T \models \psi_2$ ,



3. If  $\phi = (\psi_1 \wedge \psi_2)$  where both  $T \models \psi_1$ , as well as  $T \models \psi_2$  hold,
4. If  $\phi = \forall x\psi_1$  where for any  $u \in U$  (with  $M_{x=u}$  being the model identical to  $M$  in all details, except that  $x^{M_{x=u}} = u$ ) it must be that  $M_{x=u} \models \psi_1$ .

A special case of satisfiability is validity. As we now know, a (first-order) expression is satisfiable if a model exists that satisfies it. However, if some expression  $\phi$  is satisfied by any model, we say that the expression is *valid*, denoted  $\models \phi$ .

*Remark 8.* There are three basic reasons why a first-order expression might be valid:

1. Being a tautology similar to boolean logic. For example an expression of the form  $\phi \vee \neg\phi$ .
2. The properties of equality. For example:  $x = x$ , where  $x$  is a variable, will always be valid.
3. The meaning of quantifiers. For example: Let  $x_1, x_2$ , and  $y$  be variables.  $R(x_1, x_2) \Rightarrow \exists yR(x_1, y)$  will always be valid, because if  $R(x_1, x_2)$  evaluates to true, then by definition there must exist a  $y$  such that  $R(x_1, y)$ .

Now that we know validity, we can easily define the metalanguage equality in analog to Boolean logic:

**Definition 32** (Equivalence). *Let  $\phi_1$  and  $\phi_2$  be first-order formulas.  $\phi_1$  and  $\phi_2$  are said to be logically equivalent, written  $\phi_1 \equiv \phi_2$ , if  $\phi_1 \Leftrightarrow \phi_2$  is logically valid.*

*Remark 9.* Note that logical equivalence  $\equiv$  is not the same as the binary equivalence  $=$ , which is part of the alphabet. Logical equivalence is a metalanguage relation used to state that expressions have the same meaning.

And interesting consequence concerning validity is the following proposition used in proof systems as the main method for acquiring new valid sentences.

**Proposition 3** (Modus Ponens). *Let  $\phi_1$  and  $\phi_2$  be first-order expressions. If  $\phi_1$  and  $\phi_1 \Rightarrow \phi_2$  are valid, then  $\phi_2$  is valid Papadimitriou (1994).*

Validity is also a convenient source for simplification. To be precise we can use a few propositions to a first-order expressions into the prenex normal form as defined below:

**Definition 33** (Prenex normal form). *A formula is said to be in prenex normal form if it is of the form  $Q_1x_1\dots Q_nx_n\psi$ , where each  $Q_i$  is a quantifier, each  $x_i$  is a variable, and  $\psi$  is contains no quantifier.*

Before stating some useful propositions, let us first introduce the notion of substitution, that we will need a couple of times henceforth.

**Definition 34** (Substitutable). *Let  $\phi_1$  and  $\phi_2$  be a first-order expression,  $x$  and  $y$  be variables, and  $t$  a term. We say that  $t$  is substitutable for  $x$  in  $\phi_1$ , iff there is no  $y$  in  $t$ , such that some part of  $\phi_1$  of the form  $\forall y\phi_2$  contains a free occurrence of  $x$ .*

Making sure that a term is substitutable is simply a safety net to avoid unintended bindings.

**Definition 35** (Substitution). *Let  $\phi$  be a first-order expression,  $x$  a variable, and  $t$  a term. Let  $t$  be substitutable for  $x$  in  $\phi$ . We define the substitution of  $t$  for  $x$  in  $\phi$ , denoted as  $\phi[x \leftarrow t]$ , to be the expression resulting by replacing each free occurrence of  $x$  in  $\phi$  with  $t$ .*

Let us state a few propositions concerning substitution.

**Proposition 4.** *Let  $\phi$  be a first-order expression,  $x$  a variable, and  $t$  a term. Any expression of the form  $\forall x\phi \Rightarrow \phi[x \leftarrow t]$  is valid (Papadimitriou, 1994).*

**Proposition 5.** *Let  $\phi$  be a first-order expression and  $x$  a variable. If  $\phi$  is valid, then so is  $\forall x\phi$  (Papadimitriou, 1994).*

**Proposition 6.** *Let  $\phi$  be a first-order expression and  $x$  a variable. If  $x$  does not appear free in  $\phi$ , then  $\phi \Rightarrow \forall x\phi$  is valid (Papadimitriou, 1994).*

*Proof.* Any model that satisfies  $\phi$  will also satisfy  $\forall x\phi$ , as the  $x = u$  part of the definition of satisfaction becomes irrelevant (Papadimitriou, 1994).  $\square$

As we know, we can replace any first-order expression with an equivalent one. We now list a few equivalence relations (without proof), that can be used to put any first-order expression into the prenex normal form:

**Proposition 7.** *Let  $\phi_1$  and  $\phi_2$  be first-order expressions. Let  $x$  and  $y$  be variables. Then (Papadimitriou, 1994):*

1.  $\forall x(\phi_1 \wedge \phi_2) \equiv (\forall x\phi_1 \wedge \forall x\phi_2)$ .
2. If  $x$  does not appear free in  $\phi_2$ ,  $\forall x(\phi_1 \wedge \phi_2) \equiv (\forall x\phi_1 \wedge \phi_2)$ .
3. If  $x$  does not appear free in  $\phi_2$ ,  $\forall x(\phi_1 \vee \phi_2) \equiv (\forall x\phi_1 \vee \phi_2)$ .
4. If  $y$  does not appear free in  $\phi_1$ ,  $\forall x\phi_1 \equiv \forall y(\phi_1[x \leftarrow y])$ .

Now that we understand validity, we can concern ourselves with finding a system to reveal validity of expressions. As we have seen, there are three basic kinds of validity, (1) Boolean validity, (2) the properties of equality, and (3) the properties of quantifier. For the system that we are going to use (Papadimitriou, 1994), we assume that the signature  $\sigma$  is fixed.

**Definition 36** (Axioms). *Let  $\phi_1$  and  $\phi_2$  be first-order expressions,  $x$  be a variable, and any  $t$  or  $t_i$  be a term. We introduce a countable infinite set of logical axioms denoted as  $\Lambda$  containing the generalization of the following axioms (Papadimitriou, 1994):*

- Any expression whose Boolean form is a tautology
- Any expression of the form  $t = t$

- Any expression of the form  $\left(\bigwedge_{i=1}^n (t_i = t'_i)\right) \Rightarrow (f(t_1, \dots, t_n) = f(t'_1, \dots, t'_n))$ , where  $f$  is a  $n$ -ary function symbol.
- Any expression of the form  $\left(\bigwedge_{i=1}^n (t_i = t'_i)\right) \Rightarrow (R(t_1, \dots, t_n) = R(t'_1, \dots, t'_n))$ , where  $R$  is a  $n$ -ary relation symbol.
- Any expression of the form  $\forall x \phi_1 \Rightarrow \phi[x \leftarrow t]$ .
- Any expression of the form  $\phi_1 \Rightarrow \forall x \phi_1$ , with  $x$  not free in  $\phi$ .
- Any expression of the form  $(\forall x(\phi_1 \Rightarrow \phi_2)) \Rightarrow (\forall x \phi_1 \Rightarrow \forall x \phi_2)$ .

*Remark 10.* A first-order theorem consists out of a set of axioms in a particular signature. We assume that our  $\sigma$  is *fixed*, so that we avoid having to state it explicitly all the time.

Starting from the axioms the system we use will generate new valid first-order expressions by a method based on *Modus Ponens*.

**Definition 37** (First-order theorem). *Let  $S$  be a finite sequence of first-order expressions  $S = (\phi_1, \phi_2, \dots, \phi_k)$ , and  $\Lambda$  be a set of axioms. The expression  $\phi_k$  is called a first-order theorem, denoted  $\vdash \phi_k$ , if for each expression  $\phi_i$ ,  $1 \leq i \leq k$  either (a)  $\phi_i \in \Lambda$ , or (b) there are two expressions of the form  $\psi, \psi \Rightarrow \phi_i$  among the expressions  $\phi_1, \dots, \phi_{i-1}$ . We say that  $S$  is the proof of  $\phi_k$ .*

But what if we ask ourselves, not whether a sentence is satisfied by all models, but by our favorite model (whatever this may be)? One way to bridge that gap would be the *axiomatic method*. However, in general our favorite model may have an axiomatization that consists of infinite set of expressions. Thus, to answer our question, we need a *proof system* that allows for proofs from infinitely many premises.

**Definition 38** (Valid consequence). *Let  $\Delta$  be a set of expressions, and  $\phi \notin \Delta$  be another expression. We say that  $\phi$  is a valid consequence of  $\Delta$ , written  $\Delta \models \phi$ , if any model that satisfies each expression in  $\Delta$  also satisfies  $\phi$ .*

This implies that the valid consequences would be all the properties (and only those) of our favorite model. Hence generating those in a systematic way would be desirable. For this purpose we extend the proof system to be helpful in identifying valid consequences  $\Delta$  (Papadimitriou, 1994).

**Definition 39** ( $\Delta$ -first-order theorem). *Let  $\Delta$  be a set of expressions. Let  $S$  be a finite sequence of first-order expressions  $S = (\phi_1, \phi_2, \dots, \phi_k)$ , and  $\Lambda$  be a set of expressions. The expression  $\phi_k$  is called a  $\Delta$ -first-order theorem, denoted  $\vdash \phi_k$ , if for each expression  $\phi_i$ ,  $1 \leq i \leq k$  either (a)  $\phi_i \in \Lambda$ , or (b)  $\phi_i \in \Delta$ , or (c) there are two expressions of the form  $\psi, \psi \Rightarrow \phi_i$  among the expressions  $\phi_1, \dots, \phi_{i-1}$ . We say that  $S$  is the proof of  $\phi_k$  from  $\Delta$ .*

*Remark 11.* A  $\Delta$ -first-order theorem would be an ordinary first-order theorem if we allowed all the expressions in  $\Delta$  to be added to our logical axioms. In that context the expressions of  $\Delta$  are called to non-logical axioms of the system (Papadimitriou, 1994).

## Computational Questions

There are some important computational questions that are worth mentioning. As we know, first-order expressions can be encoded as strings in an appropriate alphabet  $\Sigma$ . The same alphabet can also encode proofs, which, as we have seen, are just sequences of expressions.

*Question 1 (THEOREMHOOD).* Given the encoding of an expression  $\phi$ , is  $\phi$  a first-order theorem ( $\vdash \phi$ )?

**Proposition 8.** *THEOREMHOOD is recursively enumerable.*

*Proof.* The Turing machine, that accepts the language of THEOREMHOOD, tries all possible finite sequences of expressions (proofs) and reports “yes”, if one of them is indeed a proof of the given expression (Papadimitriou, 1994).  $\square$

*Question 2 (VALIDITY).* Given the encoding of an expression  $\phi$ , is it valid?

Using the  $\Delta$ -first-order theorem, we can simplify even proofs of validity (Papadimitriou, 1994). This can be done using three interesting results that formalize pattern of thought very common in mathematical reasoning, namely *the deduction technique, arguing by contradiction, and justified generalization.*

**Theorem 1 (The Deduction Technique).** *Let  $\phi_1, \phi_2$  be expressions, and  $\Delta$  be a set of expressions. If  $\Delta \cup \{\phi_1\} \vdash \phi_2$ , then  $\Delta \vdash \phi_1 \Rightarrow \phi_2$  (Papadimitriou, 1994).*

The next proof method is arguing by contradiction. Basically, that means that if we want to  $\phi$ , we assume  $\neg\phi$  and arrive at a contradiction.

**Definition 40 (Contradiction).** *Let  $\phi$  be an arbitrary expression, then a contradiction can be defined as  $\phi \wedge \neg\phi$ .*

**Definition 41 (Consistency).** *Let  $\Delta$  be a set of expressions. If  $\Delta \vdash \phi$ , for any expression  $\phi$  (including contradictions) we say that  $\Delta$  is inconsistent. If no contradiction can be proved from  $\Delta$ , then we say  $\Delta$  is consistent.*

**Theorem 2 (Arguing by Contradiction).** *Let  $\phi$  be an expressions, and  $\Delta$  be a set of expressions. If  $\Delta \cup \{\phi\}$  is inconsistent, then  $\Delta \vdash \phi$  (Papadimitriou, 1994).*

**Theorem 3 (Justified Generalization).** *Let  $\phi$  be an expressions,  $x$  be a variable, and  $\Delta$  be a set of expressions. If  $\Delta \vdash \phi$  and  $x$  is not free in any expression of  $\Delta$ , then  $\Delta \vdash \forall x\phi$  (Papadimitriou, 1994).*

Next we are going to state reassuring property of this proof system, called the *soundness theorem*, showing that the proof system only proves valid consequences

**Theorem 4 (The Soundness Theorem).** *Let  $\phi$  be an expressions, and  $\Delta$  be a set of expressions. If  $\Delta \vdash \phi$ , then  $\Delta \models \phi$  (Papadimitriou, 1994).*

Furthermore, the reverse of the soundness theorem, called the *completeness theorem* due to Kurt Gödel, states that the proof system is able to prove all valid consequences (Papadimitriou, 1994).

**Theorem 5** (Gödel's Completeness Theorem). *Let  $\phi$  be an expressions, and  $\Delta$  be a set of expressions. If  $\Delta \models \phi$ , then  $\Delta \vdash \phi$  (Papadimitriou, 1994).*

Let  $\phi$  be some first-order expression. If we take a look a the last two theorems, it is now clear that  $\models \phi$ , if and only if  $\vdash \phi$ , and we can revisit the initial question concerning the computational problem VALIDITY. Since soundness and completeness theorems identify valid sentences with first-order theorems, it is the same as THEOREMHOOD.

**Corollary 1.** *VALIDITY is recursively enumerable (Papadimitriou, 1994).*

But this is not the only consequence of the completeness theorem, as the following important property of first-order logic shows.

**Theorem 6** (The Compactness Theorem). *If all finite subsets of a set of sentences  $\Delta$  are satisfiable, then  $\Delta$  is satisfiable (Papadimitriou, 1994).*

**Corollary 2.** *If a sentence has a model, it has a countable model (Papadimitriou, 1994).*

Of course, a countable model can still be either finite or infinite. So is it true, that all models have a countable infinite model?

**Theorem 7** (Löwenheim-Skolem Theorem). *If a sentences  $\phi$  has finite models of arbitrary large cardinality, then it has an infinite model (Papadimitriou, 1994).*

We will see why this is important, when we take a look at the limitations of first-order logic.

## Limitations

Although powerful, first-order logic has his limitations. First of, FOL is *undecidable*, which means that a sound, complete and terminating decision algorithm is impossible.

