UNIVERSITY OF OSLO
Department of Informatics

Processing Electronic Medical Records

Ontology-Driven Information Extraction and Structuring in the Clinical Domain

Master thesis

Lars-Erik Bruce

August 1, 2012

Abstract

Natural Language Processing (NLP) and Information Extraction (IE) systems can annotate free text with meta-data, thus making it possible to automate processes within clinical care as well as helping medical research gain insight into the tremendous amount of data which lies hidden in electronic medical records. In this thesis we will present an overview of recent developments of methods and resources used in intelligent and robust systems that can extract accurate meta-data automatically from unstructured sources. Further we will evaluate a modern IE system by using its structured output to automatically diagnosing patients based on their discharge summary. This is done by combining the system with machine learning algorithms.

We will not presume any background knowledge of linguistics or medical science from the reader. All essential concepts needed to understand the problem formulations are thoroughly explained, including the nature of the subject to be analysed (unstructured clinical text), through the building blocks of NLP and IE to handling the newly structured information with a computer. We give a bird's-eye view of IE in the clinical domain (chapters 1-5) as well as a worm's-eye view on a specific system used in a specific experiment (chapters 6-7).

We will conclude that even if the ultimate goals of IE still rests on scientific development and discoveries in the future, there is no need to delay any project aiming to extract and structure information within clinical records. We see that it is at least three reasons for this: Firstly, the technology already available to us may benefit Information Retrieval (IR) systems actively used today, and aggregating the structured information can give us valuable clues for initial medical research, for instance when trying to uncover relations between findings, symptoms and drug use.

Secondly we see that the basic building blocks engineered today is likely to be key components also in future systems. Even if research in NLP and IE in the clinical field is scarce, we can today begin to develop new components handling basic NLP tasks geared towards finding linguistic structure in clinical texts, providing the future with useful and needed building blocks.

Last, but not the least, we see that perhaps the strongest barrier in this research field is the lack of annotated free text that can be used to train and evaluate IE systems. Annotated material is necessary when comparing different methods and systems for NLP and IE and is also necessary when training new

NLP and IE modules. Annotating such material is costly, both keeping the data anonymous and private and annotating with linguistics and semantics are tedious tasks. But once it is created it could be available for any further studies within the field. Therefore, we see the urgent need of releasing medical records now, if we want to see productive IE software in the future.

Acknowledgements

I would like to thank my supervisors professor Jan Tore Lønning and professor Arild Waaler for providing me with the initial idea for this thesis. I feel especially indebted to Jan Tore for his valuable comments and support throughout my work.

Markus Lømo, Ole Johan Strandbekk and Elin Munkerud, thank you for proof-reading the manuscript, thus making this thesis more intelligible. Arne Skjærholt deserves credit for helping me with all LaTeX-related issues.

Christina Nilsson at IHTSDO, thank you for all your help guiding me through licensing issues with SNOMED CT.

I would also like to thank all my fellow students for keeping me focused throughout these years: Rune Lain Knudsen, Emanuele Lapponi, Lars Jørgen Solberg, Brendan Lee, Johan Benum Evensberget, Charlotte Løvdahl, Murhaf Fares, Sindre Wetjen and all the rest, as well as fellow nerds in #lalaland@EFNet who have provided a pleasant diversion.

Last, but not least, thank you Elin for your love and support, and bearing with me throughout this endeavour.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The vast amount of information accumulating in all parts of our society is overwhelming. The health care industry is no exception. The rapid growth of Electronic Medical/Health Records (EMR/EHR) and the latest development of sophisticated Natural Language Processing (NLP) techniques and algorithms in general and Information Extraction (IE) in particular makes it valuable to investigate old and new NLP methods that can contribute in medical research and health care. Hospital archives of such records contain a tremendous amount of information about patients, diseases and other findings, valuable information that can be used when data-mining causal links between symptoms, illnesses and diagnoses etc. Much of this information is unfortunately hidden from the computer. The records are written in an unstructured or semi-structured manner, not suitable for searching, summarizing, statistical analysing or decision assistance, as it is meant for human eyes.

With Information Extraction technology, it could be possible to i) Extract mentioned entities within written free text, such as diagnoses, allergies and symptoms. ii) Filter out or mark entities which are negated ("The patient tested negative on ..."), speculated ("The symptoms may be caused by ...") or not about the patient ("with a family history of ..."). iii) Find other pieces of information, such as relations between entities, in what period of time events occurred, etc. iv) Store this information in a structured manner (a manner that the computer can read and "understand"), so that it can be served to the end user when needed, in a desired manner.

Recent developments in Information Extraction and Knowledge Modelling makes it feasible to begin exploring what is needed to make the computer "understand" the content of clinical notes and use it for other purposes than what they initially were meant for, like investigating new and unknown causal relations between findings and diseases. Making the computer "understand" the data can also prove useful in fulfilling the ordinary tasks of an EHR system in a better way, for instance automatically compile summations for given patients for the clinical personnel. When we say that the computer "understands" the content of a document, we mean that it *structures* the information in such a way that the computer knows how to handle it, infers new information from it, and performs new tasks which in other ways is currently not available.

The main topic for this thesis is computational linguistics with respect to Information Extraction, and the overall theme is "How to extract and structure information from non-structured sources?". The focus will be on the utilization of ontologies in this regard. Ontologies is a formal representation of knowledge within a domain, consisting of concepts and relations between these concepts. They can assist us in the area of information extraction because they model the domain, include the means to uncover relevant entity mentions in the text and can be used to infer even more information from what we have extracted.

When extracting information it is necessary to constrain oneself to one domain at a time; it is too big a task to build a system which understands "everything". Health care is an interesting example domain for information extraction for several reasons: The clinical domain is an crucial area because decent health care concerns all of us; everyone needs to be taken care of by medical personnel several times in their life. If NLP and IE can help provide even better health care, this alone is good enough reason to invest in it. From a technical point of view, there is active, ongoing research in the field (Meystre et al., 2008), which has led to several key resources that can be exploited, like ontologies and language technology software built specifically for the clinical domain. With these resources, we have the ability to investigate how ontologies may help in a general IE setting.

We will investigate the possibilities of utilizing NLP technologies and medical terminologies to add structure and computational meaning to clinical documents written in a natural language. This could benefit both existing information retrieval systems used in the medical domain, and converge to a robust and sophisticated information extraction system. This system could, for instance, populate a database with reliable and fruitful data extracted from patient records and discharge summaries. Instead of trying to impose more standards and rigid usage of terminologies onto clinical health personnel, one could develop software capable of "understanding" clinical free text. In this thesis we will see that, given the right tools, factual information from clinical documents can be extracted in such a manner that the computer can "understand" and thus reason over the data. We will exemplify this with a system that can recognize patients with a given diagnosis in our experiments. The system utilizes IE tools in combination with machine learning techniques (see chapter 7).

1.1 Goals to Achieve

Throughout this thesis we will explore steps that have already been taken in the field, so we eventually know what is needed to get equivalent results when developing similar tools for a new language. One of the main goals of the thesis is to depict the road available for future development in medical language processing (MLP). We want to present a clear picture both to researchers within computer science and medical health care in what steps are necessary to develop fruitful technologies for handling and processing clinical free text. The intended readers, then, include researchers from both fields. Therefore, we will present the basic building blocks concerning NLP and IE from the ground up, and try not to presume too much prior knowledge from the reader.

Perhaps the ultimate goal for the computer system is to "understand" the content of clinical text in such a way that it may give correct answers to questions like "Is there any statistical significant relations between usage of drug x and symptom y?". But fruitful systems can be built before we are able to build such sophisticated machinery. For instance, being able to detect mentions of named entities in written text, such as diagnoses and medications, makes it possible for clinical personnel to obtain usable summaries of the patient in care and serve as an utility to Information Retrieval systems. When the computer "understands" which medical entities that reported in a clinical record, the summaries are not restricted to the language used in the record. With a multi-language medical terminology the summary may as well be written in another language.

Schemes like this may sound like science fiction, but current projects already examine how to employ systems with such capabilities. The epSOS project, which aims to offer seamless healthcare to European patients across geographical borders and language borders, is developing a multi-language terminology for this kind of semantic interoperability on a large scale[1].

Ana Estelrich gave an example of a use-case in the Semantic Days conference 2010 (Estelrich, 2010): An Austrian student shows up in a hospital in Dijon, France. She complains about abdominal pains following a meal. An x-ray reveals intestinal occlusion and the physician considers keeping the patient under observation only. The physician runs a search of the patient via epSOS, which returns the student's Patient Summary from Austria. Via terminologically based technology, the French physician retrieves a summary based on a patient record written in German, and discovers that the student has undertaken an emergency cholecystectomy three years ago and had repeated sub-occlusive episodes one years ago. Based on this information, instead of having the student under observation, an abdominal scan is done showing a peritoneal bridle occlusion, and a laparotomy is performed.

Let this be an illustration of what can be achieved by building systems that can extract information from clinical reports. We will not use much space discussing possible future use. Some general use-cases are illustrated in section 2.4 (page 10) so that we have a notion of where this technology comes in handy. Instead of focusing on use-cases we will investigate how to develop IE systems, and which resources are needed for developing such systems in the clinical domain. We will discuss existing technologies, and how these are adapted to the clinical domain. How much work is needed to develop, for instance, a stable

[1]With semantic interoperability we mean that the computer is able to transmit unambiguous data, for instance by using a shared terminology or ontology. Features like these can also help for possible language obstacles.

and accurate POS-tagger[2] (a component that is helpful, if not necessary, for developing reliable IE systems) for clinical free text? What kind of resources does the researcher require before she is able to build systems which perform such a task?