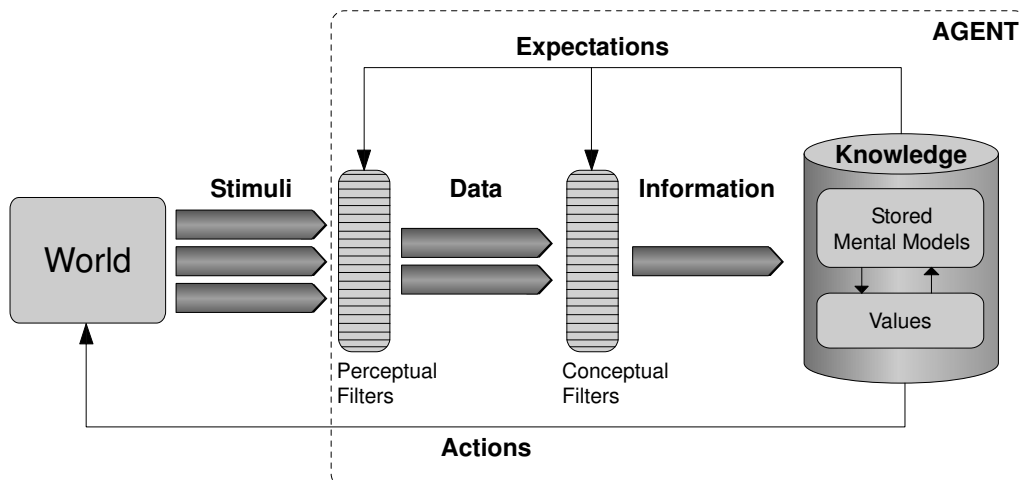
Let us use the Löwenheim-Skolem theorem to show a limitation of the expressive power of first-order logic. Without proof, let us state that any graph property can be expressed by a sentence  $\phi$ . The corresponding computational problem is called  $\phi$ -GRAPHS and has polynomial time. Now an interesting computational question arises:

*Question 3.* Are all polynomial graph properties expressible?

The answer is no. The reason for this is the property called REACHABILITY that is described by the following question

*Question 4* (REACHABILITY). Given a graph  $G$  and two nodes  $x$  and  $y$  of  $G$ , is there a path from  $x$  to  $y$ ?

**Corollary 3.** *There is no first-order expression  $\phi$ , such that  $\phi$ -GRAPHS is the same as REACHABILITY (Papadimitriou, 1994).*



**Figure 2.1:** Data, information, and Knowledge. The agent and its interaction with the world, as depicted by Boisot and Canals (2004).

This inexpressibility of REACHABILITY by first-order logic is very interesting, because it is a non-trivial impossibility theorem. This has motivated the study of possible extensions of first-order logic, such as second-order logic.

Concerning computation linguistics, first order logic is able to formalize many simple quantifier constructions in natural language. However, many features of natural language need a much richer structure and cannot be expressed with first order logic. This is in addition to already restricting complexity limitations.

## 2.3 Data, Information, and Knowledge

Our understanding of terms such as “data” is a little uncommon, but based on what we think are good reasons. One being that natural language understanding is an AI-complete problem (Waldrop, 1984). It makes more sense to ground the semantics of those terms closer to human understanding rather than “traditional” computer models.

Let us now state our understanding of certain basic terms; namely data, information, and knowledge. These terms are interpreted quite differently throughout the scientific literature (Zins, 2007).

We ground our definitions on Boisot and Canals (2004) and their sources. They describe *data* as originating in discernible differences in physical states-of-the-world, registered through stimuli. These states are describable in terms of space, time, and energy. Significant regularities in this data - whatever one qualifies as significant - then constitutes *information*. This implies that the information gained from data, depends on the agent extracting it - more precisely: his expectations, or *hypotheses*. This set of hypotheses held by an agent can then be referred to as *knowledge* and is constantly modified by the arrival of information. Figure 2.1 visualizes those stated definitions with the help of an *agent-in-the-world* (Boisot and Canals, 2004).

Since what qualifies as significant depends on the agents individual disposition,

information can only appear to be *objective*, if what constitutes as significant regularities is established through convention (Boisot and Canals, 2004).

It might be interesting to note, that based on these definitions, the commonly used term “*unstructured data*” refers to complete randomness, or noise.

### 2.3.1 Unstructured Information

Unstructured information often refers to natural language, be it in the form of written documents, speech, audio, images or video (Ferrucci et al., 2009). This implicit definition makes sense, as it is used to split information into two easily understood classes: databases content and everything else. The reason for this is mostly business motivated, as the term “unstructured” is then used to convey the message of computational inaccessibility through information retrieval methods to the “stored” information, and hence a necessity for action. For example Ferrucci and Lally (2004b) define structured information as information whose intended meaning is unambiguous and explicitly represented in the structure or format of the data and unstructured information as information whose intended meaning is only loosely implied by its form. Still, the term “unstructured information” is just vaguely defined by discussing it in contrast to “structured information”, without defining the term “information” itself.

Let us state a somewhat different but similar approach to a definition based on the last section:

**Definition 42** (Unstructured information). *Unstructured Information is the subset of information, where the information itself describes parts of what constitutes as significant regularity.*

What this essentially means, is that *information* and its *structure* are not completely separable. The best example for unstructured information is in text. The meaning of the text - its nouns, verbs, markers and so fourth - partly depends on the text itself - on the context and discourse. Even for humans it can be difficult to understand the text. Sometimes sentences have to be re-read to be understood, or are misunderstood completely. While processing text, our knowledge-base is constantly being updated by the text itself, and a combination of our previous knowledge and updated knowledge is used to overcome and interpret uncertainties. A physician would most certainly extract completely different information out of a patient file, than a linguist.

### 2.3.2 Knowledge Discovery

The term *knowledge discovery* (KD) was coined on the application on databases (Maimon and Rokach, 2005), with the goal of identifying valid, novel, useful and understandable patterns in mind. Holzinger (2013) states, that the term gained its popularity through the paper of Fayyad et al. (1996), who described a nine stage process of *knowledge discovery from data* (KDD):

1. Learning from the application domain
2. Creating a target data set
3. Data cleansing and pre-processing
4. Data reduction and projection
5. Choosing the function of data mining
6. Choosing the data mining algorithm
7. Data mining
8. Interpretation
9. Using discovered knowledge

So, the central core of this process is data mining, choosing the appropriate algorithm, and somehow discover interesting patterns. These patterns, that we previously referred to as significant regularities, would be the ominous information that we'd be hoping for, to then hopefully be able to learn something useful. Even though, these definitions evolved around structured data in databases, they go hand in hand with our understanding of knowledge

Let us state a definition based on our interpretation of what knowledge is:

**Definition 43** (Knowledge Discovery). *If knowledge consists of a set of hypotheses, then knowledge discovery is the process of finding or generating new hypotheses out of information.*

Although KDD is associated with the automatic analysis and modeling of large data repositories (Holzinger, 2013), I strongly believe that real knowledge discovery is, at least for now, a process in which the human intelligence is far superior over machine intelligence. Consequently, utilizing the tools of *Human-Computer Interaction* (HCI) to assist the human in this process might yield very fruitful results.

For this reason, we will strongly distinguish between KDD, which is an established field that deals mainly with aspects of machine intelligence and automatic data mining (Holzinger, 2013), and KD, which is the process of generation new hypotheses out of information.

### 2.3.3 Human-Computer Interaction (HCI)

The field of HCI evolved from the interest of computer science in input-output technology. Today it deals mainly with human perception, cognition, intelligence and sense-making in the context of computer science (Holzinger, 2013). So its focus is literally the interaction between humans and machines.

Recently, Holzinger (2013) stated HCI's main research question as: “***What is interesting?***”



This essential question, however, does not always have a simple logical answer. Beale (2007) describes this “interest” as a perspective on relationships between data, which is influenced by tasks and prior knowledge. Thus “interest” is essentially a human construct.

### 2.3.4 HCI-KDD

Consequently, a novel approach is to combine HCI & KDD (Holzinger, 2012) in order to enhance human intelligence by computational intelligence. The main contribution of HCI-KDD is to *enable* end users to *find and recognize* previously unknown and potentially useful, usable, and interesting information. Yet, what is interesting is a matter of research (Beale, 2007).

HCI-KDD may be defined as the process of identifying novel valid, and potentially useful data patterns, with the goal to understand these patterns (Funk and Xiong, 2006). This approach is based on the assumption that the domain expert possesses explicit domain knowledge and by enabling him to interactively look at his data sets, he may be able to identify, extract and understand useful information, as to gain new - previously unknown - knowledge (Holzinger, Scherer, Seeber, Wagner and Müller-Putz, 2012).

## 2.4 Computational Linguistics (CL)

The term "*Computational Linguistics*" stands for an interdisciplinary field that studies the statistical and rule-based modeling of natural language from a computational perspective. It belongs to the cognitive sciences and has applied as well as theoretical components, that come from linguistics, computer science, psychology, and mathematics (Uszkoreit, 2000). Over the past decades, there have been great advantages in the area of computational methods for extracting meaning from text (McNamara, 2010; Rosenfeld, 2000).

### 2.4.1 Morphology

Most natural languages have some system to generate words and word forms from smaller units in a systematic way (Clark et al., 2010; Mitkov, 2003). The seemingly infinity of words in a language is produced by a finite collection of smaller units called *morphemes*. Simply put, morphology deals with the structure of words. These morphemes are either semantic concepts like *door*, *house*, or *green*, which are also called roots, or abstract features like *past* or *plural* (Mitkov, 2003). Their realization as part of a word are then called *morph*, such as *door* or *doors*.

The information expressed with morphology varies widely between languages. In Indo-European languages for example, distinct features are merged into a single bound form (Mitkov, 2003). These languages are typically called *inflectional languages*. Inflections do not change the POS category, but the grammatical function. Inflections and derivations convey information such as tense, aspect, gender or case.

## 2.4.2 Lexicography

The term “*computational lexicography*” can have different meanings. Hanks (Mitkov, 2003) listed two common interpretations:

1. Restructuring and exploiting human dictionaries for computational purposes
2. Using computational techniques to compile new dictionaries

In this paper we refer to the exploiting of human dictionaries for computational purposes.

## 2.4.3 Finite-State Technology

Many of the basic steps in NLP, such as tokenization and morphological analysis, can be carried out efficiently by the means of finite-state transducers (Mitkov, 2003). These transducers are generally compiled from *regular expressions*, which is a formal language for representing sets and relations (Mitkov, 2003).

## 2.4.4 Text Segmentation

Text segmentation is an important step in any NLP process. Electronic text in its raw form is essentially just a sequence of characters. Consequently it has to be broken down into linguistic units. Such units include *words, punctuation, numbers, alphanumerics*, etc. (Mitkov, 2003). This process is also referred to as *tokenization*. Most NLP techniques also require the text to be segmented into sentences and maybe paragraphs as well (Mitkov, 2003).

## 2.4.5 Part-of-Speech Tagging

Most tasks NLP require the assignment of classes to linguistic entities (tokens) (Clark et al., 2010). Part-of-Speech (POS), for instance, is an essential linguistic concept in NLP, and POS tagger are used to assign syntactic categories (e.g. noun, verb, adjective, adverb, etc.) to each word (Clark et al., 2010; Manning and Schütze, 1999).

Automatic part-of-speech taggers have to handle several difficulties, including the ambiguity of word forms in their part-of-speech (Schmid, 1994), as well as classification problems due to the ambiguity of periods (’), which can be either interpreted as part of a token (e.g. abbreviation), punctuation (full stop), or both (Clark et al., 2010).

## 2.4.6 Information Extraction

The build-in tools of ICA follow the idea of information extraction (IE). Grishman (2003) (Clark et al., 2010; Mitkov, 2003) defines IE as the process of automatically

identifying and classifying instances of entities, relations and events in a text, based on some semantic criterion.

Typical tasks are *name-*, *entity-*, *relation-* and *event extraction* (Clark et al., 2010).

## 2.5 Unstructured Information Management Architecture (UIMA)

Initially developed by IBM, UIMA now is an established *OASIS standard*<sup>2</sup>. Its objective is to support inter-operability of *analytics*, which is the term they use for a software object or network service that performs some sort of analysis, across *platforms*, *frameworks*, and *modalities* (Ferrucci et al., 2009).

To be more precise, Ferrucci and Lally (2004b) describe UIMA’s high-level objective as two fold:

1. To accelerate scientific advances by enabling the rapid combination of best of breed solutions (frameworks) for different modalities.
2. To accelerate the transfer of analytics implementations to the product by providing an architecture (and framework) that promotes reuse and supports flexible deployment options.

### 2.5.1 History and Motivation

IBM realized that market indicators suggested an increasing market need for commercial applications in text- and voice analytics (Ferrucci and Lally, 2004b). These application areas they identified included life sciences, e-commerce, technical support, advanced search, as well as national and business intelligence. Ferrucci and Lally (2004b) reported that IBM had over 200 employees from six major labs all over the world working on *Unstructured Information Management* (UIM) technologies. Their primary focus being NLP at that time.

The idea was to overcome a few major challenges when dealing with (third-party) UIM assets, by creating a common framework and engineering discipline. Ferrucci and Lally (2004b) state the following:

- **Organizational Structure.** Geographically dispersed teams make jointly development and reuse of technologies difficult.
- **Skills Alignment.** Inefficiencies arise when forcing specialists to perform tasks that do not fall into their skill set, but are better of delegated to where the actual expert knowledge resides.
- **Development Inefficiencies.** Reuse and integration often requires too much effort for adaption, resulting in unnecessary “reinvention of the wheel”.

---

<sup>2</sup>As of March 2009

- **Speed to product.** Technology transfer from the research lab to the commercial products often requires a significant rewrite effort for researchers.

As stated before, skills alignment was a major challenge that needed to be addressed. Language technology experts, for example, might not be trained software engineers. Yet one of UIMA’s objectives is to enable efficient deployment in robust and scalable system architecture.

As a result, Ferrucci and Lally (2004b) identified the following development roles analogous to the separation of roles in Sun’s J2EE platform:

- **Annotator Developer.** The focus lies on developing core algorithms, be it of statistical or rule-based nature. Its their job to define the interfaces, descriptions, and algorithms in an UIMA conform way. The annotator developer is insulated from technical inter-operability, external control, distribution, and deployment concerns.
- **Analysis Engine Assembler.** The analysis engine assembler combines various annotators created by the developer, by considers available engines in terms of their capabilities for the task at hand. The assembler is insulated from: algorithm development, technical interoperability, external control, distribution and deployment concerns.
- **Analysis Engine Deployer.** The analysis engine deployer decides how these assembled analysis engines, as well as the required resources, are deployed on particular hardware.

## 2.5.2 Specification

To begin with, Ferrucci et al. (2009) introduce and describe the following terms and we will follow their definitions:

The subject of analysis (sofa) is referred to as an *artifact*. If, for example, we analyze a collection of documents, than each document would be an artifact. The artifact *modality* is the “mode of communication” the artifact represents (text, sound, video, etc.). The union of an artifact and its meta-data is called *artifact data*.

We stated earlier, that the principal objective of the UIMA specification was to support inter-operability among analytics. This objective was subdivided into the following four design goals (Ferrucci et al., 2009):

1. **Data Representation.** Support the common representation of the artifacts and their meta-data independently of their initial representation.
2. **Data Modeling and Interchange.** Render the transfer of the artifacts and their meta-data platform independent.
3. **Discovery, Reuse and Composition.** Support the discovery, reuse and composition of independently-developed analytics.

4. **Service-Level Inter-operability.** Support concrete inter-operability of independently developed analytics based on a common service description.

There are seven elements to the UIMA standard. Following, we will provide a brief summary of their description, as initially depicted by Ferrucci et al. (2009).

## Common Analysis Structure (CAS)

The subsystem of UIMA is the *Common Analysis System* (CAS), which handles data exchanges between the various UIMA components, including analysis engines and unstructured information management applications. CAS supports data modeling via a type system and is independent of any programming language. It provides data access through a powerful indexing mechanism, hence provides support for creating annotations on text data (Gotz and Suhre, 2004). Furthermore, it delivers a neutral, object-based representation scheme that is aligned with UML (Ferrucci et al., 2009).

To break it down, CAS is the common data structure containing the *artifact* and its *meta-data* (annotations). The meta-data is essentially an object graph generated by the analysis work-flow, where objects are instances of classes (types) in a type system. The reason, why the artifact and its annotations are separate, is to support the stand-off annotation model. This means, that the artifact itself is not modified in the process. All changes are made to the meta-data.

## Type System Model

One of the design goals was *data modeling and interchange*. To support that goal, UIMA requires each CAS to conform to a user-defined schema, called *type system* (Ferrucci et al., 2009). A type system is essentially a collection of type definitions, each of which can specify various attributes.

*Example 4.* Lets say we want to create an annotator that identifies persons and organizations within some text documents. We might end up utilizing a type system that specifies the following types and attributes: (1) One type called **Person** with attributes along the line of *First Name* and *Last Name*, and (2) another type called **Organization** that has the attributes *Name* and *Location*.

## Base Type System

To establish a basic level of inter-operability, UIMA specifies a base type system providing definitions for commonly-used, domain-independent types (Ferrucci et al., 2009). The most significant of which is the *Annotation and Sofa Type System*. However, the basic type system also includes:

- Primitive Types,
- Views, and
- Source Document Information

## Abstract Interfaces

The abstract interfaces of UIMA define standard component types and operations that UIMA services implement (Ferrucci et al., 2009). As mentioned before, all UIMA services operate on the CAS. To enable that, abstract interfaces have to be specified.

The super-type of all UIMA components is called *Processing Element* (PE), which defines methods to access the PE meta-data. *Analytic* is a subtype of PE and performs some form of analysis on the CAS. Another subtype of PE is called *Flow Controller*. Its purpose is to specify the route a CAS takes through different analytics.

## Behavioral Meta-data

The *behavioral meta-data* of an analytic describes what that analytic does (Ferrucci et al., 2009). It specifies what types of CAS it can process, what elements (or annotations) it needs as input (be it required or optional), as well as what kind of changes the analytic does to the CAS. These changes can include the creation, deletion, or simply modification of annotations.

It might be interesting to note that analytics are not required to declare behavioral meta-data.

## Processing Element Meta-data

On the other hand, each PE is required to specify its *processing element meta-data* (Ferrucci et al., 2009). This meta-data describes the PE itself by specifying identification information (i.e. a unique name), possibly configuration parameters for its implemented algorithm, behavioral meta-data, the utilized type system, and beyond-UIMA extensions.

## WSDL Service Descriptions

Essentially, this element facilitates inter-operability by specifying a WSDL description of the UIMA interfaces (Ferrucci et al., 2009). It also specifies a binding to a concrete SOAP interface that must be implemented by compliant services.

### 3. Related Work

For many years, IBM research groups from various countries are working on the development of systems for text analysis, and text-mining methods to support problem solving in life science. The best known system today is called Biological Text Knowledge Services and integrates research technologies from multiple IBM research labs. BioTeKS is the first major application of the so-called Unstructured Information Management Architecture (UIMA) initiative (Ferrucci and Lally, 2004*a*). These attempts go back to a text mining technology called TAKMI (Text Analysis and Knowledge Mining), which has been developed to acquire useful knowledge from large amounts of textual data - not necessarily focused on medical texts (Nasukawa and Nagano, 2001).

BioTeKS was originally intended to analyze biomedical text from MEDLINE abstracts, where the text is analyzed by automatically identifying terms or names corresponding to key biomedical entities (e.g., proteins, drugs, etc.) and concepts or facts related to them (Mack et al., 2004). MEDLINE has been often used for testing text analytics approaches and meanwhile a large number of Web-based tools are available for searching MEDLINE. However, the non-standardized nature of text is still a big issue, and there is much work left for improvement. A big issue is in end-user centred visualisation and visual analytics of the results, required for the support of the sensemaking processes amongst medical professionals (Holzinger, Simonic and Yildirim, 2012; Holzinger, Yildirim, Geier and Simonic, 2013).

Many solutions for data analytics are available either as commercial or open-source software, ranging from programming languages and environments providing data analysis functionality to statistical software packages to advanced business analytics and business intelligence suites.

Prominent tools focusing on statistical analysis are IBM SPSS, SAS Analytics as well as the open-source R project for statistical computations. Each of the aforementioned tools provides additional packages for text analysis, namely IBM SPSS Modeler, a data mining and text analytics workbench, SAS Text Analytics and the `tm` package for text mining in R.

Software focusing on text mining and text analysis like the Apache UIMA project or GATE (General architecture for text engineering) are aimed at facilitating the analysis of unstructured content. Several projects based on the UIMA framework provide additional components and wrappers for 3rd-party tools, with the purpose of information extraction in the biomedical and the healthcare domain, including Apache `cTAKES` (clinical Text Analysis and Knowledge Extraction System) and the BioNLP UIMA Component Repository. Other solutions for knowledge analysis utilize machine learning algorithms and techniques, with the most prominent

frameworks Weka (Waikato Environment for Knowledge Analysis) and RapidMiner.

In the medical field, the secondary use of electronic health record (EHR) data is an important domain. Chute et al. (2011) state, that the entire categories of clinical research are fundamentally dependent on effective secondary use of clinical information. This includes - but is not limited to - clinical trials, outcomes research, and best evidence discovery. A big step towards an effective and ethical use of EHR data for secondary purposes is SHARPN (Chute et al., 2011). The SHARPN project focuses on three main themes, namely *Normalization*, *Phenotypes*, and *Data Quality/Evaluation*. It is described as - and I quote - “a collaboration among 16 academic and industry partners committed to the production and distribution of high-quality software artifacts that support the secondary use of EMR data” (Chute et al., 2011).

Concerning medical standards for UIMA, Wu et al. (2013) created a common type system for clinical NLP that is fully functional in cTAKES, with an end target of deep semantics based on Clinical Element Models.

Kano et al. (2009) describe U-Compare, a joint project between the University of Tokyo, the University of Colorado School of Medicine and the National Centre for Text Mining at the University of Manchester. Its purpose is to share and compare text mining tools with UIMA and claimed to provide the world’s largest collection of typesystem compatible UIMA resources.

To our knowledge there are only a few publications concerning the integration of UIMA into clinical routine:

Garvin et al. (2012) built a natural language processing system to extract information on left ventricular ejection fraction, which is a key component of heart failure, from “free text” echocardiogram reports to automate measurement reporting and to validate the accuracy of the system using a comparison reference standard developed through human review. For this purpose they created a set of regular expressions and rules to capture “ejection fraction” using a random sample of 765 echocardiograms. The authors assigned the documents randomly on two sets: a set of 275 used for training and a second set of 490 used for testing and validation. To establish a reference standard, two independent experts annotated all documents in both sets; a third expert resolved any incongruities. The test results for documentlevel classification of EF of  $< 40\%$  had a sensitivity (recall) of 98.41%, a specificity of 100%, a positive predictive value (precision) of 100%, and an F measure of 99.2%. The test results at the concept level had a sensitivity of 88.9% (95% CI 87.7% to 90.0%), a positive predictive value of 95% (95% CI 94.2% to 95.9%), and an F measure of 91.9% (95% CI 91.2% to 92.7%) - consequently, the authors came to the conclusion that such an automated information extraction system can be used to accurately extract EF for quality measurement (Garvin et al., 2012).



## 4. Materials

In our experiment, we are interested to investigate the potential to support a medical doctor (dermatologist) in his research by applying NLP techniques to selected medical sample records in the form of jointed electronic *doctor's letters* called *out-patient cards*. These files contain a number of structured information, such as *patient name* and *date of birth*, but most of it is unstructured information in the form of written electronic text. To realize this, we used *IBM Content Analytics* (ICA) in combination with *ICA Studio 3.0* and the *Apache UIMA* framework.

### 4.1 Electronic Patient Files

As of now, electronic patient files contain large portions of data which has been entered in non-standardized format. Especially in the German speaking countries, text seems to be the preferred way of documentation and communication when it comes to patient care.

One interesting type of an electronic patient file is the electronic *doctor's letter*. Its primary purpose is the communication between physicians. The content of a doctor's letter usually consists of a summary of the patients *status*, a review and interpretation of the patients *disease-progression*, a description of the induced *therapies* and *medications*, as well as a possible recommendation for *further steps*.

In our case, the doctor's letters where dermatology specific and limited to one clinic. This, of course, has influence on the required complexity of the NLP rule-set. And a positive at that, since this limits the domain to one medical field of one hospital and we only have to deal with a very small variation of "doctor's dialects".

However, the idea was to investigate the NLP possibilities with very limited structure available, in order to support backwards compatibility. With the restriction in place that the files are fully digital PDFs (and not scanned images). This also means, that linebreaks and punctuation can in general not be trusted. Essentially the only data source was a collection of *out-patient cards*. An out-patient card is the complete union of all available doctor's letters of a single patient. This implies that each patient was represented by one PDF file, and each PDF file resembled the collective information of one single patient.

An illustration of a out-patient card can be seen in Fig. 4.1. Each visit represents one doctor's letter. In the PDF these doctor's letters are separated via horizontal lines. The content of such a letter is usually divided into different subsections. There are different kinds of subsections, that may or may not occur in the file, but it always

starts with a brief summary of the diagnosis. This summary usually lists the past intervention, as well as the patients status afterwards.

Let us state the most commonly occurring subsections (Indicated in Fig: 4.1 by the placeholder-word “Subsection”) and their content:

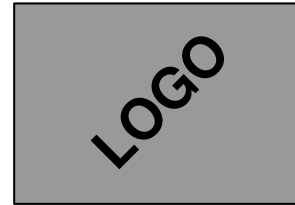
- **Letzte Durchuntersuchungen.** Lists the dates and results of the past examinations. These can, for example, include summaries of Sonograph, or CT results.
- **Anamnese.** The *medical history* (English translation) of a patient according to the patient or other people that know the patient. It is information gained by a physician by asking specific questions.
- **Inspektion und Palpation.** The results of the inspection and palpation (touching) of the patients body by the physician.
- **Auflichtmikroskopischer Befund.** The reflected-light microscopy is a non-invasive diagnostic method in dermatology for early detection of malign melanoma. This subsection lists the corresponding results and interpretation.
- **Digitale Dokumentation.** References to other digital documentation associated with the patient.
- **Histologie.** This subsection documents the results of tissue analysis; It also lists the associated dates.
- **Labor.** All the performed Labor results, such as EKG measurements or Liver function tests.
- **Therapie.** This subsection lists all the recommended or running therapies of the patient.
- **Kontrolle.** Lists suggestions or dates for future examinations.
- **Untersuchung durch.** The name of the doctor that performed the current examination that this doctor’s letter is documenting.

## 4.2 Apache UIMA Framework

Apache UIMA is an open-source implementation of the UIMA OASIS standard (see also 2.5). There are two basic frameworks: One for C++ and one for Java. These frameworks support the configuration, description, deployment, all the way up to actually executing pipelines of annotator components in a run-time environment. The Java implementation also provides an *Eclipse*-based (Eclipse Foundation, 2004) development environment for using UIMA. The framework also allows for the creation of annotators in Perl, Python, and TCL.

## Name of the Clinic / Hospital

hospital information, such as address etc.  
more hospital information.



## Addressee

addressee information

Patient name

more hospital information.

more hospital information.

**CAESAR, JULIUS**

Ambulant am 20.01.2013

Geboren am: 20.01.1901

birth date

Patientennummer: 1234567

patient ID

## Ambulanzkarte

20.01.2013

date of visit

**Befund - Amb. fuer Melanomnachsorge**

department

medical information concerning the diagnosis  
free text of arbitrary length

### Subsection

medical information concerning that subsection  
free text of arbitrary length

### Subsection

such as THERAPIE,  
or LABOR

medical information concerning that subsection  
free text of arbitrary length

### Subsection

medical information concerning that subsection  
free text of arbitrary length

20.02.2012

**Befund - Amb. fuer Melanomnachsorge**

medical information concerning the diagnosis  
free text of arbitrary length

### Subsection

medical information concerning that subsection  
free text of arbitrary length

### Subsection

medical information concerning that subsection  
free text of arbitrary length

### Subsection

medical information concerning that subsection  
free text of arbitrary length

Seite 1/20

hospital information, such as address etc.

**Figure 4.1:** An illustration of a typical out-patient card. Note that the number of pages varied between 1 and 90.

## 4.3 IBM Content Analytics with Enterprise Search (ICAwES)

*IBM Content Analytics* (ICA), formerly known as *IBM Cognos Content Analytics*, is intended to provide enterprises with tools to enable them to identify new revenue opportunities, improve customer satisfaction, and provide early problem detection. Although its full name is ICAwES, its mostly abbreviated as ICA. It follows the NLP idea of information extraction. In their publicly available *Redbook*, Zhu et al. (2011) describe ICA's key capabilities as follows:

- **Discover** new relationships in the content.
- **Refine** that content to provide business context by using search capabilities.
- **Deliver** new insight to the business users to enable focused decision making.

### 4.3.1 Terms and Definitions

To begin with, Zhu et al. (2011) introduce and describe the following terms and we will follow their definitions:

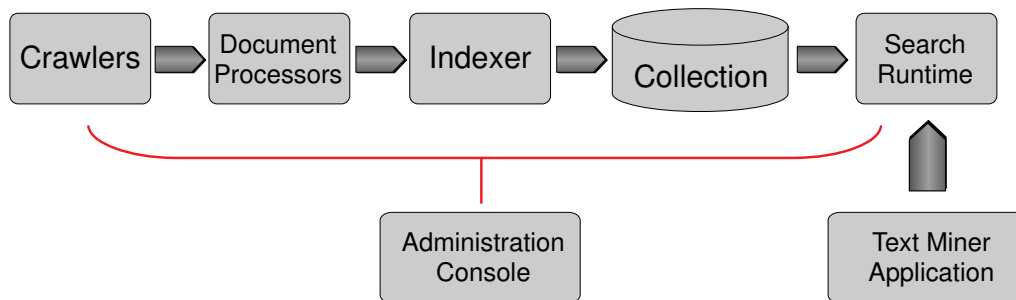
In the context of this work, the term *unstructured content* refers to free text. Following that definition, *text analytics* is the automatic process of converting unstructured content into structured information. A content analytics *collection*, or *document corpus*, is the entire group of documents available to the application for analytics purposes. Note that ICA supports multiple collections. The *facets* represent the different aspects or dimensions of the collection; They are a mechanism for navigation in the *text miner application*, which is the user interface. As we will see, facets can be populated with values obtained from either a structured or unstructured information source.