The need of a survey which investigates this matter is vital to both sides of the table. An informatics researcher may not be aware of the specific problems that could arise when entering the medical domain, and a medical researcher may not be aware of the amount or type of resources that are needed to develop a robust and reliable IE system. We hope that this thesis will shed some light on these aspects of IE, and further the process of developing new NLP and IE components for the clinical domain.

One of the conclusions that are to be drawn in this thesis is that even if full-scale IE systems lie in the future, there is no reason to delay initial development. It is fully possible already at this point to begin making the lower-end parts of the system, like a POS-tagger and spell corrector, which could be fully utilized in a future information extraction system. The building of the early resources could also benefit existing information retrieval systems.

1.2 The Route of this Thesis

In the next chapter we will define the problem formulations. The chapter begins with an overview of the field, and introduces some key terms like *Electronic Health Records* (EHR) and *Ontologies*, so that the reader is well-prepared for the presentation of the problem formulations.

In chapter 3 we will see, in general, how we can extract knowledge from a stream of text. Both the necessary Natural Language Processing (NLP) tasks and Information Extraction (IE) tasks will be covered, with an eye on recent developments in these fields for clinical free text.

Chapter 4 describes how we can, and should, handle the information we extract from clinical notes from a computational perspective. We will answer questions such as “What does it mean to say that information is *computable*?” and “How do we model domain-specific knowledge in a computer?”.

Already existing resources that are available for research and development of IE in the clinical domain will be presented in chapter 5. This includes terminologies and ontologies, NLP and IE/IR systems as well as training material that can be used for developing new tools. We will examine one tool in particular, the cTAKES framework, since this is the tool of choice for our experiments. cTAKES, along with the overlaying framework UIMA, will be discussed in chapter 6.

We will present an experiment done on a previous shared task by using cTAKES in chapter 7. This will give us some idea of what an IE system can do now “Out of the box”. In chapter 8 we answer the research questions stated in chapter 2. We will conclude by outlining what could be done when developing

[2]See section 3.1 for POS-tagging.

an IE system in the clinical domain for new languages, if they are to perform at the same level as the similar tools developed for the English language.

Chapter 2

NLP and Clinical Records

In this chapter we will define the weighty words used in the thesis title, and introduce the common background-knowledge needed to understand the problem formulations (that was not covered in chapter 1). We will discuss how the term "Electronic Medical Records" is used in the literature, introduce what we mean by ontologies and briefly explain the concepts of information extraction and structuring. We will also introduce some general use-cases, which might give some clues as to what can be achieved with these technologies, before we end this chapter by spelling out the problem formulations.

2.1 Electronic Medical Records

We want to extract and manage information from free text in electronic clinical notes, including discharge summaries, admission notes, progress notes, etc. In the clinical setting, documents like these are usually collected in Electronic Medical Record (EMR) systems or Electronic Health Record (EHR) systems which are the entities usually referred to in the literature on information extraction in the clinical domain (Meystre et al., 2008). The title of this thesis uses the EMR formulation, while other papers on the subject of IE in the clinical domain refer to EHR (Meystre et al., 2008). This makes it natural to ask what the difference is between these terms, if any. It is also sometimes difficult to understand whether EHR or EMR refer to the systems handling the clinical records or to the collection of records. We will here treat the terms as referring to the systems in accordance with how the terms are used in the MITRE (2006) report (explained below).

It seems like that those who want to separate the terms EMR and EHR think that EHR is "something more" than just an ordinary computer system dealing with clinical records. For instance, according to a report written by MITRE on major EHR models (MITRE, 2006), the idea behind EHR is to collect the data belonging to a patient, to prevent so called "information silos"[1] within clinical care institutions. The report states that when the data

[1]Information silos means that information is hard to share between different departments

is located in information silos, clinical personnel would have to open and log in to different applications to see all information belonging to one patient, or worse, the records get faxed or printed and handled like a regular paper record in the inpatient setting. EHR is supposed to give an integrated access to all data belonging to a patient, across the different sections within or perhaps also outside of the clinical health care centre. EHR is thus about sharing information across different systems. What kind of patient data that is stored in the EHR depends on the EHR-model, but it would typically include laboratory, nursing, radiology and clinical data (MITRE, 2006). That EHR is "something more" than EMR is also witnessed by a statement made by David Kibbe, the AAFP's director of health information technology:

> *EMR connotes a tool that's for doctors only and something that replaces the paper record with a database. EHR connotes more of a connectivity tool that not only includes the patient and may even be used by the patient, but also provides a set of tools to improve work-flow efficiency and quality of care in doctors offices.* (Bush, 2003)

We will not use more space discussing the different aspects of EMR and EHR, but settle with this: We are interested in any documents, fields in formula or other sources of narrative free text written in the clinical domain, i.e. any information sources which are unstructured. For the experiments in this thesis we have used the i2b2 2008 shared task data (see section 5.3), which consists of patient discharge summaries. How to utilize information extraction tools within the EMR/EHR systems should be investigated further. Such tools should generally be configured and fine-tuned for each use-case, the structures of medical records often differ dependent on the source. An example of this can be seen in 7.1, where we extend an IE tool for reading the i2b2 2008 shared task documents.

2.2 Ontologies

An ontology is a description of a specific part of reality, in this setting written in a computer-readable manner. An ontology gives the computer access to pre-defined knowledge within a domain, to be used for clearer communication between different computer systems and reasoning over new knowledge that is being fed to the computer (Hitzler et al., 2009). A simple example is a "family tree"-ontology, which can define relations such that an uncle is the brother of a parent, and that a father is a male parent. If the computer then gets the information "X is the father of Y" and "Z is the brother of X" it can compute that "Z is the uncle of Y". With such an ontology, we can also communicate the fact that "Z is the uncle of Y" in a clear and precise way, by encoding the relation between Z and Y in a computer-readable manner and with a reference

and in some cases even between different systems within the same department.

to an ontology which defines such a relation. These aspects are discussed in chapter 4.

We will see that ontologies can be very useful for locating and structuring information in free text, provided that the text and the ontology belong to the same domain. There is at least three important use-cases for an ontology when extracting information:

1. The textual description of the concept (if provided) may be used when extracting entities. This usage of ontologies will be further explained in section 3.2. A piece of software that utilizes an ontology for entity extraction will also be explained in section 6.2 and tested in chapter 7.

2. The ontology gives us the means to single out the concepts in the running text. By coupling the entity with a concept in the ontology, different systems can communicate and understand each other, provided that they use the same ontology or a mapping between the different ontologies in use. We call this feature Semantic Interoperability, which is briefly discussed in section 4.4.

3. We can also use the knowledge in the ontology to expand the computer's knowledge about the text. For instance, if the computer can extract the information "patient a has disease y", and the ontology contains the information "disease y is a kind of disease x" or "disease x are synonymous with disease y", the computer would know that "patient a has disease x". Such automatic reasoning are discussed in section 4.3.

We will describe ontologies in more detail in section 4.2, and inspect some ontologies for the medical domain in chapter 5, and in particular investigate the SNOMED-CT ontology in section 5.1.

2.3 Information Extraction

Information Extraction (IE) will be described in further detail in section 3.2. In general we are talking about extracting interesting/relevant information found in unstructured sources, such as free text, and giving it a form of structure, for instance storing it in a database. It can be enlightening to look at the task of IE by distinguishing this with Information Retrieval (IR) (Manning et al., 2008):

> *Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large text collections (usually stored on a computer).* (Manning et al., 2008, p. 1)

As such, ordinary searches on a web search-engine or in an e-mail client, as well as in documents in EMR systems in the clinical setting, are called IR. We type some words (a "query") and search documents containing the same or similar words. IR systems become more and more sophisticated, where queries can be automatically spell-corrected, expanded with synonyms and alternate spellings (plural forms, etc.) (Manning et al., 2008). In IE we are not interested primarily in fetching the correct documents for the end-user, but rather the information within the documents. An IE system would extract named entities, relations between entities etc. to create summaries and statistics about the information within the documents. For instance, from an IE perspective it makes a huge difference if the document says "The patient has diabetes" or "The patient does not have diabetes". In an IR system, these documents could very well be equally relevant for a string search such as "diabetes".

2.4 Use Cases

We will here broadly sketch two scenarios for extraction and structuring information within clinical records. The first scenario is in the clinical domain, where we want to exploit information within free text fields of EHR or EMR documents in order to assist clinicians in their work with their patients. The other scenario belongs to the field of medical research, where the extracted information can be aggregated across all patient records within EHR or EMR. We imagine that one could compute vital statistics that could prove useful when researching new causal relationships between different findings.

In the clinical setting Velupillai (2012) define three use case scenarios: Adverse event surveillance, decision support alerts and automatic summaries. If one knows what kind of triggers that might indicate adverse events one could make a surveillance system which tries to detect these triggers in patient records automatically. A system for decision support alerts could indicate to the clinician for instance when it detects two medicines that should not be mixed in the same record, or if clinical findings (for instance analgesic drugs) does not match medication dosage (for instance pain medication). An information extraction system could also create summation lists based on records belonging to one patient, and help the clinician by giving an overview of the patient by means of present and previous diagnoses, allergies and other important medical conditions.

After structuring the information of interest in clinical notes and records, the knowledge is ready to be distributed in a uniform manner and treated by different types of systems (which support the given data-structure and terminology). If the extracted semantic content that is being stored in the computer system is linked with an ontology translated to different languages, it is — as we have said — also possible to give summaries and problem lists in the languages supported by the ontology.

Another type of scenario is the researcher who wants to investigate possible links between treatments, symptoms, drug usage and diagnoses. By extracting information about these kinds of entities in clinical records, the computer is able to report on co-occurrences, possibly gaining new insight into the clinical field. The scientist could query an IE system, containing numerous records, asking for how many of them that contain two or more different entities or events. Linking each event and entity (such as symptoms) with a date may provide even more information. With large enough databases, one can begin investigating new symptoms in accordance with prior prescriptions or procedures, uncovering unknown side effects or track the general development of patients with a given diagnose.