### 4.3.2 Architecture

Zhu et al. (2011) describe the six major components of ICA as depicted in the following paragraphs; We will essentially provide a summary of their work in this subsection. The interconnection between the various components can be seen in Fig. 4.2.

**Crawlers** extract textual data from various supported data sources. The intervals, at which the crawlers check for new or updated content, can be configured via the administration console. Out of the box, there are a number of different crawlers available for different data sources.

- Web-based crawlers
- Crawlers for IBM products as data source
- File systems crawlers



**Figure 4.2:** The component architecture of IBM Content Analytics according to Zhu et al. (2011)

- Relational database crawlers
- Email crawlers

Most crawlers can be set-up to crawl multiple data sources of the same type. It is also possible to write your own custom crawler is needed; This, however, requires programming experience, while using built-in crawlers does not.

**Document Processors** are responsible from processing the crawled documents, as well as preparing them for indexing. Each document processor is utilizing an UIMA conform pipeline of annotators; Some of those are build-in out of the box, while others can be plugged-in according to ones needs.

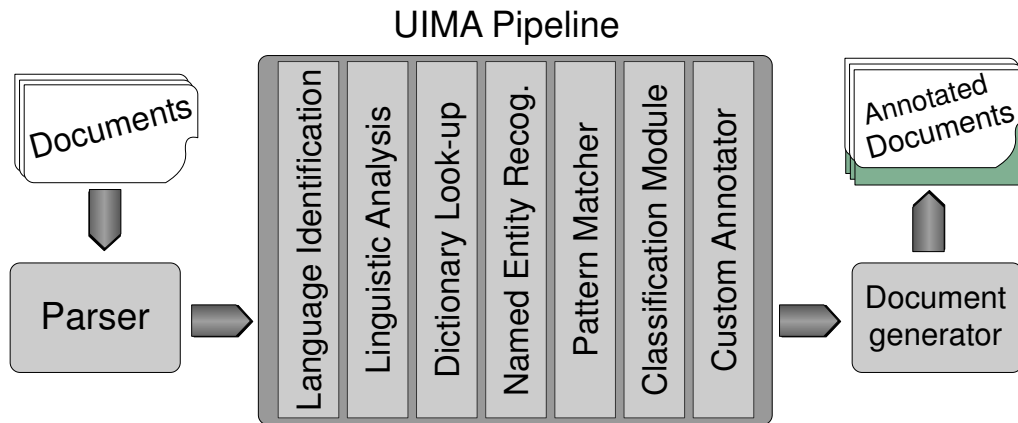
**Indexer** is responsible for building an optimized index of document content. It is based on the open source *Apache Lucene indexer*. When active, the index automatically indexes process documents. Manual rebuild of the index, among other options the administration console offers, is also supported.

**Search Runtime** is a server-based component responsible for servicing the client's search- and analytics-requests. These client service requests are made by using the Java-based programming interface called *IBM Search and Indexing API* (SIAPI); It operates remotely by using the HTTP/HTTPS protocol. Every collection is associated with at least one search runtime. It is possible to have more than one if needed.

It might be interesting to note, that in order to provide continuous operation, the search runtime operates on copies of their associated collection.

**Text Miner Application** This is the component that performs the text analytics. It provides a browser-based user interface that communicates with the text miner's web application; It runs under either Jetty or a WebSphere Application Server. The web application issues SIAPI client requests to the search run time.

**Administration Console** is - as the name suggests - the administrative component of the system. Through the console, you can create and administer collections, start and stop components, as well as monitor system activity and log files. It also provides security options, such as a user account system that can be configured.



**Figure 4.3:** The document processor architecture in IBM Content Analytics according to Zhu et al. (2011)

For the *Document Processor*, ICA utilizes a UIMA pipeline as depicted in Fig. 4.3 and described in the following paragraphs. This pipeline includes a set of fundamental annotators; Note that the first two can not be changed, while the other can be configured according to ones needs.

**Language Identification annotator** Identifies the language of the document. This fundamental information can be utilized to branch in specialized parsing rule sets. Note that the language identification is document wide. If the language is known beforehand for all documents, it can also be set manually.

**Linguistic Analysis annotator** Applies basic linguistic analysis, such as POS, to each document. Default tagging can in later stages be influenced and improved by defining own types, such as real numbers or dates, by means of character rules for the disambiguation of punctuations.

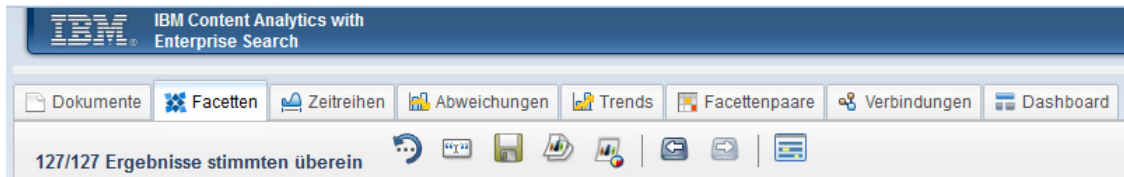
**Dictionary Lookup annotator** Matches words from dictionaries with words in the text. Note that stemming, as well as the definition of synonyms is supported. This stage can be configured in the administration console.

**Named Entity Recognition annotator** This annotator can only be activated or deactivated and not configured as of now. It extracts person names, locations and company names.

**Pattern Matcher annotator** Identifies those pattern in the text that are specified via rules.

**Classification Module annotator** Performs automatic classification. It uses natural language processing as well as semantic analysis algorithms to determine the true intent of words and phrases. It combines contextual statistical analysis with a rule-based, decision-making approach.

**Custom annotator** Custom annotators are essentially java programs that obey a given UIMA interface. This program has access to all annotations made by



**Figure 4.4:** Screenshot of the standard tabs of the browser-based Text Miner Application. Each tab represents and activates a different view on the data.

the previous annotators. It is very common that this stage is developed with *ICA Studio*.

The out-of-the-box text miner application provides the user with a number of different views and ways to interact with the annotated documents. The typical tabs associated with the views can be seen in Fig. 4.4.

The most important views in our project were the first three.

1. **Dokumente.** Shows the documents that fit the search query. Note that the search query can be generated by simply clicking through the data and annotations
2. **Facetten.** The facet view shows the facet tree generated by the developer. This tree is then filled accordingly with the extracted annotations of the files. Note that only the annotations of those documents that fit the search query are listed. This view is very useful for navigation.
3. **Zeitreihen.** Shows time plots. Note that any date can be used; even extracted ones.

## 4.4 IBM Content Analytics Studio

ICA Studio is an *Eclipse*-based (Eclipse Foundation, 2004) development suite for the creation of UIMA-conform annotators. The Studio does not depend on ICA itself, but as the name suggests it does work particularly well with ICA. This is mainly because of the build-in deployment functionality to ICA's *custom annotator* stage.

There are three basic means to create annotations:

- **Character rules** are based on regular expressions. This is useful to specify character patterns of interest such as dates or phone numbers. Those character sequences following these patterns can then be annotated and that way be used accordingly for relation extraction.

The syntax is based on JRegex, which is regular expression library for Java.

- **Dictionaries.** The creation of dictionaries in ICA is useful to create word classes. For example a dictionary “MonthNames” could be used to identify

mentioning of months within a text documents. The dictionaries also offer the possibility to associate other information with other features. For example “Oktober” could then be associated with “10” to in turn easily normalize date information.

When defining dictionaries, ICA Studio supports the manual definition of any custom morph or synonym, as well as providing the automatic generation of inflections. The canonical form is then defined as the *lemma*.

The dictionaries also allow the user to assign the part-of-speech to special words or word classes - if needed - which can be later exploited in the process of designing parsing rules.

For many supported languages, such as German and English, standard dictionaries are already build-in.

- **Parsing rules.** The modeling of relations can be done by defining parsing rules, that can operate on different levels, such as phrase or entity, as well as different scopes, such as sentence, paragraph or document. These parsing rules can also automatically be derived out of sample text passages and manually changed and optimized as needed, speeding up the process.

An interesting plug-in functionality in the ICA studio are the so called *normalizers*. They can be used to convert different formats of the same concept into one standardized format. For example: “12.10.1987” and “1987-10-12” describe the same concept - a date. The normalizers can also be used to overcome different points of reference, or units. For example: “100 pounds” could automatically be tagged with the normalized feature “45.359 kg”. It is also possible to plug-in your own custom normalizers created in Java.



## 5. Methods

In this section we discuss and document our project; how we approached it and how we used the materials at our disposal

### 5.1 Information of Interest to the Physician

In the beginning of our project, we sat down with the physician and discussed what kind of information would be of interest to him. To be more specific, we asked him what questions he would ask the computer about the content of the out-patient cards, if a computer were - for the sake of argument - able to completely understand the content of those documents.

In this process, we identified a few sample questions that indicate the physicians information interest:

- How many patients had a malign melanoma on their right upper arm?
- Does the patient have metastasis?
- What's the associated Breslow index (tumor thickness) of the melanoma?
- When are specific diagnosis common?
- What therapies does the patient have in place?

We analyzed those sample questions in order to derive the key information that need to be extracted from each document.

### 5.2 Naming Convention

The purpose of a naming convention is to allow the deduction of useful information from the component names. In this case, the components are the annotations. As mentioned before there are three basic ways of creating annotations: (1) via *character rules*, (2) via *dictionaries*, or (3) via *parsing rules*.

It turned out to be very helpful during development to have a way of quickly knowing the source of the annotation. We decided to use the following pre- and postfixes when creating annotations:

- **Dict-**. This prefix is used for every annotation created by dictionaries. For example, the annotation created by the dictionary containing the months of a year is called `DictMonth`.
- **Token-**. Character rules can have one of two prefixes. The prefix *Token-* is used when the character rules influence tokenization.
- **Char-**. On the other hand, if the character rules don't influence the tokenization the use the prefix *Char-*.
- **Raw-**. Parsing rules in general do not need a prefix. However, it is often very convenient to utilize an extra stage to fuse the annotations of various sources (e.g. character rules, dictionaries, ...) together and perform normalization. In that case, the pre-normalization annotation has the same name as the final annotation, but with the added prefix *Raw-*. For example `RawNumber`.
- **-Candidate**. This postfix indicates, that there is a high chance that the annotation might be a false positive. The context is investigated in the further steps of the pipeline; thus reducing or eliminating the uncertainty. Since text passages can be convert by more than one annotation, a false positive at this stage does not influence other interpretations of the same passage in any way. Naturally, this postfix can not occur in combination with the prefix *Token-*. For example `RawDateCandidate` for isolated (possible) year specifications such as "2006".

## 5.3 Designing an Annotation Pipeline

With the questions of interest identified, we started to design an annotation pipeline for ICA's custom annotator. This pipeline is of course able to access POS annotations out of the box, as it is build-in into the ICA's basic functionality. The pipeline of the custom annotator can be seen in Fig. 5.1.

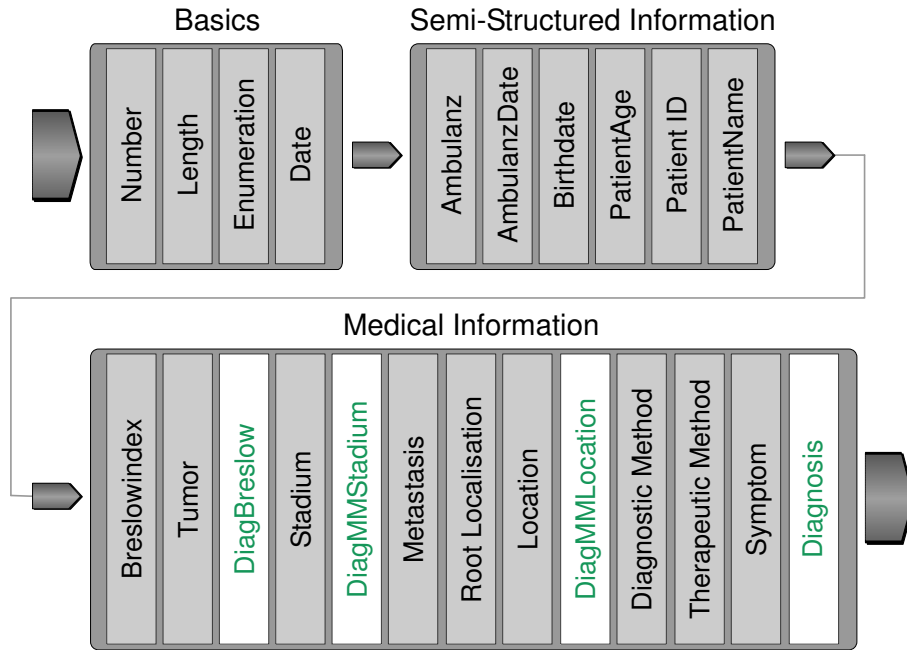
The implementation of it's annotators was mostly done with *ICA Studio 3.0*; the exceptions are one special annotator and three custom normalizer that were written with Eclipse in Java.

## 5.4 Custom Normalizer

In order to meet our goals we had to write three custom normalizer to compliment the functionality of ICA Studio 3.0.

### 5.4.1 FormulaSolver

By investigating the out-patient cards, we realized that we had to deal with numerical values a lot. For this purpose we implemented a custom normalizer called *FormulaSolver*.



**Figure 5.1:** An illustration of the custom pipeline design realized in ICA Studio. The green annotators with the white background are context sensitive; they filter out negations and suspicions.

The formula solver supports three value types:

1. **Numerical.** Any integer or real number. Examples: 2, 4.23, -1, -4.323
2. **Boolean.** Any boolean. Examples: true, false
3. **String.** Have to be enclosed in quotations. Examples: “test”, “any str”

Internally, the formula solver is a string interpreter, that tries to compute the string that you feed him as input and returns the result. A few typical examples can be seen in Table 5.1.

There are a number of different operators build-in. A list of all supported operators can be seen in Table 5.2.