This scenario is perhaps a possibility in a not-so-distant future, since the accuracy is less important. Missing an allergy or a diagnosis when retrieving a summary for a patient may be critical, but in a research setting this might be bearable, given that the researcher knows about the limits of the system. As long as most of the extracted information is precise, and most of the information within the records is recalled, this might still prove useful as an *initial* investigation of causal links between clinical events.

2.5 Problem formulations

This thesis covers many topics, ranging from NLP technologies as used in the clinical domain, to how to utilize these to extract and store information hidden in free text parts of electronic records, as well as how to manage and reason over extracted knowledge. We will, however, narrow the focus down to two problem formulations.

The most specific problem formulation is how we can utilize ontologies, taxonomies and/or terminologies in extraction technology. The main focus in our experiments will be on the SNOMED-CT ontology. We will see what an ontology is and some of the inner workings of SNOMED-CT (see sections 4.2 and 5.1). Furthermore, we will discuss how we can benefit from both the computational knowledge within the ontology as well as the text strings describing the concepts defined within the ontology. We will briefly discuss how to model domain knowledge in the computer (see section 4.2), and we will examine a system which utilizes SNOMED-CT for extracting clinical entities (cTAKES, see section 6.2). We will evaluate the performance of this system in our experiments.

We also want a broader perspective in our thesis, discussing how to build an IE system in general. This problem formulation includes questions such as: What resources do we need? What resources exists today? Successful development and evaluation of IE tools depends on having access to a large number of clinical notes. Since no such records written in Norwegian were available to use in this work, we have focused on similar tools developed for the English-speaking part of the world. We examined them to get a clear view on how one could develop similar tools for languages other than English. We will not look into specific linguistic challenges across languages.. We will instead see which modules and resources are needed to achieve similar results for non-English clinical records.

Chapter 3

Extracting and Structuring Information

The process of Information Extraction (IE) consists in uncovering information of interest from a semi-structured or unstructured source in a specific domain (Hobbs, 2002). This includes extracting Named Entities (NEs), events and relations between these entities and events. Specifically for the clinical domain we have entities like diagnoses, symptoms and drugs and events like procedures and drug usage. The main idea is to capture some kind of structure in previously unstructured documents. This usually means that we store extracted information in such a semantically well-defined manner so as to enable the computer to communicate the information and draw new inferences from the data.

We will here split the task of Information Extraction into three parts. In part one, well known Natural Language Processing (NLP) techniques are used to uncover linguistic elements and structure in the text. This includes detecting words and sentences, reveal the lexical category of each word and find phrases and parse trees over each sentence. This is covered in section 3.1, where we will give a general introduction to each task, followed by a discussion of specific challenges and approaches in the domain of clinical free text. Part two consists in extracting semantically well-defined units, such as named entities (objects with a proper name), events, relations, negated expressions and temporality, which is discussed in section 3.2. Since the particular methods used in Information Extraction are often intimately related to the domain, we will in this section for the most part investigate methods particularly used in the domain of clinical free text. The last part deals with storing the extracted information in a structured manner, with regards to retrievability, semantic interoperability and logical reasoning. This is discussed in the next chapter.

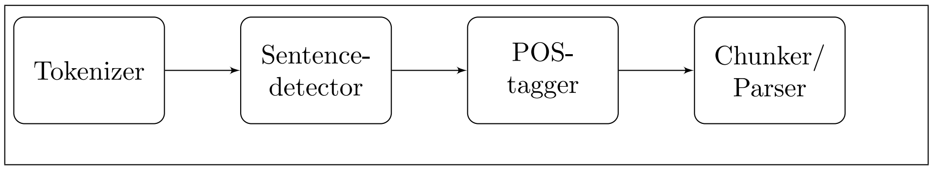

Figure 3.1: A simple NLP pipeline.

3.1 Natural Language Processing

Before extracting information from documents, it is necessary to process the text with standard NLP techniques. From an IE perspective, this is often seen as "pre-processing". These steps are taken before we actually extract semantic information from the text (Hobbs, 2002; Meystre et al., 2008). We will mainly follow the typical route in an NLP pipeline[1] for processing text when we introduce the different tasks. The different tasks can be viewed as modules in a pipeline, following a route similar to the one pictured in figure 3.1 (albeit a minimal one). Unstructured text is served to the first module in the pipeline, and leaves the last module annotated with linguistic information. The tokenization process consists in uncovering words and other textual units. POS-tagging deals with uncovering the lexical categories (verb, noun, etc.) of each word. Chunking is the task of discovering phrases in the text, and parsing uncovers the linguistic structure in the text.

We will investigate these tasks, first with a general perspective and then with a closer examination of the challenges and specifics for the domain of clinical text. The precision of such pre-processing tasks are crucial for the further success of information extraction, but proves difficult in the domain of health care records: As we will see, clinical free text is in many cases harder for computational processing, as it is disease-ridden with misspellings, bad grammar and unconventional abbreviations. This needs to be dealt with in a proper manner. Further, the lack of corpora tagged with information about tokens, word-senses, POS-tags, correct spellings etc. makes it difficult both to train a system to uncover such information in clinical texts, and to evaluate the different solutions.

The line between the "pre-processing" steps and information extraction steps is not clear. For instance Meystre et al. (2008), in their review of IE within the clinical domain, treats word sense disambiguation (WSD) as a pre-processing task, but uncovering the meaning of a word could arguably be an IE-task as well. We will here handle tokenization, spelling correction, WSD, sentence and section-detection, POS-tagging and chunking/parsing as typical "pre-processing" steps. Tasks such as discovering named entities and their relations, negation detection and handling temporality will be treated as IE-tasks.

[1]A pipeline is a modular software framework where the input is treated in a specific order.

We will first introduce the typical NLP tasks, and explore what specifically has been done for the clinical domain. Later we will see how these are implemented in some of the best known systems for processing clinical text (see section 5.2). In the section Words and Tokens we will see how to identify and separate the words and punctuations in a running text. Handling spelling errors and acronyms in the clinical domain will also be discussed. We will see the importance of detecting sentences and sections. We will further explain the process of Part of Speech (POS)-tagging, where we identify the lexical category and inflectional status for each word. Lastly we will have a quick look on chunking and parsing of text.

Words and Tokens

The first step in any text processing task, be it indexing documents for search engines or creating a summary of the text, is to split the stream of text into tokens. This process, called tokenization, tries to identify the entities a text is made up of, such as words, numbers, punctuations and names. As straightforward as this may seem, the task is not necessarily trivial. For instance, a tokenizer (the component performing tokenization) should in most cases recognize “New Orleans” as a single token (a name), not two, despite the fact that the entity is build up of two words. One would also divide “I’m” into several tokens, to convey that it is really two words. Usually one would also treat “Dr.” as one token, instead of splitting it up in two tokens or neglecting the period.

A standard approach to tokenization is to simply split the text on given characters (Jurafsky and Martin, 2008), usually white-space characters and punctuation[2], then use rules to merge tokens back together to create tokens like “New Orleans” and “Dr.”. Depending on the system, one could remove white-space and/or punctuation from the stream of tokens handled further in the NLP pipeline. In Figure 3.2 we see some examples of different tokenization schemes. In output (a) we split the stream of text on white-space, in (b) we split on white-space and punctuation, and remove punctuation. Example (c) is output from the cTAKES system[3], where “follow-up” is treated as one token and “1-2” as three. Some approaches also try to identify the type of each individual token, with types such as “word”, “number”, “person title”, “name”, etc.

One of the important things to keep in mind with regards to tokenization, is the fact that components further down in an NLP pipeline depend on a uniform stream of tokens. For instance, a POS-tagger will not perform well if it is used on text that is tokenized differently than from the material it has been trained on. Therefore, when developing a system for processing clinical text, a standard scheme of tokenization should be chosen, where everyone involved in

[2]This approach would not work on text written in languages where words are not separated by white-space, such as Chinese.

[3]An information extraction system for clinical text, discussed in section 5.2 page 54 and section 6.2 page 62.

input She was given explicit instructions to follow-up in clinic with Dr. Santo Rabalais 1-2 weeks.

(a) $\boxed{\text{She}}\ \boxed{\text{was}}\ \boxed{\text{given}}\ \boxed{\text{explicit}}\ \boxed{\text{instructions}}\ \boxed{\text{to}}\ \boxed{\text{follow-up}}\ \boxed{\text{in}}\ \boxed{\text{clinic}}\ \boxed{\text{with}}\ \boxed{\text{Dr.}}\ \boxed{\text{Santo}}\ \boxed{\text{Rabalais}}\ \boxed{\text{1-2}}\ \boxed{\text{weeks.}}$

(b) $\boxed{\text{She}}\ \boxed{\text{was}}\ \boxed{\text{given}}\ \boxed{\text{explicit}}\ \boxed{\text{instructions}}\ \boxed{\text{to}}\ \boxed{\text{follow}}\ \boxed{\text{up}}\ \boxed{\text{in}}\ \boxed{\text{clinic}}\ \boxed{\text{with}}\ \boxed{\text{Dr}}\ \boxed{\text{Santo}}\ \boxed{\text{Rabalais}}\ \boxed{\text{1}}\ \boxed{\text{2}}\ \boxed{\text{weeks}}$

(c) $\boxed{\text{She}}\ \boxed{\text{was}}\ \boxed{\text{given}}\ \boxed{\text{explicit}}\ \boxed{\text{instructions}}\ \boxed{\text{to}}\ \boxed{\text{follow-up}}\ \boxed{\text{in}}\ \boxed{\text{clinic}}\ \boxed{\text{with}}\ \boxed{\text{Dr}}\ \boxed{\text{.}}\ \boxed{\text{Santo}}\ \boxed{\text{Rabalais}}\ \boxed{\text{1}}\ \boxed{\text{-}}\ \boxed{\text{2}}\ \boxed{\text{weeks}}\ \boxed{\text{.}}$

Figure 3.2: Example input and outputs with different tokenization schemes.

the same project adhere to the same tokenization scheme. We will also see that different tokenization strategies influence tasks like detecting and expanding abbreviations.

Spell-cleaning in the clinical domain

After settling with a specific procedure on how to tokenize the text, it is often fruitful to process the single tokens further. Correcting misspellings would certainly improve performance when extracting information. This is particularly so in health records; Ruch et al. (2003) reports that misspellings in medical records is about 10%, which is higher than text from other genres. Prioritizing the development of a spelling cleaner in IE system for the clinical domain is therefore a good idea. It is worth pointing out that some spelling correction methods use context-information, such as POS-tags, when cleaning misspelled words. In cases like these, the spelling correction module needs to be placed further down in the NLP pipeline.

The classic version of a spelling cleaner is to use the "string-edit-distance" between a misspelled word and each word in a dictionary. One measures how many individual edits one needs in order to transform the misspelled word into each dictionary word, and select the one with fewest edits. Inserting a new character, deleting an extra character or replacing a character is usually considered as one "step", the sum is called Levenshtein score or edit-distance. Some improvement can be made, for instance: When the replaced character is closer to the replacement on the keyboard, this could be viewed as less costly than if the characters are far apart. For instance in the misspelling "lyngs" the normal edit-distance between the correct spellings "lungs" and "longs" is the same. We only need to replace one letter, the edit-distance score is accordingly 1. But since the letter "y" is closer to "u" on the keyboard than "o", replacing "y" with "u" could generate a lower score than replacing "y" with "o". The correctly spelled word (out of the two alternatives) with lowest edit-distance score is then “lungs”.

Another useful tool for spelling correction is Metaphone. This algorithm detects whether two words are similar-sounding by creating a “code” for the word based on pronunciation. When similar-sounding words are processed by the algorithm, the same phonetic code is output. With such an algorithm one can then compute the phonetic encoding of all the words in a dictionary. If one has a misspelled word one can then retrieve all similar-sounding words from the lexicon, and use heuristics (such as the edit-distance) to determine which one of these is most likely correct. For a discussion on the Metaphone algorithm, compared with the newer Double Metaphone algoritm, see the article *The Double Metaphone Search Algorithm* by its inventor Phillips (2000).

Tolentino et al. (2007) made a spell cleaner for clinical text with the help of a domain specific lexicon[4] and a more general lexicon[5]. Their spelling correction method consists of four stages. Error detection (stage 1) detects spelling errors by looking up each word token in the dictionaries. If the word is not in a dictionary it is marked as a spelling error. Word list generation (stage 2) extracts candidate words, using a range of methods like the Metaphone algorithm and selecting legal words with any extra character inserted in the misspelled word. Word list disambiguation (stage 3) sorts the candidate words with the lowest edit-distance. Since two or more words can be “tied”, several extra methods are used to give a score. The last step consists of error correction, basically replacing the misspelled word.

When we know what kind of word or entity we are confronted with, for instance a diagnosis from the diagnosis section of a record or a drug from a medication section, it is also possible to restrict the candidate correct words. For instance, Levin et al. (2007) used the Metaphone algorithm when spell-cleaning drug names from free text, checking misspelled words in the “medication” field in patient records against a drug dictionary. One could possibly try something similar with both medications, allergies and diagnoses given that these fields are marked in the records.

Word Sense Disambiguation (WSD)

WSD is another important NLP task that contributes to IE. When a word can have several meanings it seems intuitive that we must discover the correct one prior to extracting reliable information from the document. Often WSD is done after POS-tagging; a word which can have several senses can be disambiguated by knowing the part of speech in itself. The word “dose”, for instance, would mean “amount of medication” if it is a noun and “giving a patient medication” if it is a verb. The task of WSD, then, is to assign a *sense* to an ambiguous *word* from a set of possible senses the given word can have.

[4]The UMLS Specialist Lexicon, as described on page 50.

[5]WordNet.

Liu et al. (2001) use an unsupervised technique[6] by deriving senses from abstracts and building a corpus automatically. They used the UMLS Metathesaurus (described on page 49), MEDLINE abstracts and the Clinical Data Repository for automatically generating a sense-tagged corpus. They developed an evaluation set which only consisted of abbreviated terms, since this could be obtained automatically.

Abbreviations in clinical notes

With *abbreviations* we typically mean acronyms (BP for blood pressure), shortening of words (pt for patient), contraction of phrases (t/d/a for tobacco, drugs or alcohol) and symbols (etoh for alcohol). Other ways of shortening words, terms or phrases could also be included, like writing “2/2” instead of “second to”. The importance of revealing the words behind abbreviations before automatically extracting information should be self-evident.

Resolving abbreviations and acronyms is in itself a case of WSD, and is also of major importance for the overall quality of IE. The reported amount of abbreviations in clinical notes is huge, Xu et al. (2007) reported that 17.1% of the word tokens in a set of admission notes were abbreviations. The sheer amount is not the only problem with abbreviations in clinical notes; they are often ambiguous, i.e. different terms can have the same abbreviations. Different departments within the domain of health care, such as the laboratory or the general practitioners, often use different sets of abbreviations.

Xu et al. (2007) and Wu et al. (2011) have investigated abbreviations in the clinical domain and how to detect these with machine learning techniques. Xu et al. examined ten admission notes where abbreviations were annotated by a domain expert, whereas Wu et al. did a larger project with 70 discharge summaries annotated for abbreviations.

Xu et al. performed a comprehensive study of abbreviations in clinical free text. They grouped them into four types, listed in Table 3.1 with examples and frequency rate: Acronyms, shortened words, contractions and a last group “Other” for all other forms of shortenings. The frequency ratios – which were calculated one the basis of one hundred example abbreviations randomly selected from the material – tells us that most abbreviations are acronyms and shortened words.

We think that Xu et al. (2007) have done three important things in their work with abbreviations; they have investigated several abbreviation detection methods, analysed the abbreviations found in the clinical notes thoroughly and analysed the coverage of abbreviations in some of the most well-used terminologies for the medical domain. They developed a machine learned decision tree for detecting abbreviations, which performed well on the selected material. The decision trees looked at features for each token, such as word formation (length, type of characters, etc.), document frequency and whether the word

[6] A unsupervised technique means that we are not giving ready-annotated material to the computer. This is explained in more detail in the POS-tagging section below.

Abbr. Type	Examples	Frequency
Acronym	BP-Blood Pressure	50 %
Shortened Words	Pt-Patient Sx-Symptoms	32 %
Contraction	t/d/a-tobacco,drugs or alcohol	9 %
Others	etoh-alcohol	9 %

Table 3.1: Different types of abbreviations and their frequencies as reported by Xu et al. (2007).

exists in a dictionary. Their abbreviation detection method examined each token in the running text. For the best technique they investigated, 91.4% of the tokens selected as abbreviations were actual abbreviations (330/361), and of all the abbreviations in the clinical notes, the technique spotted 80.3% of them (330/411).

Xu et al. (2007) also report that their error analysis uncovered mistakes when detecting abbreviations due to how the text was tokenized; when an abbreviation were separated into several tokens their system could not detect the whole abbreviation but at best only each part. As examples they introduce “S. Aureus” and “ex tol”. “S. Aureus” is here tokenized as “$\boxed{\text{S}}\ \boxed{\text{.}}\ \boxed{\text{Aureus}}$”, so their abbreviation detection techniques did not uncover “S. Aureus” as a targeted abbreviation (but it did find “S” and Aureus”). Something similar happened to “$\boxed{\text{ex}}\ \boxed{\text{tol}}$”.

Token Normalization

When treating tokens one should also consider normalizing them before bringing them further in the NLP pipeline. Often the same word may take different forms, for instance “patient” and “Patient” are spelled differently (with or without a capital P), but in most cases we want to treat these as the same word. The computer will treat “Patient” and “patient” as two distinct words, unless explicitly told otherwise. The normal approach here is to simply reduce all letters in a text to lowercase. We could instead lowercase only the first letter in the first word in every sentence, so that we can keep upper cased letters in proper names, etc (Manning et al., 2008).

The removing of diacritics could also improve further processing, for instance by converting “ï” in a text to “i”, thus treating “naive” and “naïve” as the same word. It is unknown how much this could benefit the clinical setting; in a quick experiment with 612 discharge summaries[7] we discovered that no words contained either “ï”, “ê”, “é” nor “è”. For languages such as English and Norwegian, where such diacritics have a marginal status (Manning et al., 2008), performing this normalizing step is perhaps not so important.

Often we want to *lemmatize* the word-tokens, which means reducing the words into their base-form (i.e. their lemma, canonical form or dictionary

[7]From the i2b2 2008 shared task, see section 5.3.

form). This can be necessary for instance when mapping terms from free text to an ontology. **Run**, **runs**, **ran** and **running** are, for example, forms of the same lexeme ***run***. A similar (but more crude) method is stemming, which uses several generic rules to chop of parts of a word in order to reduce it to a base form. While lemmas correspond to the lexicon-form of a word, the result of stemming two different words could result in the same stem. As an example, “stocks” and “stockings” could both be stemmed to “stock” (Jurafsky and Martin, 2008). A reverse technique is also sometimes employed, whereby the token is expanded into several inflected variants. We see an example of this in the cTAKES system (see section 5.2).

Sentences and Sections

After tokenization is done, it is valuable to identify sentences and sections in the documents. Finding sentences is often necessary before performing POS-tagging, and must be done before finding phrases and building parse trees. The simplest way to split a running narrative text into sentences is to split the text at punctuation — periods, question marks and exclamation points — but problems occur with the period because it is not only used to mark the end of sentences (Jurafsky and Martin, 2008). Different approaches to disambiguating periods based on machine learning are used, for instance a Maximum Entropy classifier (Guergana et al., 2008).