**Table 5.1:** Some example inputs and the corresponding output that the formula solver would return.

Input	Output
$(6 + 2 * 2) / 2$	5.0
$2 * 3$	6.0
"abc" + "def"	abcdef
$3 > 2$	true
$2*4 \neq 16 / 2.0$	false
$-1.32 * -1$	1.32
$2^{(10)}$	1024
$(3.5 > 3) \ \&\& \ (3.5 < 4)$	true

**Table 5.2:** The build-in operators of the FormulaSolver.

Type	Operators
Logical operators	&&,   , ==, !=, >=, <=, >, <, !
Arithmetic operators	+, -, *, /, ^
Functions	abs, sqrt
Special	toFormat

The special `toFormat` operator is used as a wrapper for Java's `DecimalFormat` class. Its purpose is - as the name suggests - to format real numbers in a specified way. For example to limit the number of positions after the decimal point.

## 5.4.2 DistanceNormalizer

Understanding length measurements is very important when dealing with medical records from dermatology. Both, tumor thickness and invasion depth are specified in either millimeter or centimeters.

The distance normalizer is a tool written by us for ICA Studio, that can convert length measurements from one unit of distance to another. This is important for normalization, because the tumor thickness, for example, is sometimes specified in centimeter and sometimes in millimeter.

The supported units of the distance normalizer are: *mm*, *cm*, *m*, *km*, *inch*, and *feet*.

## 5.4.3 DateDifferenceCalculator

Naturally the age of a person changes over time. Thus a document containing the complete medical history of a patient holds various information that is associated with different aged versions of the patient.

To determine the patients age at specific points within the document two dates are needed: The birth date of the patient, and a date of reference.

Essentially the *DateDifferenceCalculator* is a custom normalizer for ICA Studio. Its functionality is pretty straight forward: It outputs the age difference between the two dates it gets as input.

## 5.5 Extract Basic Annotations to Build on

Quite early in our investigation, we realized that we needed a few key annotations to build on. These basic annotations are independent of the actual task and even unrelated to medicine.

Since everything else will depend on the precision and recall of those annotations, it was very important to devote much thought into their implementation. Those annotations include:

- **Reel Numbers.** Out of the box, the current version of ICA Studio (i.e. 3.0) does not recognize real numbers. A reel number in German is separated with a colon (e.g. “1,32”, “0,01235”, “123”, etc.).
- **Length Measurements.** We needed the recognition of reel numbers mostly for length measurements, such as the tumor thickness.
- **Enumerations.** Enumerations are generally important for date recognition in the German language. Although dates are often denoted in a standard numerical manner (e.g. “12.10.1987”), they can also be described with the use of enumerations.
- **Dates.** There are a lot of ways to denote dates; especially in the German language. Our goal was to recognize the most common ones and create annotations in a convenient format to operate on.

### 5.5.1 Annotation for (Reel) Numbers

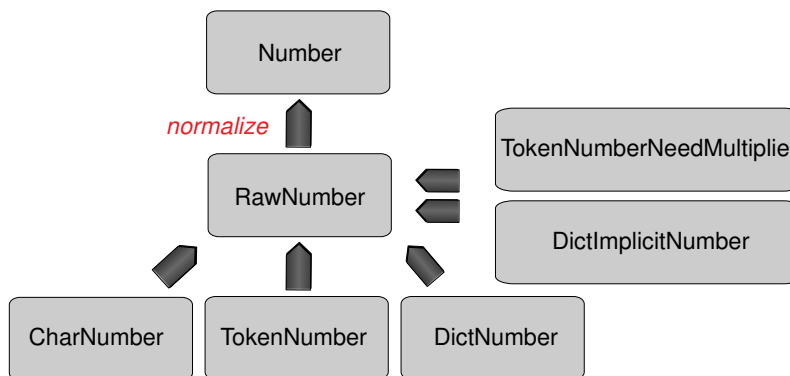
Natural and reel numbers can be denoted in fairly different ways. We decided to support the following (absolute positive) ways:

1. **Numeral.** The usual mathematical way, but with a colon as separator. e.g.: “1,32”, “0,01235”, “123”
2. **Textual.** Denotation with words. This is sometimes done for positive natural numbers. e.g.: “*fünf*” (meaning five), “*hundert*” (meaning hundred).
3. **Implicit.** It is quite common for medical professionals to denote smaller numbers implicitly. In our case we approximated them with the typical values suggested by the physician. e.g.: “*wenigen*” (meaning a few), “*ein paar*” (meaning a couple).
4. **With multiplier.** Rather rare in the medical field but quite common when dealing with amounts of money. e.g.: “1k” (meaning 1000)

We implemented 1. and 4. with the means of character rules. The annotations they create are called `CharNumber`, `TokenNumber`, and `TokenNumberNeedMultiplier` respectively.

Type 2. and 3. are realized with custom dictionaries. Their annotations are called `DictNumber` and `DictImplicitNumber`

All the five annotations above are then fused and normalized into one annotation called `Number` as depicted in Fig. 5.2. Its features are the *value* (with a dot as separator) and a boolean called *is\_integer* that is true if the value is indeed an integer.



**Figure 5.2:** The annotation pipeline for numbers. Note that the last step is for normalization.

## 5.5.2 Annotation for Length Measurements

The length measurements build on the number annotation. Essentially, if there is a number in front of a length unit it will be recognized as a length measure. The following example showcases a length measurement with some text.

Malignes Melanom interscapulär re (Invasionstiefe 1,3 mm) 5/2010

- ... Number annotation
- ... Unit of length: Millimeters

In some cases there is no space between the number and the unit, resulting in the tokenizer misinterpreting the string as just one token. This case is separately annotated via character rules as **TokenLength**.

Another special case rises from the way length is commonly described in natural language. The following example demonstrates a number representing a length measurement that can only be identified by context called **RawLengthByContext**:

... winzige, 1 bis 2 mm im DM haltende Verdichtungsbezirke ...

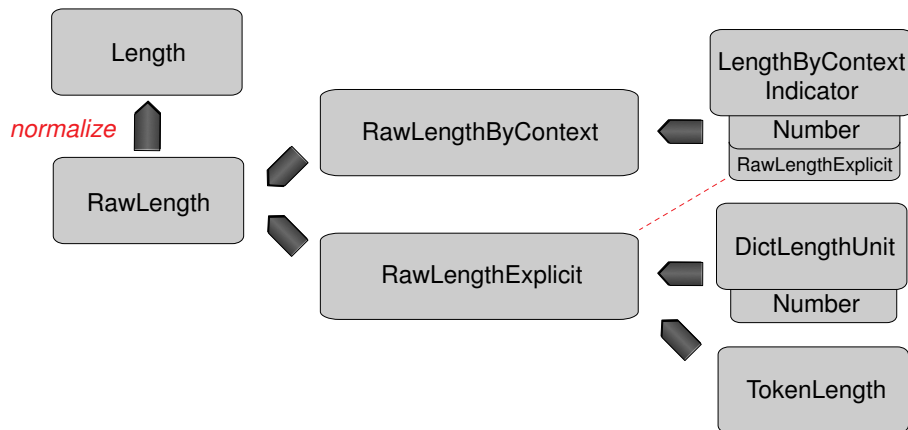
- ... Number annotation that indeed is a length measurement by context
- ... Length annotation defining the unit of length (in this case millimeter)
- ... Length by context indicator. Other examples are “auf” and “und”

All these annotations are fused into an annotation called **Length** as depicted in Fig. 5.3. Its features are the *value\_as\_mm*, *value\_as\_cm*, *value\_as\_m*; all created with the *DistanceNormalizer*.

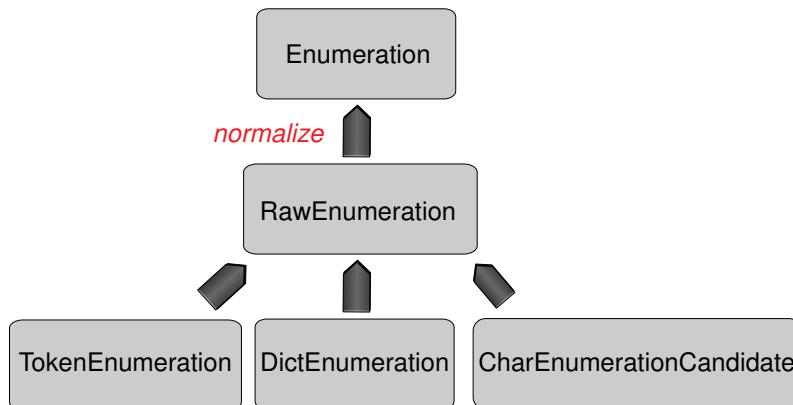
## 5.5.3 Annotation for Enumerations

Enumerations are fairly similar to numbers. The difference is the context. For example “2.”, “first” and “2nd” are all English enumerations. To be more precise the following ways of denoting enumerations are supported:

1. **Numeral.** The same as natural numbers. It is not clear if this indeed is an



**Figure 5.3:** The annotation pipeline for length measurements. Note that the last step is for normalization.



**Figure 5.4:** The annotation pipeline for enumerations. Note that the last step is for normalization.

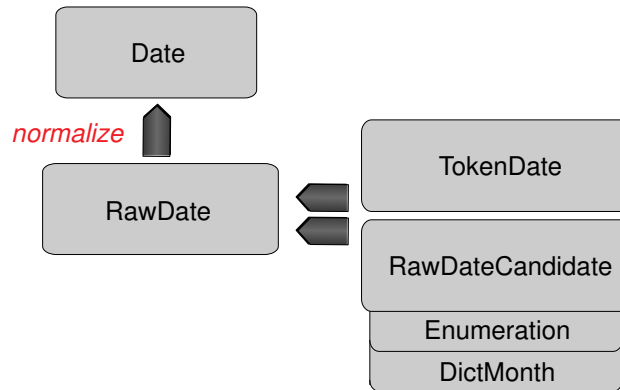
enumeration; there is uncertainty involved. e.g.: “2”, “5”, “7.”

2. **Explicit.** This means that the token can not be confused with anything else. e.g.: “3ter”, “15-ten”,
3. **Textual.** Denotation with words. e.g.: “erster” (meaning first), “vierter” (meaning fourth).

The annotation pipeline for **Enumeration** is depicted in Fig. 5.4. It’s features are *value*, *possible\_day*, and *possible\_month*. The last two are Booleans; they are true if the enumeration could possible denote a day or a month respectively. For example “13th” can not possible denote a month.

#### 5.5.4 Annotation for Dates

Correct recognition of dates is very important. Luckily its is uncommon for medical experts to denote the dates in a textual manner; so most dates were numerical and



**Figure 5.5:** The annotation pipeline for dates. Note that the last step is for normalization.

easy to annotate with character rules. To be more precise the following ways of notation are supported:

1. **Explicit.** This means that the token can not be confused with anything else. e.g.: “12.10.1987”, or “4/1987”. Note that in German dates are usually denoted dd.mm.yyy or mm/yyyy.
2. **Textual.** Denotation in combination with month names. e.g.: “April 2009”, or “Oktober 2012”.

The annotation pipeline can be seen in Fig. 5.5. The features of *Date* are *day*, *month*, *year*, *normalized*, as well as two Booleans *year\_explicit* (e.g. “19.02.14” is not explicit) and *year\_specified* (e.g. “2. April” has no year specified).

## 5.6 Extract Semi-Structured Information

Since our requirement was to use only the PDF versions of the out-patient cards as information source, we had to extract the actual patient information (i.e name, birth date, ID, ...) from the documents as well; normally those are available in a relational data base. Luckily, the way these information are stored within the documents is standardized. That fact made it quite easy to tailor simple parsing rules with a 100% recall and precision.

The following example indicates how the patients name and birth date were encoded on every page except the first one. The surrounding “separators” were used as triggers.

Patient: Mustermann, Max, 10.10.1910 ¶

- ... Surname of the patient
- ... First name of the patient
- ... Birth date of the patient
- ... Separator of the annotations



The patient ID was - strangely enough - not always present in the files. However, in those cases it was present it was also encoded easily recognizable.

The corresponding annotations are listed below:

- **PatientName**. Has the features *first\_name*, *last\_name*, and *full\_name*.
- **PatientID**. The unique ID of the patient. This annotation does not always exist, since not all files contain the patient ID. Has the feature *id*.
- **BirthDate**. The identified **Date** that represents the date of birth of the patient. It has the same features as **Date**.
- **PatientAge**. Every date of visit is associated with the according age of the patient at that time. This annotation is created with the help of the *DateDifferenceCalculator*. Its feature is *age*.

## 5.7 Extract Medical Information

The more challenging part of this work was the extraction of the medical information of interest. This is partly because of the various non-standardized multilingual abbreviations.





### 5.7.1 Annotation for Breslow's depth

The *Breslow's depth*, commonly referred to as *Breslow-index* or *Breslow-Level*, is used as a prognostic factor in melanoma of the skin as a description of the depth of the tumor. The system was named after its original creator *Alexander Breslow* (Breslow, 1970).

The Breslow's depth, if there is any, is usually specified in the diagnosis summary of a doctor's letter. The measurement itself is mostly abbreviated with "TD" followed by the actual value and its unit; the emphasis is on mostly, because there are variations. Still, some synonym of "tumor thickness" is always present. A typical example can be seen below:

St.p.	MM	,	TD	,	1,5 mm	,	OS li.	innen,	11/2010
-------	----	---	----	---	--------	---	--------	--------	---------

	...	Abbreviation for "Malign Melanoma", specifying the tumor
	...	Abbreviation for "Tumordicke" (meaning <i>tumor thickness</i> )
	...	Annotation of a length specification
	...	Annotation of a date specification

To begin with, we extract the actual measurement which results in the annotation **Breslowindex**.

If this annotation is in close proximity on the right side of a **Tumor** annotation of the *type* "Malignes Melanom", the annotation **DiagMMBreslow** is created. There are two variations of this annotation: (1) With or (2) without a specified date.

A date is present if the Breslow’s depth is specified in the sentence that states the tumor’s removal. In that case, the date is either at the beginning or at the end of the sentence. However, since the **Breslowindex** annotation requires an actual measurement of the tumor thickness to be present, the precision of the recognition was expected to be high either way (meaning that a false positive should be highly unlikely).

### 5.7.2 Annotation for the Stadium of a Malign Melanoma

The *stadium* of a malign melanoma is specified in a similar way; most of the time it is located in the diagnosis summary of a doctor’s letter. A typical example can be seen below:

```
Malignes Melanom Stadium IV (pT-4b, N-O, M-1b AJCC 2009)
B-RAF V600E
```

- ... Annotation for a tumor of the type “Malign Melanoma”
- ... Keyword used as trigger for a following stadium specification
- ... Character string or number specifying the stadium

Analogous to **Breslowindex** the stadium specification itself is extracted separately as **Stadium**. If this annotation is in close proximity on the right side of a **Tumor** annotation of *type* “Malign Melanoma”, it is extracted as **DiagMMStadium**. Note, that since an actual stadium value is specified, a false positive was expected to be highly unlikely.

### 5.7.3 Annotation for Location Information

There are various things in a doctor’s letter that can be associated with a location, such as tumors and performed diagnostic methods. However, before a location can be linked to another concept, it has to be annotated first.