Finding the different sections of the document (section segmentation) is also convenient, if not necessary, when uncovering the information within it. It seems intuitive that some sections of a clinical record are more important than others. For instance would sections named “Primary diagnosis” and “History of present illness” seem more important than sections named “Comments” and “Family history”. Even if we do want to extract information from the latter sections, it seems important not to relate, for instance, diagnoses found in “Family history” to the patient the record belongs to. The work of Cho et al. (2003) contains a lot of references to work in section segmentation in general, and describes an algorithm developed for medical reports in particular. Childs et al. (2009) developed a rule based system for uncovering obesity and comorbidities in order to meet challenge described by Uzuner (2008), where discovering the different sections of clinical discharge summaries was a main ingredient.

POS tagging

Part of Speech (POS) tagging is the process of labelling a running sequence of tokens with a set of POS-tags. A POS-tag typically identifies lexical categories (verb, noun, etc.) and inflectional features (present or past tense, etc.) of word-tokens. Punctuations are also often tagged. POS-tagging is therefore intimately related to the tokenization process. Given a string of tokens as input, and a set of category-tags, a tagger tries to assign the best tag for each token. Often it is not given by the morphological structure of the word itself what category it belongs to. The word “dose” could be either a verb (giving a patient medication) or a noun (the amount of medicine the patient should take). We need to reveal the context of the word (i.e. look at surrounding words) in order to identify what lexical category it belongs to.

She	was	given	explicit	instructions	.
PRP	VBD	VBN	JJ	NNS	.

Figure 3.3: Example output from POS-tagging.

Knowing the POS-tag of each word is a valuable resource when further extracting information from any text, and is often necessary before parsing or chunking. It is also often used when extracting named entities from the text. Therefore, a POS-tagger is important in any information extraction system.

The part of speech of the word is typically defined either as the syntactic-morphological behaviour of the lexical item (e.g. a determiner often precedes a noun, a present tense verb often ends with ’ed’) or in some semantic term (e.g. verb is often a type of event, process or action). There are varying granularity of the tags in different settings, and hence different sizes of the set of POS-tags. For instance, the Brown Corpus contains 87 simple tags, while the Penn Treebank (PTB) tagset is reduced to 36+12 (the final twelve being tags for punctuation and currency). The varying granularity typically stems from decisions such as whether the tag should reflect inflection of the word or include some syntactical information. In some cases a word can have it own set of tags, for instance in the tagset for the Brown Corpus ’have’ has it own base-form tag, but not in PTB (Jurafsky and Martin, 2008).

It is common to divide the lexical categories into open and closed classes. The open class of words — nouns, verbs, adjectives and adverbs — is the main bearer of meaning in text. This is a dynamic set of words, and when new terms are coined, they often belong to one of these classes. On the other hand, the closed classes consist of a static set of words often devoid of any meaning, and it is rare to see new members of this class. (Some examples of closed classes includes prepositions, articles and conjunctions.) This information can be valuable in POS-tagging; when a tagger discovers a new word, we know that it is unlikely that the word belongs to one of the closed lexical categories. When using a statistical approach to POS-tagging, this can be indirectly learned from the training material, while rule based POS-tagging approaches may have rules like “If the word is unknown, do not consider tagging it as something in the closed classes”[8].

The tagging of a sequence of tokens is often done either by a rule-based procedure, with several rules describing what tag a word should get, or a stochastic one like a Hidden Markov Model (HMM). In a rule-based approach linguistic experts write rules for how to assign POS-tags. A typical approach is to assign all possible POS-tags for running words in the text (according to a dictionary),

[8]One would probably never see a rule like this explicitly stated.

and then use rules for disambiguating word-tokens which are assigned several POS-tags (Jurafsky and Martin, 2008).

A stochastic approach to POS-tagging involves computing statistics over large amounts of texts hand-annotated with correct tags, called a *gold-standard*. The input is a sequence of tokens, and the task is to assign a sequence of POS-tags. The tagger uses probabilities to assign POS-tags. We have most likely never seen this exact sequence of tokens before, so we must compute the probabilities differently than by looking at the whole sequence at a time.

By looking at how frequent a word is assigned different POS-tags[9] in the hand-annotated texts, we can compute the statistical correlation between words and POS-tags. Also, by looking at sequences of POS-tags in the annotated material, we can use these to compute the probability of the different sequences of POS-tags that can be applied to the running word-tokens. We can then combine these probabilities in order to compute the most probable sequence of POS-tags given a sequence of tokens.

A typical example of a stochastic model used in POS-tagging is the Hidden Markov Model (HMM) (Jurafsky and Martin, 2008). A HMM uses a combination of *emission probabilities* and *transition probabilities*. Emission probabilities express the probability of seeing a word w assigned to a POS-tag T ($P(w|T)$). Transition probabilities represent the probability of seeing a POS-tag T_{i+1} after a POS-tag T_i. We can compute these probabilities with the gold-standard. When seeing a new sequence of tokens we could then, for all possible sequences of tags compute which one of those are most likely according to emission and transition probabilities[10].

POS tagging in the clinical domain

While there are plentiful of resources with tagged data available for training a tagger, the accuracy of the tagger drops when used on domain-specific texts like EHRs (Coden et al., 2005). This is probably due to a number of reasons, for instance spelling errors and bad grammar in clinical notes. The biggest source of degrading performance is possibly all the unknown words. For instance the accuracy of the TnT tagger on known tokens in the NEGRA corpus is 97.7%, this figure drops to 89% on unknown words. The figures are similar for the Penn Treebank corpus (Brants, 2000).

Since many words used in the clinical domain are rare or non-existing in corpora with general text, it is natural to assume that tagging accuracy will drop. Accuracy and reliability is of the essence when extracting information, and it is therefore important to have a good POS-tagger when making an IE system. How can we obtain this for clinical text? When using a stochastic approach, it seems that the best bet is to use tagged in-domain text. This is, however, not always an option, since making such a corpus is costly and when it first is made it is difficult to obtain for other scientific purposes because of

[9]Or how frequent a POS-tag is assigned to different words.

[10]An algorithm (Viterbi) exists where we do not need to compute this for all possible sequences of POS-tags to find the most likely sequence.

Training material	Accuracy
TB2	88%
MED	92%
TB2+MED	93%
TB2+Lexicon	88.82%

Table 3.2: Reported accuracy figures in Coden et al. (2005), evaluations on MED

the need of keeping clinical documents confidential (Uzuner et al., 2007). To resolve this critical aspect of POS-tagging text in the clinical domain, Coden et al. (2005) tested different strategies of mixing general training material with both a small set of tagged clinical notes and a domain lexicon.

Coden et al. wanted to see how one could boost a tagger trained primarily on out-of-domain content, both with and without using in-domain training material. They used a subset of the Penn Treebank (TB2) as the general training material, and a corpus of clinical documents (MED) to train and evaluate the different settings. The classifier was an HMM working on tri-grams. When they used a lexicon in addition to TB2 for training their POS-tagger, they manipulated the emission probabilities of the stochastic model.

When they trained the classifier with TB2 and evaluated on MED, they got an accuracy around 88%. Using MED to train the classifier, the accuracy was 92%. Training on both TB2 and MED, they reached an accuracy of 93%. Since tagged clinical documents is difficult to obtain, they also tried to train the tagger using TB2 plus a lexicon of the 500 most frequent words from a collection of clinical documents (minus stop words) together with the POS-tagged PTB-material. They then achieved an accuracy of 88.82%.

Their experiments give us important clues as to how we can achieve good results for POS-tagging in the clinical domain. The utopian setting is having access to a large amount of in-domain texts annotated with POS-tags. If this is not obtainable, using a smaller in-domain corpus together with a bigger general corpus seems like the next best thing. If none of these are available, one can train a POS-tagger using a lexicon of the most frequent words from the clinical domain, which gains a small improvement.

Another, rule-based, approach of POS-tagging clinical data has been developed by Dwivedi and Sukhadeve (2011). They worked on a collection on homoeopathy texts, including books, medical reports and prescriptions, and manually annotated 125 sentences for evaluating their system. In a step-by-step manner, their system analyses a sentence and finds phrases, followed by the clauses and finally the remaining modifiers. This is done with 485 grammar rules with the stemmed versions of the words. This system, then, actually "parses" or chunks the input text, and assigns POS-tags accordingly. In their final evaluation of the system, they achieved an accuracy of 88.93%, similar to the result of using a stochastic tagger trained on general purpose corpora plus a domain-specific lexicon. But be aware that these systems were evaluated on

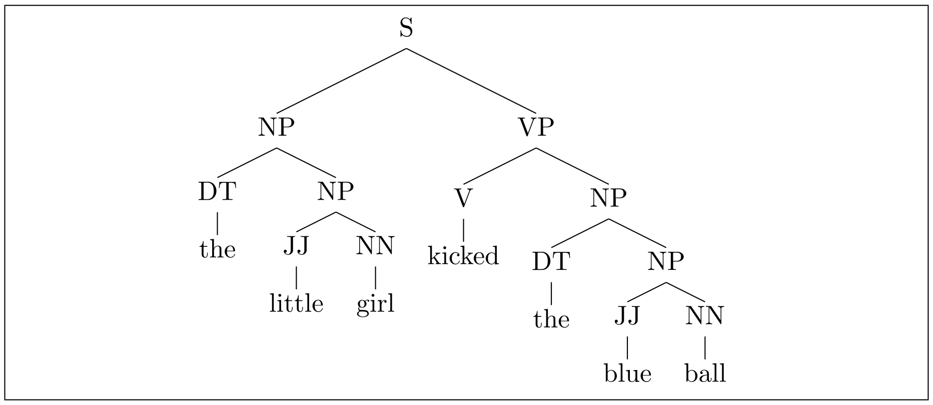

Figure 3.4: A syntactically parsed tree.

different data-sets.

Since it appears to be less work developing a domain-specific lexicon and use this with freely available state-of-the art stochastic taggers, this seems to be the best choice when tagged in-domain resources are unavailable or scarce. On the other hand, if the rules used to capture POS-tags also identify useful syntactical structures, using a rule-based method may be worthwhile.

Phrases and Chunking

The last step before IE is to discover linguistic structure in each sentence. This includes tasks like parsing and chunking. Parsing is the process of creating some *hierarchical* linguistic structure for the input text. This can involve everything from morphological structure, to syntactical and semantic structure (Jurafsky and Martin, 2008). Morphological structures can be useful in order to uncover when one is to link the word-tokens in the running text to semantic entities, by using all inflection variants of the word. Syntactic structure might be useful for identifying which parts of the text are likely to reveal valuable information. Syntactic structure also reveals important clues about entities mentioned in the text, such as whether the entity in question is an object or a subject of the verb. We will not cover parsing here, since most of the investigated IE-system (see section 5.2) tends to rely on detecting flat syntactical *chunks* of text instead of building hierarchical analysis.

Finding the chunks of a text is usually called chunking or shallow parsing. Detecting noun phrases (NPs) is especial helpful when uncovering *entity mentions* which we are trying to detect in the information extraction process. Chunking generally consists of finding noun phrases, verb phrases and in some cases preposition phrases and adjective phrases. These phrases lack any hierarchical structure, but are instead segments within sentences corresponding to the open Part of Speech classes (Jurafsky and Martin, 2008). A typical example of a chunked sentence is given in figure 3.5. We can see this in contrast with the syntactically parsed tree in figure 3.4. As for the noun phrases (NPs)

$\Big[_{NP}$ The little girl$\Big]$ $\Big[_{VP}$ kicked$\Big]$ $\Big[_{NP}$ the blue ball$\Big]$.

Figure 3.5: A chunked sentence.

The $\Big[$ little girl$\Big]$ $\Big[$ kicked$\Big]$ the $\Big[$ blue ball$\Big]$.

Figure 3.6: A semantic chunked sentence.

the chunk equals the same text span as the highest NP-nodes in the tree, while the VP-chunks equal the highest V-node.

As with POS-tagging, chunking could both be based on rules or machine-learning. When using machine-learning, detecting the phrases is done in a similar manner as tagging POS-tags, but we use tags which identify the beginning of a phrase, inside a phrase or outside a phrase instead.

Chunking in the clinical domain

Bashyam and Taira (2007) argue that since it is difficult to obtain grammatically correct sentences in the clinical domain — sentences are often partial and often lack strict punctuation — there are disadvantages of using chunking methods based on syntax. They propose instead a semantic chunker, and define a semantic phrase to be "a sequential set of word tokens which can be effectively replaced by a single word belonging to the same semantic category as the phrase" (Bashyam and Taira, 2007). We can see an example of this in figure 3.6. The only visible difference in the end-result we can find here, is that we lose the determiner "the" from the NP chunks.

They treat chunking as a sequential classification task, where they label the start, end, inside of and outside of chunks, as well as single-token chunks. Support Vector Machines (SVMs), which is a stochastic tool for supervised machine-learning, were used to build the (model for the) classifier. After taking the typical NLP steps described above (tokenizing, POS-tagging, sentence and section segmentation, etc.) they had a domain-expert annotate anatomy phrases within 1250 sentences of radiology reports. From their test set of 423 phrases, their system correctly identified 350 of these (Bashyam and Taira, 2007).

One interesting aspect of the work of Bashyam and Taira (2007) is that they focused on finding *anatomy phrases*. This could perhaps serve as a good starting point for detecting clinical entities (see section 3.2). For instance the SNOMED-CT ontology (see section 5.1) defines terms describing *body structures*. Knowing that a phrase is an anatomy phrase, one could then focus on the *Body structures* part of SNOMED CT when coupling phrases with named entities. This should be further studied when developing clinical named entity recognizers.

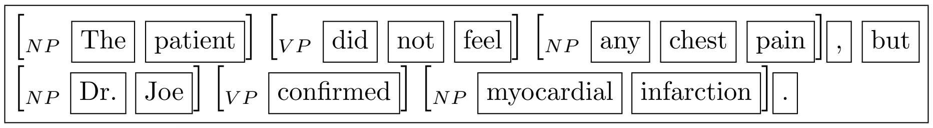

Figure 3.7: A tokenized and chunked sentence, ready for information extraction.

Summary

Within all major fields of NLP there have been undertaken research of how to do the different tasks within the clinical domain, and we have here briefly discussed some of this work. We have seen that the language in clinical notes differ from general text, both regarding grammar and vocabulary. We see that more annotated textual material are needed for the further improvement of the different tasks, as well as investments into annotating such texts with word senses, full forms of abbreviations, POS-tags, phrases and parse-trees.

Some methods do not rely on annotated material, but this comes at the cost of accuracy. This includes using a lexicon in addition to general tagged data, or making a rule-based system for POS-tagging(Coden et al., 2005; Dwivedi and Sukhadeve, 2011). We would, never the less, always need annotated material in order to *evaluate* how the different methods performs.

3.2 Information Extraction

After the above described pre-processing procedures, we are ready to begin extracting information from the narrative text. This basically means that we recognize entities, relations and events in the text and store these in a structured and retrievable manner. In this thesis we take the perspective on Information Extraction (IE) as a task centred around recognizing *terms* and *entities* of interest in the running text. In general IE systems this would typically be named entities, such as references to individual persons, cities and companies (Jurafsky and Martin, 2008). In the clinical setting we are more interested in entities such as medicines, diagnoses and events such as procedures and drug usage. We are further interested in finding relevant and important aspects of these entities and events. These include: Negation, speculation and status, whether the extracted entity or event is negated or being speculated ("the test results were **negative**", "the patient **may** have asthma") or about someone else ("the patient has **a family history** of"). Temporality, whether it is a past, current or future event and whether one event occurred before or after another. Relations between entities and/or events, ("tissue damage **located in** right thigh").

Named Entities and Terms

Named entity recognition and classification (NERC) is an essential task when recovering and structuring information from free text. It is also a useful task in Information Retrieval systems, where the aim is to retrieve all documents containing a specified term or entity. A named entity (NE) is, according to Jurafsky and Martin (2008, page 761), "anything that can be referred to with a proper name". We want to investigate the running *tokens* in the text, and uncover which of these are named entities and classify what kind of NE it is. Examples of NE types/classes include *Person*, *Organization* and *Location*.

Generally NERC includes finding names of people, localizations, companies and dates, while more specific approaches are engineered towards finding genes or proteins, etc. When dealing with clinical records it is naturally important to find mentions of medications and medication usage, (including dosage and frequency of use), diagnoses and other medical conditions and findings, operations and other kinds of procedures and of course mentions of the patient itself, family members and possibly medical personnel and perhaps medical institutions and doctors.

Jurafsky and Martin (2008) treat named entity recognition as a sequence classifying task, similar to the machine-learned, stochastic, approach to POS-tagging and chunking explained in the sections above. A gold-standard training corpus is used to train a model to label each token as the beginning of a named entity, inside a named entity or outside a named entity. The model creates statistics over features in the running text (such as surface form, POS-tags, etc.) in order to compute how plausible it is that a named entity starts or ends at a given token in the sequence of tokens. Classification, then, consists of detecting the kind of each named entity (person, city, company, etc.) This approach is, as we will see, not so regular in the clinical domain, possibly because of the lack of training corpora and also possibly because IE systems in the clinical domain typically map a phrase directly to a concept in a terminology or ontology.

A thorough survey of NERC-related research has been performed by Nadeau and Sekine (2007), which describe the development from early hand-crafted rule-base systems, through supervised learning approaches and ending with semi-supervised and unsupervised techniques. In supervised learning we use annotated corpora to train a model which looks at pre-defined features and learns how to distinguish named entities based on these features. The features include the token part-of-speech, frequency or rate of a sequence of tokens (n-gram), digit patterns (for dates, etc.), casing of letters in the token, morphology (for instance tokens ending in "ish" or "an" might imply nationality, such as *Danish* and *Norwegian*), etc.[11]

Supervised learning utilizes a hand-annotated corpora to train a system (usually based on some kind of stochastic model) to annotate named entities. This would typically work in the same manner as in supervised POS-tagging and supervised chunking (Jurafsky and Martin, 2008). Semi-supervised learning is mainly driven by bootstrapping, where the system begins with a small number of unambiguous "seeds" and uses some general clues to estimate other possible entity-mentions in the text. These clues could be grammatical, like

[11]For a full list of fruitful features, see (Nadeau and Sekine, 2007).

sequences of POS-tags and subject-object relations, or orthographical clues like sequences of preceding words, word-bags, capitalization and titles (for instance is "Mr." likely to be a part of a person-entity)(Nadeau and Sekine, 2007).

Complete unsupervised learning mechanisms can be used as well, where no pre-annotated material is used. For instance one could use external vocabulary resources to extract both lists of entities and types (like WordNet). One could also build a named entity classifier by *clustering* groups of tokens given features and clues used in semi- and supervised machine learning. Since annotated material is hard to get by in the clinical domain, and we have access to terminologies and ontologies that can be used, unsupervised learning will be our primary focus when dealing with named entities from now on.

Named Entities and Terms in the Clinical Setting

Supervised learning in named entity recognition — which utilizes hand-labeled corpora during training — remains the dominant method for NERC (Nadeau and Sekine, 2007). Unfortunately there is little annotated data available in the domain of clinical health records, especially so when it comes to NEs. A possible exception is the data described by Ogren et al. (2008). Recently the i2b2 also released their corpus used for the i2b2 2010 shared task, where *concept mentions* and their co-references are annotated (see section 5.3). This could perhaps be used to learn a named entity recognizer (but not a classifier). Nevertheless, the small amount of annotated material in the clinical domain has lead to the fact that most research uses semi- or unsupervised NE-recognition methods.