We approached the annotation **Location** by detecting two basic specifications adjacent to each other; namely a *location* and a *direction*. These two annotations were realized with the use of dictionaries. A typical example of a complete location specification can be seen below:

```
Z.n. Melanom Knie rechts (Tumordicke 1,25mm, Level III) OP 8/2001
```

- ... DictLocation: Specification of a location. Associated with body part “Bein” (leg)
- ... DictDirection: Specification of a direction.

The dictionary **DictLocation** hold common words that specify a location on the human body; it also associates these words with other information, such as the body part and body system it belongs to (such as cardiovascular, lymphatic, or respiratory). For example: the location “lung” is part of the body part “chest” and belongs to the “respiratory system”.

Additional information can be provided via direction specification. Note that a direction does not need to be specified for the location to be recognized. Common

words specifying a direction, such as “left” or “right”, are stored in the dictionary `DictDirection`.

The annotation `Location` has the features *body\_part*, *direction*, *payload* (which is the lemma of the recognized word), and *normalized* (with is the fusion of location and direction).

#### 5.7.4 Annotation for the Location of a Malign Melanoma

If a `Tumor` of type “Malign Melanoma” is close to a `Location` it will be annotated as `DiagMMLocation`. A typical sentence specifying the location of a melanoma can be seen below:

Z.n. Melanom Knie rechts (Tumordicke 1,25mm, Level III) OP 8/2001

- ... Annotation of a tumor of type “Malign Melanoma”.
- ... Annotation of a complete location.

The features of `DiagMMLocation` are *type* (which is inherited from `Tumor`), *body\_part*, *direction*, and *normalized*.

#### 5.7.5 Annotation for Diagnostic Methods

A performed diagnostic method is always linked to a location. An example would be an MRI of the head. However, some locations are implied by the method, such as a electrocardiography, which is always associated with the heart.

A problem of the German language, however, is that it is very common to fuse words together. A typical example of this would be the word “Oberbauchsono”, which is the union of the concepts *sonography* and *upper abdomen*.

To realize this annotation despite the obstacles, we used two dictionaries called `DictDiagnosticMethods` for the methods alone, and `DictDiagnosticMethodsLocation` for commonly fused words.

If a `DictDiagnosticMethods` annotation is very close to a `Location` annotation or the method is self-describing (such as an ECG), it is recognized as `DiagnosticMethod`. A typical example of how a performed diagnostic method is documented can be seen below:

Radiologischer Befund: Dr. Mustermann Nirvana vom 23.10.2012  
CT Schädel: unauffällig. Falxmeningeom, playueförmig,  
 rostralseitig, 2x 1 cm groß

- ... Abbreviation for computed tomography; a diagnostic method
- ... Annotation of a location.

On the other hand, a `DictDiagnosticMethodsLocation` annotation is also treated as self-describing and annotated accordingly.

The last annotation `DiagnosticMethod` has the features *method*, *body\_part*, *location*, and *direction*.

### 5.7.6 Annotation for Therapeutic Methods

Therapeutic methods will be the focus of the follow-up project, and thus received very little attention. They are annotated using a dictionary called `DictTherapeuticMethods` and result in the annotation `TherapeuticMethod` without any context check. This means that negations and suggestions of therapeutic methods get wrongly recognized as well. Its features are *type*, and *category*.

### 5.7.7 Annotation for Symptoms and Medication

Symptoms and Medication are annotated through a dictionary called `DictSymptom` and `DictMedication` respectively. As of now, there is no check in place to recognize negations. The presence of a word results in the annotations `Symptom` and `Medication`.

### 5.7.8 Annotation for Diagnosis

The annotation `Diagnosis` is created to later provide an overview in ICA. Its feature is called *type* and describes the actual diagnosis; for example “Malign Melanoma” or “Metastasis”.

For example: a malign melanoma is annotated as a diagnosis, if either a *tumor depth*, a *stadium*, or a *location* of a malign melanoma was identified in the file.

## 6. Results

In this section we will present the results of this project. First we will show the final implementation from a user's point of view. After that, we will state the quality of the discussed main annotations.

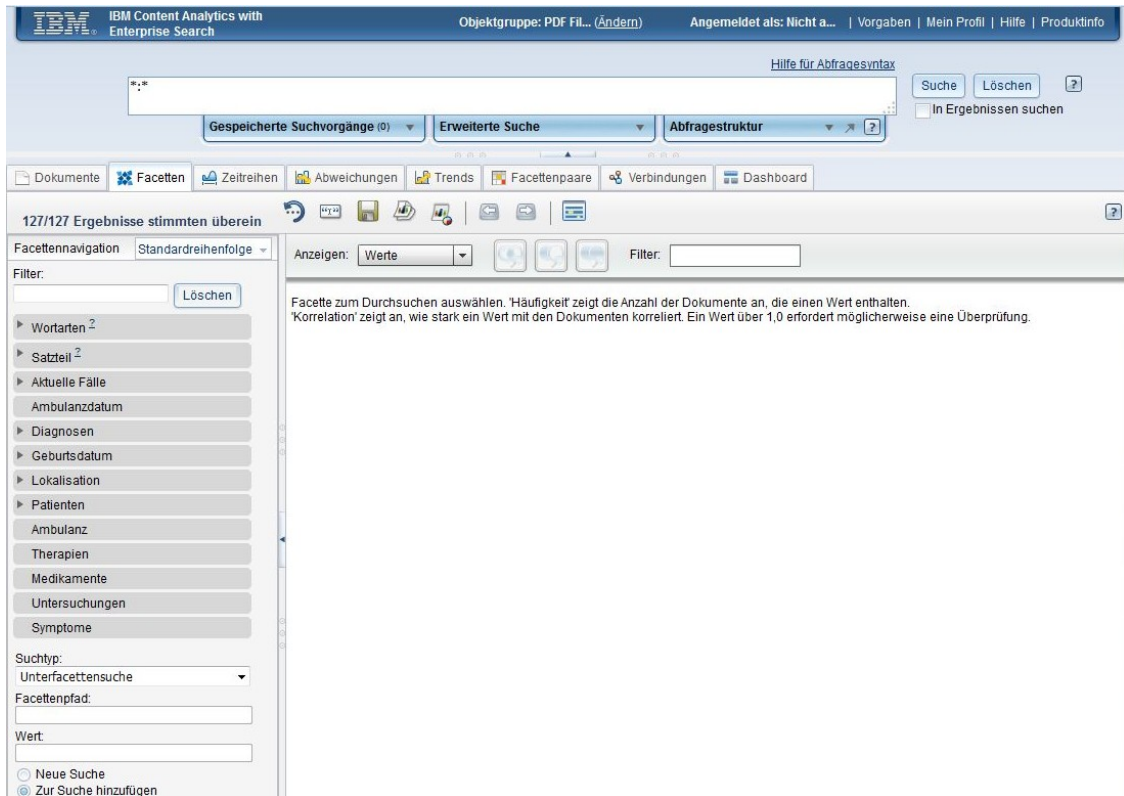
For the duration of this project, we've had a total of 127 unique out-patient cards at our disposal; this corresponds 1:1 to unique patients. It took a total of 6 hours to annotate all documents.

### 6.1 User Interface

To enable the user to navigate the collection, we build a facet tree for ICA according to the information of interest to the physician. A screenshot of the text miner application in the facet view can be seen in Fig. 6.1.

There are eleven main facets; some of which contain multiple child facets.

- **Aktuelle Fälle.** The first facet serves as navigation assistance for current cases. It is divided in *today*, *this week*, *this month*, and *this year*. The associated dates are taken from the "Ambulanzdatum" facet.
- **Ambulanzdatum.** Displays all the dates where one or more patients were at the hospital, clinic, or other in the files mentioned institution.
- **Diagnosen.** The diagnosis facet itself displays all the possible diagnosis and the number of patients where such a diagnosis was performed. Its sub-facets are "Malignes Melanom" and "Metastasen".
- **Geburtsdatum.** It displays all the birth dates of the patients. This facet is divided into ten year intervals, i.e. 10 years ago, 20 years ago, up to 90 or more years ago.
- **Lokalisation.** The localization facet displays all body parts and body systems with the associated number of patients that have had something performed or diagnosed at that location. The catalog for localization is based on NHUMI.
- **Patienten.** Displays the semi-structured information about the patients. It is divided in *first name*, *last name*, *id*, and *age*.
- **Ambulanz.** Also concerned with semi-structured information about the departments. This facet lists all the departments with the number of patients that visited the department at least once.



**Figure 6.1:** Screenshot of the text miner application’s facet view. The facet tree is located on the left side of the screen. Selecting a facet results in the corresponding annotations to be displayed on the right side. These annotations can be selected and their presence in a document made obligatory; this is a way to limit the documents to those of interest to the user. You can see the number of documents that fit the currently selected requirements near the top left corner.

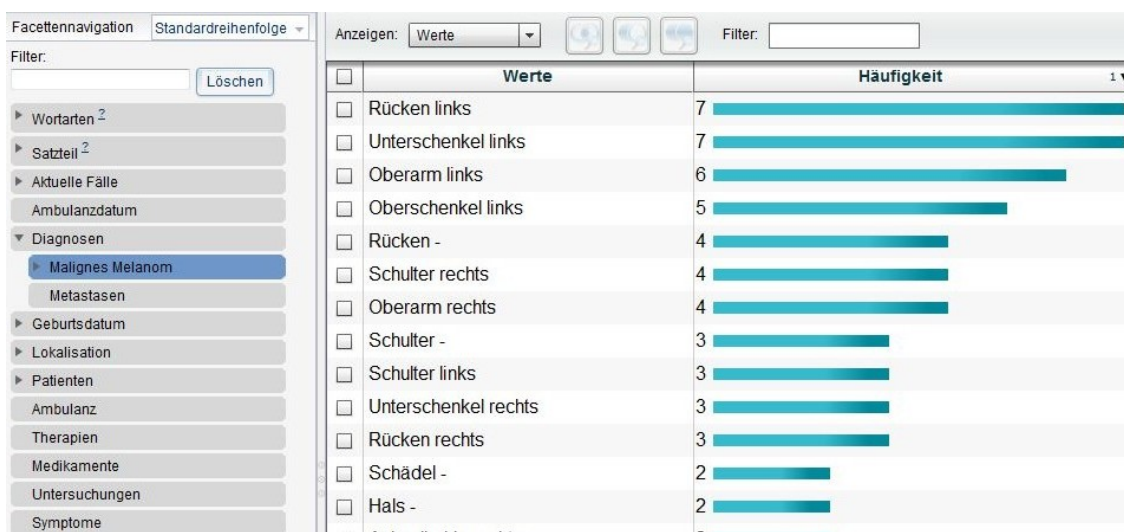
- **Therapien.** Displays all extracted therapies and therapy categories.
- **Medikamente.** This facet lists all medicaments with the number of patient files in which the medicament was mentioned. It is a simple dictionary match.
- **Untersuchungen.** The facet displaying the diagnostic methods with the number of patient that had the method performed at least once.
- **Symptome.** This facet lists all symptoms with the number of patient files in which the symptom was mentioned. Just like the medicaments facet, it is a simple dictionary match.

As mentioned before, we extracted two possible diagnosis. Each of which can be present on its own, not at all, or in combination with the other; these diagnosis are *malign melanoma* and the existence of *metastasis*. A screenshot of the selected facet can be seen in Fig. 6.2.

The diagnosis facet has two child-facets, namely “Malignes Melanom” and “Metastasen”. When selecting the *malign melanoma* facet, all the identified locations of



**Figure 6.2:** Screenshot of the diagnosis facet. It displays the supported diagnosis with the number of patients where such a diagnosis is present.



**Figure 6.3:** Screenshot of the malignant melanoma facet; a sub-facet of diagnosis. It displays the identified locations of the malignant melanoma found in the patient files, and the number of patients where such a diagnosis is present.

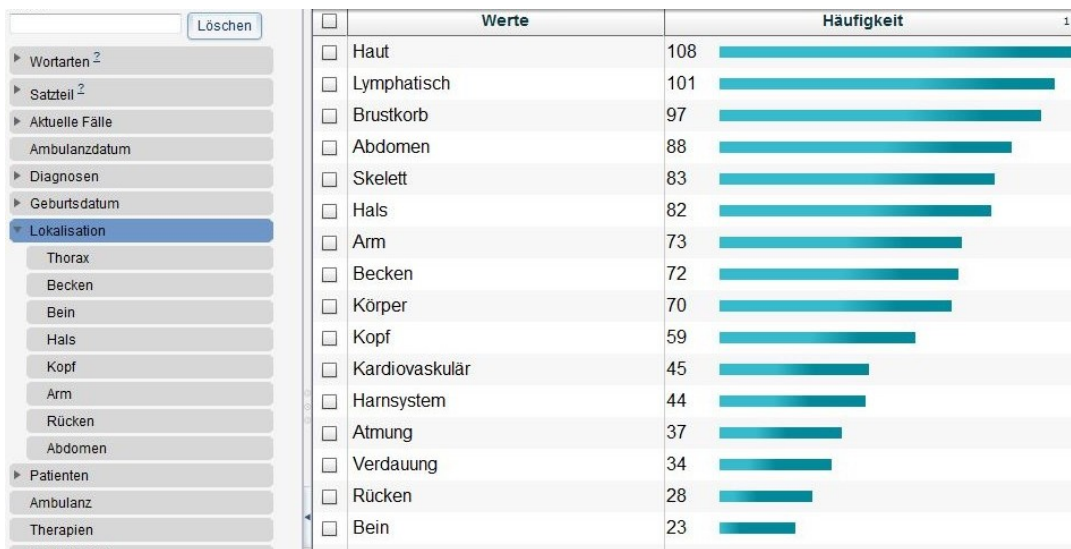
melanomas (limited to the currently available patients) are displayed (See Fig. 6.3). The melanoma facet has three sub-facets for *tumor depth*, *diagnosis date*, and *stadium*.

Selecting the metastasis facet results in the listing of all the identified metastasis. However only the location of specific metastasis are known. A screenshot of the facet and the possible metastasis locations can be seen in Fig. 6.4.

The localization facet shows all the body parts and body systems with the associated number of patients that have had something performed or diagnosed at that location. There are eight basic body parts; namely *head*, *neck*, *thorax*, *back*, *abdomen*, *pelvis*, *arm*, and *leg*. The body systems contain the *cardiovascular*-, *lymphatic*-, *respiratory*-, *urinary*-, and *digestive*- system. Each body part has its own sub-facet that displays the information associated with that particular part. A screenshot of the localization facet can be seen in Fig. 6.5, while Fig. 6.6 illustrates the information associated with the *head* sub-facet.



**Figure 6.4:** Screenshot of the metastasis facet; a sub-facet of diagnosis. It shows the identified metastasis with the number of patients where such a diagnosis is mentioned.