In clinical free text we are — as we have mentioned — typically interested in entities such as diseases, medications, allergies, procedures and persons. This is admittedly not precisely like *named* entity recognition seeing that we are often interested in entities without a proper name [12]. In this first step of IE, we want to uncover the text-spans that represent one of these types of entities, and also map each occurrence to a concept in the medical terminology. This differs from the typical NE methods; instead of classifying the NE, we want to get direct knowledge about which entity this is in accordance to a terminology. For instance, in a running text like "the patient was treated with aspirin" instead of finding the NE "aspirin" and classify this as a NE-type of "medication", we want to annotate it in accordance to a nomenclature such as SNOMED CT and say that it is |*Aspirin (product)*| with SnomedId C-60320 and/or |*Aspirin (substance)*| with SnomedId F-61BB8.

We find two plausible reasons why this is so: In the clinical domain we typically have several standard terminologies, which may ease the NE process (discussed further below). Second, the main target of NE in the clinical domain is to uncover which concepts are referred to in the text. We are therefore more concerned here with the mapping from terms and phrases in the narrative

[12] We do want to find singular entities and events that can be pointed out, in this sense covering what Nadeau and Sekine (2007) refer to as "rigid designators".

clinical text into a terminology or ontology than finding and classifying *named entities* as such.

Oliver and Altman (1994) performed one of the earlier attempts of clinical term extraction, where they mapped medical terms found in hospital admission summaries to *SNOMED-III*[13] concepts. They used a kind of supervised learning, and built a training and test-set based on electronically stored admission summaries of patients that had congestive heart failure. 197 phrases where used for the training set. These were coupled with the correct SNOMED term resulting in a total of 139 SNOMED terms used.

From the training set, where terms are coupled with SNOMED terms, they had to find some way of mapping new *unseen* phrases to the correct term. That is, they wanted to build a method which given an input phrase returns the SNOMED terms relevant to the phrase. For this task they used a mathematical tool named linear least squares fit (LLSF).

They built two sparse matrices, designated as matrix A and matrix B. A contains 197 columns — one for each phrase that was mapped to a SNOMED concept in the training set — and 365 rows — one for each unique token found in the phrases. An entry A_{ij}[14] is set to 1 if word i is found in phrase j. B is similar, there is 197 columns for each text phrase and 139 rows, one for each SNOMED term. If the SNOMED term i is relevant to phase j B_{ij} is set to 1, otherwise 0.

When doing this, each phrase is a *vector*, mapping a phrase represented by a vector of 0's and 1's into a set of SNOMED terms (also a vector where relevant terms is marked as 1). They then used a method to create a mapping from a vector a (consisting of 1's and 0's) into a vector b, where vector b indicates which SNOMED terms that are relevant, given the tokens found in the phrase that a represents. This could then be used to map phrases from a text into a SNOMED term. A similar approach with phrases mapped to ICD-9 codes was also performed by Yang and Chute (1993).

The system was evaluated against 116 sentences taken from admission histories for five patients which had congestive heart failure. Instead of using phrases, they used entire sentences as input. They could achieve a high *sensitivity* (recall) and lower specificity, 90% and 83% or low sensitivity and high specificity, 42% and 99.9%, depending on how they adjusted their mathematical method. So with this method it is possible to adjust performance depending on whether we want to find as many concepts as possible, or be as precise as possible.

The downside of this mapping mechanism is that it is costly to make a mapping-function for all the terms in a terminology. This procedure depends on a domain expert which assigns relevant terms to the different phrases, i.e. it is a supervised machine learning algorithm. Another negative aspect of this procedure is that the method they developed worked on sentence-level, which means that the system does not identify where in the sentence the SNOMED

[13]For more about SNOMED see section 5.1

[14]A_{ij} means "the content of the cell in row i, column j.

Entity Class	**Examples**
Dosage	40 mg/day
Blood Pressure	105mm of Hg
Demography	69 year-old man
Duration	3 week
Date	May 1991
Quantity	55 mm

Table 3.3: Examples of administration entities (Patrick et al., 2007).

term is represented. This makes it for instance more difficult to identify whether a identified SNOMED term is negated or not.

A more recent approach is performed by Patrick et al. (2007), which attacks all these problems. They mapped free text into SNOMED CT[15] concepts. They pre-processed the narrative with common NLP tasks like tokenization, sentence detection and normalization (down-casing, stemming, spelling variant generation), POS-tagging and chunking. They first extracted what they call *administration entities*, which include entities such as Date, Dosage, Demography and Duration. Regular expressions were used to extract these entities. Examples of administration entities can be seen in table 3.3.

Their system generates all possible n-grams[16], from uni-grams up to the length of the sentence. This is done after stemming and removal of stop words. For each word in the sentence, they collect all SNOMED CT descriptions which contains the word. Then, for each possible n-gram of the sentence, they keep a list of the *intersection* of all SNOMED CT descriptions collected for the word tokens in the n-gram. So now, each n-gram has a list of all SNOMED CT descriptions which contains all the word tokens in the n-gram.

They score each alternative term for each n-gram based on how many tokens there are in the n-gram which can be found in the SNOMED CT description (number of tokens in the n-gram over number of tokens in the description). The winning term is the "candidate" for the given n-gram. Overlaps then need to be dealt with: For instance in the phrase "bacterial pneumonia" the whole phrase (bi-gram) is linked to the description "bacterial pneumonia", but the uni-gram "pneumonia" is also found as a SNOMED CT description. This is done by finding all possible configurations of n-grams not overlapping each other, and scoring each configuration based on numbers of tokens which also exist in the candidate SNOMED-CT terms over number of tokens in the sentence. In the phrase "bacterial pneumonia" then, the description "bacterial pneumonia" gets a score of 1.0 while the phrase "pneumonia" gets a score of 0.5[17]. This means

[15]SNOMED CT is a newer variant of SNOMED III.

[16]All bi-grams over a sentence "the patient had fever" are "the patient", "patient had" and "had fever", three-grams are "the patient had" and "patient had fever", while the four-gram is the sentence in itself.

[17]Remember, though, that we are doing this over whole sentences, and not phrases. "Bacterial pneumonia" is just a convenient example for how we score the different configurations.

that their system prefers mapping maximum numbers of tokens to a term, and that no terms which the text is mapped to overlap in the input-text.

There are two benefits to this procedure. Firstly, it is completely unsupervised. Secondly, it depends on little linguistic knowledge (only knowledge about spelling variants of the tokens is used). It is not dependent upon finding phrases in the running text, so tokenization and normalization are the only pre-processing techniques used for mapping text to SNOMED-CT concepts. [18].

Patrick et al. (2007) argue that constricting the mapper to noun-phrases may increase the precision of the system, but that some concepts might be missed because SNOMED-CT terms cross the noun-phrase chunk. For instance, the SNOMED CT term "Third degree burn of elbow" is chunked "[$_{NP}$third degree burn] of [$_{NP}$elbow]". This could be missed if the system only considered NP chunks. But this could be avoided by using shallow semantic parsing, as described in section 3.1. The system we used in our experiments also identified "Third degree burn of elbow" as a complete NP chunk (as well as treating "of" and "elbow" as chunks in and of themself).

Entity relations

The next step after extracting and classifying the named entities is to discover the relations between them. Take for instance this example sentence from the CMC 2007 corpus (see page 55):

(1) This is a 8 year old with Trisomy 18 with cough and fever.

After finding the entities **patient**[19], **Trisomy 18**, **cough** and **fever** one would want to extract the three relations between **patient** and the syndrome and symptoms. If we operate with the two relations "has syndrome" with |*person*| as domain and |*syndrome*| as range, and "has symptom" with same domain and |*symptom*| as range, an entity relation detection system could report these four relations:

- |*person001*| |*has syndrome*| |*Trisomy 18*|.
- |*person001*| |*has symptom*| |*cough*|.
- |*person001*| |*has symptom*| |*fever*|.
- |*person001*| |*is a*| |*patient*|.

[18]They do state that the text is POS-tagged and chunked as part of the pre-processing steps, but the algorithm they describe does not seem to use this information when mapping concepts to text.

[19]The committed reader may have noticed that the "patient" is not mentioned directly in the text. An IE system should be able to infer that there is a *patient* in this text nonetheless.

The use of the same identifier for the patient underlines the fact that the computer “knows” it is the same person. If one asks the computer “What is the frequency of patients which had a fever and were coughing”, the computer should be able to sum up all different persons with both a |*has symptom*| relation to **cough** and **fever**.

Contextual Features

After finding the named entities of interest, it is fruitful to identify whether the entity has been negated (“the patient did **not** ...”), or whether the uncovered entity describes someone else than the patient (“the patient has a **family history** of ...”). After finding named entities such as diagnoses and procedures, then, we want to look at the *context* of the entity, to affirm or negate its status accordingly.

Chapman et al. (2007) developed an algorithm named ConText especially for the clinical domain. This algorithm uses regular expressions to locate contextual features indicating negation, temporality and experiencer and is an extension of the similar NegEx algorithm for uncovering negations. The negation feature has two values: *Negated* and *affirmed*, the temporality feature has three: *Recent*, *historical* and *hypothetical*, and the experiencer feature has two: *Patient* and *other*.

NegEx locates a *trigger term* (indicating a negation) and a *indexed term* (an annotated entity), where both terms must be within a sentence and there is no more than 5 word-tokens or annotated entities[20] in between. Either the *trigger term* can be in the far left of the scope and the *indexed term* on far right, or vice versa. The trigger term may negate the indexed term if several requirements are met: For instance the word “but” terminates the negation if it appears in between the negation and index terms, and the negation term is on the left (“She denies headache but complains of dizziness.” Example from (Chapman et al., 2007)). For more on the NegEx algorithm see (Chapman et al., 2001).