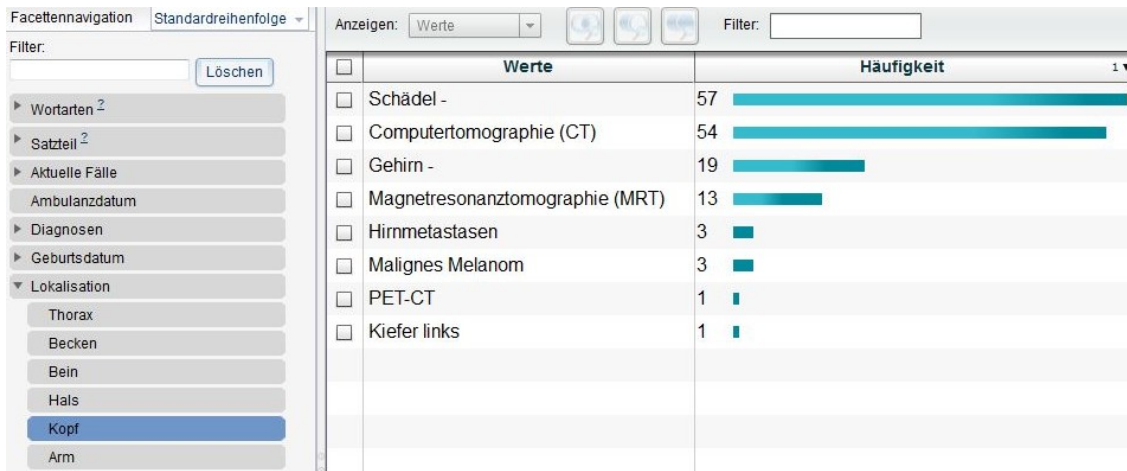
**Figure 6.5:** Screenshot of the localization facet. It displays all the body parts and body systems. The digits on the right show the number of patient files where something that belongs to the associated part or system is mentioned. Note that each body part has its own sub-facet.

The facets can also be used to perform correlation analysis. This can be done by switching to the correlation view and selecting the two facets of interest. Figure 6.7 illustrates a correlation analysis between the location of a malignant melanoma and the existence of various metastasis.

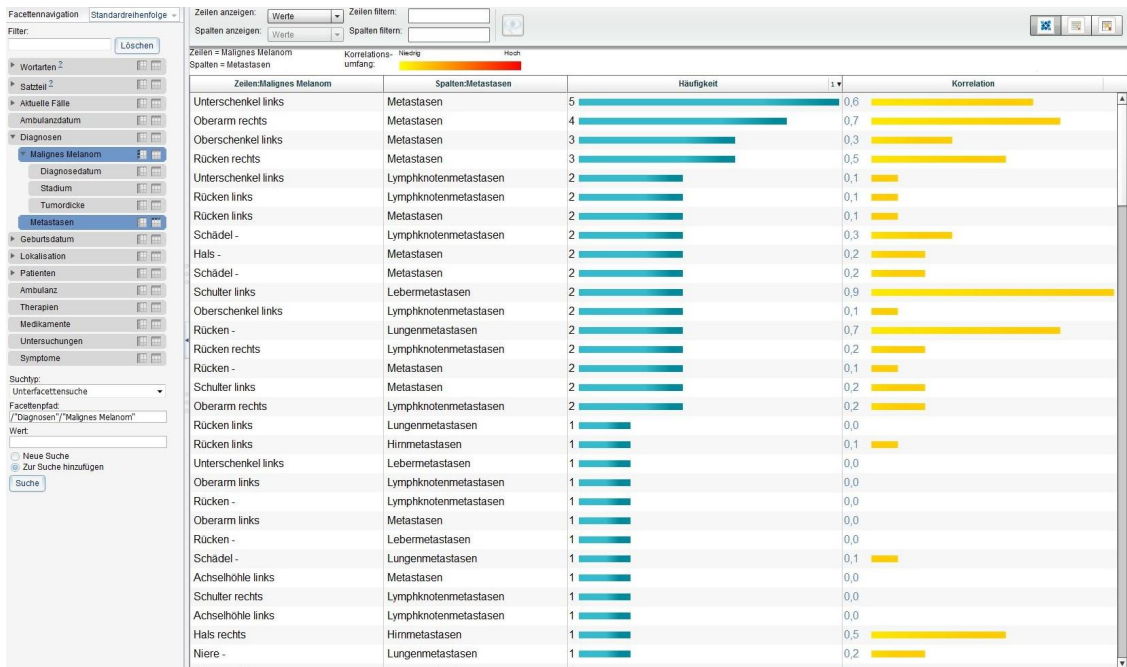
Another potentially interesting view on the data are the time plots. Figure 6.8, for example, shows a plot that demonstrates the number of patients each year that have at least one malignant melanoma diagnosed.

Similar to the time plots are the deviation plots. Figure 6.8 shows a deviation plot that demonstrates the number of patients over their birth year that received the corresponding diagnosis at least once in their life.

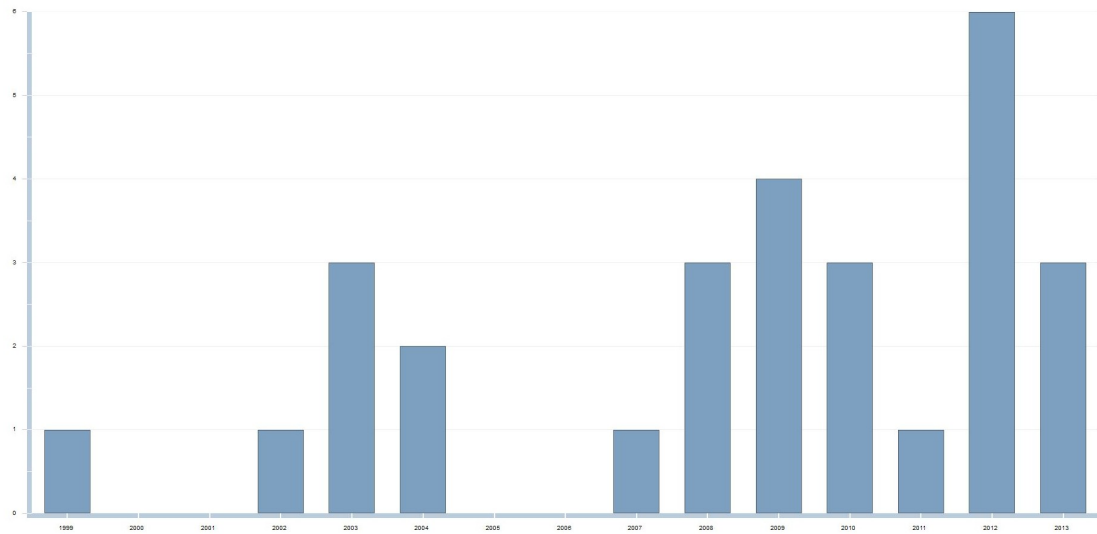




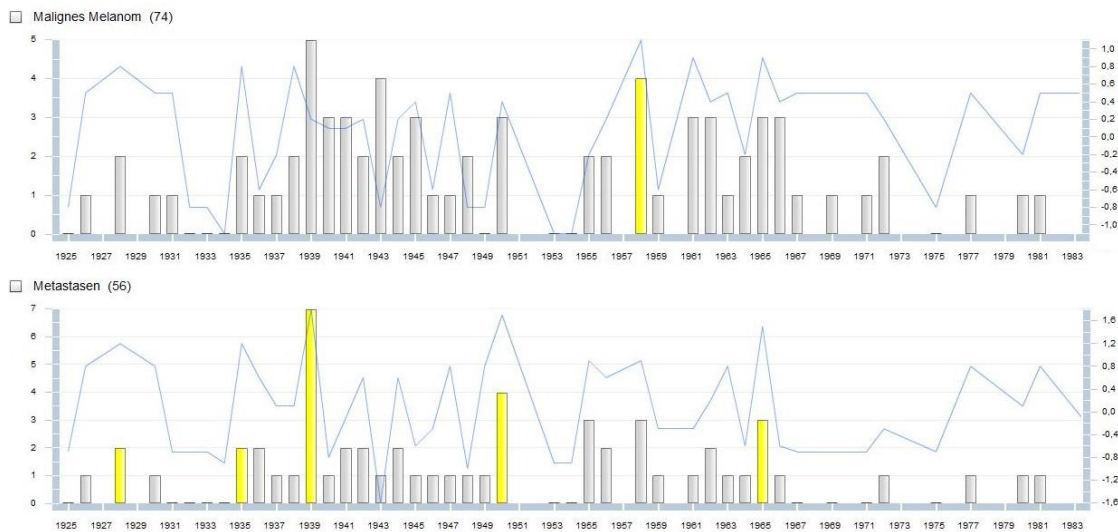
**Figure 6.6:** Screenshot of the facet for the head; a sub-facet of localization. It displays the information associated with the head of a patient, e.g. brain metastasis, MRI of the skull, etc.



**Figure 6.7:** Screenshot of a correlation analysis between the location of a malignant melanoma and the existence of various metastasis.



**Figure 6.8:** Screenshot of a time plot that displays the number of patients over the years where they had a malign melanoma diagnosed.



**Figure 6.9:** Screenshot of a deviation plot. This particular example displays the number of patients over their year of birth that received the corresponding diagnosis at least once in their life.

**Table 6.1:** The precision and recall of the key annotations.

Annotation	Precision	Recall
Diagnosis: Malign Melanoma	97 %	85 %
Diagnosis: Metastasis	91 %	81 %
Malign Melanoma: Breslow’s depth	95 %	89 %
Malign Melanoma: Location	86 %	78 %

## 6.2 Annotation Quality

We tested the annotation pipeline on 127 documents, corresponding to 127 unique patients. The quality of the key annotations, measured as *precision* and *recall*, were calculated by comparing the results against those of a human. To be more precise, 10 of the 127 documents were analyzed at the beginning of the project while the remaining 117 documents were analyzed **after** the computational results were established; this was done in order to reduce the bias of the rule sets. An overview of the results can be seen in Tab. 6.1.

First of all, the annotation of the “semi-structured” information (e.g. patient information) was tailored to the standardized document layout, resulting in a precision and recall of both 100 %. This, however, is not to be considered as a meaningful result, as it is to be expected.

The detection of patients, that had a malign melanoma diagnosed at some point in their life, yielded a precision of 97 % with a recall of 85 %. The tumor depth (Breslow’s depth) annotation yielded a precision of 95 % with a recall of 89 %. Lower results were obtained for the annotation of the malign melanoma’s location, with a precision of 86 % and a recall of only 78 %.

In contrast to malign melanoma, the annotation rules for metastasis detection are not context sensitive. This means that every occurrence of a word (or its inflections) that is present in the metastasis dictionary was annotated as a diagnosed metastasis. The detection of diagnosed metastasis yielded a precision of 91 % with a recall of also 91 %. Interestingly, the whole reason that the recall was not 100 % in that case, is that the word “Mikrometastase” was not present in the dictionary. The low precision is of course a result of the missing context sensitivity.



## 7. Discussion and Lessons Learned

The idea and motivation behind this work was to create a prove of concept that tailored solutions to the (bio)medical domain could be realized in a realistic time frame and with sufficient quality.

The point of this particular project was to investigate the potential of *IBM Content Analytics* in the biomedical field; a domain that presents a lot of obstacles for computational linguistics. As briefly mentioned before, those main obstacles are (a) the weak grammatical structure, (b) the mixture of languages, and (c) the large portion of (non-standardized) abbreviations.

Realizing an annotation scheme is an iterative task. Even though we as humans know which information is important to us (e.g. the type and length of a therapy), formulating a rule to identify such information is not a trivial undertaking.

Concerning ICA, we feel that the simple and easy-to-use interface enables the developers to perform fast prototyping. The on-the-fly testing gives you fast feedback of the “quality” of your rules. On the other hand, a lot of basic - and needed - functionality, such as POS tagging and named entity identification, is already build in. The supported scalability for the document collection and the supported parallelization for the document processor’s pipeline are also crucial.

In order to indicate and to some extend demonstrate that ICA enables the developer to quickly tailor quality annotation schemes to a specific domain, we calculated the precision and recall of the key annotations. Even though we are happy with those results, there are a few shortcomings and grey areas of this project worth discussing.

Firstly, we put our main focus on the detection of information concerning malign melanoma. This resulted in other conditions and diagnosis to be ignored completely. However, since our goal was to test ICA’s potential in the medical field and not to implement a marketable product, this kind of shortcoming was to be expected.

The reason for the low quality of the location annotation for a malign melanoma, is that the location was linked to a malign melanoma by proximity; as it turns out there are quite a few cases where that was wrongly considered to imply a connection between the location specification and the melanoma.

A gray area is the interpretation of the concrete numerical annotation quality results. Of course, rule-based annotation schemes perform well when tailored to a small document collection. We did, however, design the rule-sets as generic as possible to allow for scalability. It might be worth mentioning, that the recall of the melanoma detection would be well above 90 % if the dictionaries containing the words that describe a location (such as *hand* or *finger*) had a few more entries. The reason for the lack of dictionary entries is that the human analysis of the whole

document collection was done after the computational results were known; which we did on purpose to reduce the bias of the results.

## 8. Conclusions

Given its complex nature, general solutions to text understanding are not available yet, however, the business need is here now. This problem can meanwhile only be successfully addressed with specialized approaches, such as rule-based or statistical annotation schemes, to the domain or customer needs. This in return shifts the need to enable developers to quickly develop and test these annotation schemes. By speeding up the development process and providing rapid feedback, the developer can focus more time into building and improving the annotation schemes themselves. Because of the many possible solutions to a problem, where most of them are imprecise, the quality of the annotations strongly depend on the skills of the developer formulating them.

Our personal experience suggests that ICA is able to provide the necessary flexibility and modularity to efficiently tailor solutions for the purpose of knowledge discovery from unstructured biomedical data. To clarify: ICA/UIMA is not intended as a substitute to *MATLAB* when researching and testing methods, but as a means to test those methods in combination with human intelligence. This is aligned with our vision of HCI-KDD.





## 9. Future Work

Since computational linguistics is still far from achieving natural language *understanding*, there are seemingly endless possibilities for future research. Our intentions are twofold.

Now that we identified ICA and UIMA as a flexible and modular backbone with an adequately user friendly client application, we would like to shift our attention to the investigation of new methods for the purpose of NLP.

The (semi-)automatic identification of similar patient cases, for example, is an important and interesting topic. In this work, we addressed this problem by heavily including the human in the loop and providing him/her with navigational tools to select the conditions that should be present in the patient file. On the other hand, the work of Wagner et al. (2012) raised our interest in the application of topological methods to identify similarities within a collection of text documents. Thus we will move forward, build on our current ICA/UIMA backbone and are eager to investigate computational topology for knowledge discovery from biomedical text documents.

Concerning a successor to this work: The physician we worked with shares our enthusiasm about ICA's potential for the medical field (Holzinger, Stocker, Ofner, Prohaska, Brabenetz and Hofmann-Wellenhof, 2013). With this prove-of-concept project concluded, we would like to pursue a longer project with more resources in order to realize a solution intended as prototype for future clinical practice.



# List of Figures

2.1	Data, information, and Knowledge. The agent and its interaction with the world, as depicted by Boisot and Canals (2004). . . . .	36
4.1	An illustration of a typical out-patient card. Note that the number of pages varied between 1 and 90. . . . .	49
4.2	The component architecture of IBM Content Analytics according to Zhu et al. (2011) . . . . .	51
4.3	The document processor architecture in IBM Content Analytics according to Zhu et al. (2011) . . . . .	52
4.4	Screenshot of the standard tabs of the browser-based Text Miner Application. Each tab represents and activates a different view on the data. . . . .	53
5.1	An illustration of the custom pipeline design realized in ICA Studio. The green annotators with the white background are context sensitive; they filter out negations and suspicions. . . . .	57
5.2	The annotation pipeline for numbers. Note that the last step is for normalization. . . . .	60
5.3	The annotation pipeline for length measurements. Note that the last step is for normalization. . . . .	61
5.4	The annotation pipeline for enumerations. Note that the last step is for normalization. . . . .	61
5.5	The annotation pipeline for dates. Note that the last step is for normalization. . . . .	62
6.1	Screenshot of the text miner application’s facet view. The facet tree is located on the left side of the screen. Selecting a facet results in the corresponding annotations to be displayed on the right side. These annotations can be selected and their presence in a document made obligatory; this is a way to limit the documents to those of interest to the user. You can see the number of documents that fit the currently selected requirements near the top left corner. . . . .	68
6.2	Screenshot of the diagnosis facet. It displays the supported diagnosis with the number of patients where such a diagnosis is present. . . . .	69

6.3	Screenshot of the malign melanoma facet; a sub-facet of diagnosis. It displays the identified locations of the malign melanoma found in the patient files, and the number of patients where such a diagnosis is present. . . . .	69
6.4	Screenshot of the metastasis facet; a sub-facet of diagnosis. It show the identified metastasis with the number of patients where such a diagnosis is mentioned. . . . .	70
6.5	Screenshot of the localization facet. It displays all the body parts and body systems. The digits on the right show the number of patient files where something that belongs to the associated part or system is mentioned. Note that each body part has its own sub-facet. . . . .	70
6.6	Screenshot of the facet for the head; a sub-facet of localization. It displays the information associated with the head of a patient, e.g. brain metastasis, MRI of the skull, etc. . . . .	71
6.7	Screenshot of a correlation analysis between the location of a malign melanoma and the existence of various metastasis. . . . .	71
6.8	Screenshot of a time plot that displays the number of patients over the years where they had a malign melanoma diagnosed. . . . .	72
6.9	Screenshot of a deviation plot. This particular example displays the number of patients over their year of birth that received the corresponding diagnosis at least once in their life. . . . .	72

## List of Tables

5.1	Some example inputs and the corresponding output that the formula solver would return. . . . .	57
5.2	The build-in operators of the FormulaSolver. . . . .	58
6.1	The precision and recall of the key annotations. . . . .	73



## References

- Balter, M. (2000), ‘Archaeology: Paintings in italian cave may be oldest yet.’, *Science* **290**(5491), 419–21.
- Beale, R. (2007), ‘Supporting serendipity: Using ambient intelligence to augment user exploration for data mining and web browsing’, *Int. J. Hum.-Comput. Stud.* **65**(5), 421–433.
- Blandford, A. and Attfield, S. (2010), ‘Interacting with information’, *Synthesis Lectures on Human-Centered Informatics* **3**(1), 1–99.
- Boisot, M. and Canals, A. (2004), ‘Data, information and knowledge: have we got it right?’, *IN3 Working Paper Series* **0**(4).
- Breslow, A. (1970), ‘Thickness, cross-sectional areas and depth of invasion in the prognosis of cutaneous melanoma’, *Annals of Surgery* **172**(5), 902–908.
- Chomsky, N. (1956), ‘Three models for the description of language’, *IRE Transactions on Information Theory* **2**, 113–124.
- Chute, C. G., Pathak, J., Savova, G. K., Bailey, K. R., Schor, M. I., Hart, L. A., Beebe, C. E. and Huff, S. M. (2011), ‘The SHARPN project on secondary use of Electronic Medical Record data: progress, plans, and possibilities.’, *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium 2011*, 248–256.
- Clark, A., Fox, C. and Lappin, S., eds (2010), *The Handbook of Computational Linguistics and Natural Language Processing*, Blackwell Handbooks in Linguistics, John Wiley & Sons.
- Cornet, R. and de Keizer, N. (2008), ‘Forty years of SNOMED: a literature review.’, *BMC medical informatics and decision making* **8 Suppl 1**.
- Dolin, R. H., Alschuler, L., Boyer, S., Beebe, C., Behlen, F. M., Biron, P. V. and Shvo, A. S. (2006), ‘HI7 clinical document architecture, release 2’, *Journal of the American Medical Informatics Association* **13**(1), 30–39.
- Eclipse Foundation, T. (2004), ‘Eclipse - the eclipse foundation open source community website’, <http://www.eclipse.org>. Last visit: 15.08.2013.
- Erhardt, R. A. A., Schneider, R. and Blaschke, C. (2006), ‘Status of text-mining techniques applied to biomedical text’, *Drug Discovery Today* **11**(7-8), 315–325.

- Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996), ‘The kdd process for extracting useful knowledge from volumes of data’, *Commun. ACM* **39**(11), 27–34.
- Ferrucci, D. and Lally, A. (2004a), ‘Building an example application with the unstructured information management architecture’, *IBM Systems Journal* **43**(3), 455–475.
- Ferrucci, D. and Lally, A. (2004b), ‘Uima: an architectural approach to unstructured information processing in the corporate research environment’, *Nat. Lang. Eng.* **10**(3-4), 327–348.
- Ferrucci, D., Lally, A., Verspoor, K. and Nyberg, E. (2009), Unstructured Information Management Architecture (UIMA) Version 1.0, OASIS Standard, Technical report, OASIS.
- Funk, P. and Xiong, N. (2006), ‘Case-based reasoning and knowledge discovery in medical applications with time series’, *Computational Intelligence* **22**(3-4), 238–253.
- Garvin, J. H., DuVall, S. L., South, B. R., Bray, B. E., Bolton, D., Heavirland, J., Pickard, S., Heidenreich, P., Shen, S. Y., Weir, C., Samore, M. and Goldstein, M. K. (2012), ‘Automated extraction of ejection fraction for quality measurement using regular expressions in unstructured information management architecture (uima) for heart failure’, *Journal of the American Medical Informatics Association* **19**(5), 859–866.
- Gotz, T. and Suhre, O. (2004), ‘Design and implementation of the uima common analysis system’, *IBM Systems Journal* **43**(3), 476–489.
- Gregory, J., Mattison, J. E. and Linde, C. (1995), ‘Naming notes - transitions from free-text to structured entry’, *Methods of Information in Medicine* **34**(1-2), 57–67.
- Gunter, D. T. and Terry, P. N. (2005), ‘The emergence of national electronic health record architectures in the united states and australia: Models, costs, and questions’, *J Med Internet Res* **7**(1), e3.
- Hauser, M. D., Chomsky, N. and Fitch, W. T. (2002), ‘The faculty of language: What is it, who has it, and how did it evolve?’, *Science* **298**(5598), 1569–1579.
- Holzinger, A. (2012), ‘On knowledge discovery and interactive intelligent visualization of biomedical data - challenges in human-computer interaction & biomedical informatics’.
- Holzinger, A. (2013), *Human-Computer Interaction & Knowledge Discovery (HCI-KDD): What is the benefit of bringing those two fields to work together?*, Springer, Heidelberg, Berlin, New York, pp. 319–328.
- Holzinger, A., Geierhofer, R., Modritscher, F. and Tatzl, R. (2008), ‘Semantic information in medical information systems: Utilization of text mining techniques to analyze medical diagnoses’, *Journal of Universal Computer Science* **14**(22), 3781–3795.



- Holzinger, A., Kainz, A., Gell, G., Brunold, M. and Maurer, H. (2000), Interactive computer assisted formulation of retrieval requests for a medical information system using an intelligent tutoring system, *in* ‘World Conference on Educational Multimedia, Hypermedia and Telecommunications’, Charlottesville (VA): AACE, pp. 431–436.
- Holzinger, A., Scherer, R., Seeber, M., Wagner, J. and Müller-Putz, G. (2012), *Computational Sensemaking on Examples of Knowledge Discovery from Neuroscience Data: Towards Enhancing Stroke Rehabilitation*, Vol. 7451, Springer, Heidelberg, New York, pp. 166–168.
- Holzinger, A., Simonic, K. and Yildirim, P. (2012), Disease-disease relationships for rheumatic diseases: Web-based biomedical textmining and knowledge discovery to assist medical decision making, *in* ‘36th International Conference on Computer Software and Applications COMPSAC’, IEEE, Izmir, Turkey, pp. 573–580.
- Holzinger, A., Stocker, C., Ofner, B., Prohaska, G., Brabenetz, A. and Hofmann-Wellenhof, R. (2013), Combining hci, natural language processing, and knowledge discovery - potential of ibm content analytics as an assistive technology in the biomedical field, *in* A. Holzinger and G. Pasi, eds, ‘Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data’, Vol. 7947 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 13–24.
- Holzinger, A., Yildirim, P., Geier, M. and Simonic, K.-M. (2013), *Quality-based knowledge discovery from medical text on the Web Example of computational methods in Web intelligence*, Springer, Heidelberg, New York, pp. 145–158.
- Kano, Y., Jr., W. A. B., McCrohon, L., Ananiadou, S., Cohen, K. B., Hunter, L. and Tsujii, J. (2009), ‘U-compare: share and compare text mining tools with uima.’, *Bioinformatics* **25**(15), 1997–1998.
- Kreuzthaler, M., Bloice, M., Faulstich, L., Simonic, K. and Holzinger, A. (2011), ‘A comparison of different retrieval strategies working on medical free texts’, *Journal of Universal Computer Science* **17**(7), 1109–1133.
- Krüger-Brand, H. E. (2010), ‘E-Health in Österreich: Pragmatischer Ansatz trägt Früchte’, *Dtsch Arztebl International* **107**(28-29), A–1395–A–1397.
- Krüger-Brand, H. E. (2011), ‘E-Health-Strategien: Drei Länder, drei Wege’, *Dtsch Arztebl International* **108**(11), A–562–A–564.
- Lifschitz, V. (2009), ‘Lecture notes on mathematical logic’, University Lecture.
- Lovis, C., Baud, R. H. and Planche, P. (2000), ‘Power of expression in the electronic patient record: structured data or narrative text?’, *International Journal of Medical Informatics* **58**, 101–110.
- Mack, R., Mukherjea, S., Soffer, A., Uramoto, N., Brown, E., Coden, A., Cooper, J., Inokuchi, A., Iyer, B., Mass, Y., Matsuzawa, H. and Subramaniam, L. V. (2004), ‘Text analytics for life science using the unstructured information management architecture’, *IBM Systems Journal* **43**(3), 490–515.

- Maimon, O. and Rokach, L. (2005), *Data Mining and Knowledge Discovery Handbook*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Manning, C. D. and Schütze, H. (1999), *Foundations of statistical natural language processing*, MIT Press, Cambridge, MA, USA.
- McNamara, D. S. (2010), ‘Computational methods to extract meaning from text and advance theories of human cognition’, *Topics in Cognitive Science* **3**(1), 3–17.
- Mitkov, R. (2003), *The Oxford Handbook of Computational Linguistics (Oxford Handbooks in Linguistics S.)*, Oxford University Press.
- Nasukawa, T. and Nagano, T. (2001), ‘Text analysis and knowledge mining system’, *IBM systems journal* **40**(4), 967–984.
- Nowak, M. A., Komarova, N. L. and Niyogi, P. (2002), ‘Computational and evolutionary aspects of language’, *Nature* **417**(6889), 611–617.
- Papadimitriou, C. M. (1994), *Computational complexity*, Addison-Wesley, Reading, Massachusetts.
- Pinker, S. and Bloom, P. (1990), ‘Natural-language and natural-selection’, *Behavioral and Brain Sciences* **13**(4), 707–726.
- Rosenfeld, R. (2000), ‘Two decades of statistical language modeling: where do we go from here?’, *Proceedings of the IEEE* **88**(8), 1270–1278.
- Schmid, H. (1994), Probabilistic part-of-speech tagging using decision trees, in ‘International Conference on New Methods in Language Processing’, Manchester, UK, pp. 44–49.
- Simpson, S. G. (2011), ‘Mathematical logic’, University Lecture.
- Sipser, M. (1996), *Introduction to the Theory of Computation*, 1st edn, International Thomson Publishing.
- Stuckenschmidt, H. and van Harmelen, F. (2005), *Information Sharing on the Semantic Web*, Springer, Berlin.
- Tufféry, S. (2011), *Data Mining and Statistics for Decision Making*, Wiley Series in Computational Statistics Wiley Series in Com, Wiley.
- Turing, A. M. (1950), ‘Computing machinery and intelligence’, *Mind* **59**(236), 433–460.
- Uzskoreit, H. (2000), ‘What is computational linguistics?’, [http://www.coli.uni-saarland.de/~hansu/what\\_is\\_cl.html](http://www.coli.uni-saarland.de/~hansu/what_is_cl.html). Last visit: 28.09.2012.
- Wagner, H., Dlotko, P. and Mrozek, M. (2012), Computational topology in text mining., in M. Ferri, P. Frosini, C. Landi, A. Cerri and B. D. Fabio, eds, ‘CTIC’, Vol. 7309 of *Lecture Notes in Computer Science*, Springer, pp. 68–78.

- Waldrop, M. M. (1984), ‘Natural-language understanding’, *Science* **224**(4647), 372–374.
- Weizenbaum, J. (1966), ‘Eliza - a computer program for study of natural language communication between man and machine’, *Communications of the ACM* **9**(1), 36–45.
- Wu, S. T.-I., Kaggal, V., Dligach, D., Masanz, J. J., Chen, P., Becker, L., Chapman, W. W., Savova, G. K., Liu, H. and Chute, C. G. (2013), ‘A common type system for clinical natural language processing’, *J. Biomedical Semantics* **4**, 1.
- Yndurain, E., Bernhardt, D. and Campo, C. (2012), ‘Augmenting mobile search engines to leverage context awareness’, *Internet Computing, IEEE* **16**(2), 17–25.
- Zhu, W.-D., Iwai, A., Leyba, T., Magdalen, J., McNeil, K., Nasukawa, T., Patel, N. and Sugano, K. (2011), *IBM Content Analytics Version 2.2: Discovering Actionable Insight from Your Content*, Vervante.
- Zins, C. (2007), ‘Conceptual approaches for defining data, information, and knowledge: Research articles’, *J. Am. Soc. Inf. Sci. Technol.* **58**(4), 479–493.