As said, ConText expands NegEx by also detecting temporality and experiencer. Different trigger terms are used, for instance “father(’s)” indicate that the experiencer value is *other*, “should he” indicate that the temporality value is *hypothetical* while “history” indicate that the temporality value is *historical*. In both NegEx and ConText there is also a list of pseudo-trigger terms, that contains the trigger term, but is not actually a trigger. For negated events this is terms like “no increase” and “not extended” (Chapman et al., 2007).

Unfortunately, not all negations are clear cut, let us examine this example sentence from the i2b2 2008 corpus (see page 57):

(2) She denies dysuria or hematuria.

[20]This means that if several word tokens is annotated as an entity, it counts as “one word”.

Two problems arrive here. First, it is not always the case that we can infer that for instance **dysuria** did not occur even if the word "deny" appear close to it. Take this made-up example:

(3) She denies dysuria to be a big problem.

Second, a patient's denial of an illness does not make it the case that the illness is not present. Whether the IE system should attack this problem is another topic.

Extraction of Temporal Clues

After uncovering entities, events, relations and negations the natural next step is to uncover temporality within narrative clinical text. Uncovering temporal information from text is highly relevant when performing information extraction in general, and is viewed as essential in the clinical setting (Tao et al., 2010). By extracting temporal clues we can perform better analysis of the progression of diseases, different causes of clinical situations and new causal links between events like drug usage and symptoms. For this we need to identify both date references ("underwent surgery on August 9"), reference to future and past ("2-3 weeks"), and relative assertions between events, like in example sentence 4 (taken from i2b2 2008 corpus, see page 57). Dealing with temporality is perhaps the most difficult part of IE, both regarding extracting temporal information and how to store and treat such information on the computer.

(4) One week after his myocardial infarction , he underwent an exercise tolerance test MIBI.

Zhou and Hripcsak (2007) give a review of temporal reasoning with medical data, where we see that extracting temporality from the text is a complex problem. Questions like *Is time continuous or discrete?*, as well as different considerations of how fine grained one should extract temporality and how one should store information about temporality are not yet settled. Zhou and Hripcsak also discuss the structure of time. The ordinary view is a "timeline" where events occur on this line. But repeated actions (like daily or weekly doses of medicine) could be described by circular time, future events could be described by a branched tree (the future may have several outcomes) and parallel time-lines could describe concurrent events. Take for instance this sentence, from the i2b2 2008 corpus (see page 57):

(5) she has also recently completed a course of Levaquin for urinary tract infection

There are some alternatives on how to represent this. The IE system can ignore past events entirely, treating them as non-existing. This may seem viable in some respects; previous treated illnesses would be of little interest for instance when asserting ICD-9-CM codes to the record/patient. In other respects we may miss out on important information. If we do not record the previous use of medications (like Levaquin) or covered illnesses (like urinary tract infection) important causal relations between medications and/or illnesses is left out. One could for instance be interested in the question "How frequently do patients treated with medicine x have symptom y?" (for instance when searching for possible new side effects from drug usage).

A second option is to leave out temporality, but store the information that the patient uses Levaquin and has urinary tract infection. This could lead to important gains of knowledge, but seems hazardous: This is not correct. The best option (if available) seems to be retrieving and storing temporal information one finds in EHR. This means that some sort of temporality-information should be included in almost every entity relation stored in the system. In addition there is the level of granularity. Should we just extract information about past/present/future, or should we extract exact dates (where possible) or something in the middle? It is no doubt, then, that managing temporality is a challenging task, and must be considered when developing IE systems.

Chapter 4

Knowledge Representation

After extracting information we need to store it in such a way that nothing of importance is lost, and such that everything is available for reuse. All this while considering *computability* or *machine readability*, meaning that the computer is able to treat the structured data in a well-defined manner. We also want to be able to capture the meaning, or the semantics, within the text by adding formal constraints such that a mechanical procedure can calculate over the semantic content.

There is not much focus on knowledge representation in our experiments (see section 7), except the plain fact that extracted information needs to be saved and restored. Why, then, devote a whole chapter to this subject? We are motivated to bring in this because it is vital to address the treatment of information in any IE system and because we utilize ontologies which have these capabilities in our work. Extracted information has to be stored somehow, and it is worthwhile reflecting on how. Furthermore, some of the concepts introduced in this chapter will be used later in the thesis, such as *inheritance* and *taxonomy*. It is, for instance, easier to introduce the SNOMED-CT ontology with this chapter as a background.

When adding structure to data, it is beneficial to follow a common standard, so that retrieved information can easier be communicated and used between several parties (Hitzler et al., 2009). Several standards for classification of diagnoses are used in the clinical domain, but we will here focus on SNOMED-CT, which also contains semantic constraints and relations useful when trying to represent and reason over knowledge contained within clinical documents.

It is also valuable to have access to *general knowledge* (as opposed to the specific one we extract) about the domain in which we operate when extracting information. Such general knowledge range from simple terminologies over domain-specific terms, into full-blown ontologies which define entities, relationships between entities and other formalized knowledge about the domain of interest. We are utilizing such general knowledge through the SNOMED-CT Ontology and UMLS Metathesaurus in our experiments. To have a firm grasp of what such models can perform is a strong enough motivation to study the formal aspects of the technologies behind in detail. In this chapter we will therefore look into knowledge modelling, reasoning with knowledge and exchanging information.

The focus will be on three main points from the perspective of Information Extraction. We want to be able to use the stored information, we want to lose as little information as possible when storing new extracted information and we want to be able to communicate the information to other parties while minimizing the amount of information being lost in the process. For this we need a robust way of handling and representing extracted data, and we need a common vocabulary that every communicating party knows. All these aspects are part of how we handle knowledge in the computer.

4.1 Representing information

By "representing knowledge" we here mean the ability to store information with its semantic features, so that we can retrieve precisely the information we seek and so that the computer can reason over the information and infer new knowledge where possible. When we want to query the computer after a specific entity, we must be able to describe this entity unambiguously through an interface. The computer needs to know where these entities are found, i.e. the data needs to be structured somehow. For instance, through an interface the end-user may select a specific medical term from a formalized vocabulary – say |*Bronce diabetes (disorder)*|. The computer can then scan all documents[1] and retrieve those that contain strings annotated with the terminology. Further, the computer could filter out (disregard) documents only containing negated mentions of the concept.

There are many alternatives as to how we can store structured information in the computer. Whilst relational databases are the classic alternative, other options exist. For instance the UIMA framework for processing unstructured information use Typed Feature Structures (see chapter), which is a well-studied data structure used in NLP (Götz and Suhre, 2004). With Semantic Web technologies RDF-triple stores are also becoming more popular, where we also have the ability to semantically constrain the information such that more precise knowledge can be defined (more on this in section 2).

We choose here to use RDF and its semantic extensions RDFS and OWL to discuss how we can store, represent, exchange and automatically reason over structured information. We do this mainly because SNOMED-CT — which we will discuss in the next chapter — is defined within terms from this paradigm, but also because it is easier to introduce the subject of knowledge representation by a standardized apparatus. We also think that semantic technology can be used to store and retrieve new information we extract from clinical documents in a robust and flexible manner.

[1]Instead of "scanning all documents" the computer could of course access a database over all documents containing the term, etc. We are not interested in the technical implementation.

RDF triples

To exemplify how we can capture information in a machine-readable way, we are going to look at Resource Description Framework (RDF), which has gained popularity over the recent developments in the Semantic Web (Hitzler et al., 2009). Originally RDF was designed for modelling metadata, but has evolved into a general modelling method for describing relations between objects of interest. As an assertional language, it can express computer-readable propositions (Hayes, 2004). It consists of subject-predicate-object triples, where the predicate typically indicates a relation from the subject to the object. RDF triples do not carry any semantics on their own; semantics must be agreed upon between producers and consumers of statements made with RDF. More precise formal vocabularies can be developed to express formal semantics that can be mechanically reasoned over (see sections 4.2 and 4.3). The RDF framework is not a serialization (specification on how to store data on disk), but a general description of how to model information as propositions. When used, a specific serialization must of course be chosen (Klyne and Carroll, 2004). As an overall summation, the following quote from a W3C Recommendation may be enlightening:

> RDF has an abstract syntax that reflects a simple graph-based data model, and formal semantics with a rigorously defined notion of entailment providing a basis for well founded deductions in RDF data. (Klyne and Carroll, 2004)

A set of RDF-triples builds up a graph, where subjects and objects are nodes and predicates are arcs between the nodes. Nodes can be resources, either globally named sources via an URI[2], locally named sources ("blank nodes") or, for object-nodes, literals. Literals are specific, constant, values with datatypes such as "string" and "number". These datatypes are not defined with RDF, but rest on the XML Schema Datatypes instead (Klyne and Carroll, 2004). Predicates can only be a resource (that is, not a datatype). With RDF we can then name simple facts by using the predicates as relations between subject and object. If we want to express that a person is 30 years old, we can do this by typing an RDF-triple like in example 1. "ex:John" and "ex:hasAge" are here short forms of the respective URIs, while "30ˆ rdf:XMLLiteral" is a plain literal, representing the number 30.

(1) ex:John ex:hasAge "30ˆ rdf:XMLLiteral"

.

Some properties follow are defined within the RDF standard, we can for instance express that resources belong to a class. We can then (for instance) say that $|John|$ is a type of $|Person|$, or that $|John|$ is a type of $|Man|$. This

[2]Uniform Resource Identifier, defined here as a string of characters for identifying a resource.

has the same effect as saying that a concept is a child of another concept, such that when saying that $|John|$ is a type of $|Person|$ everything that is true about $|Person|$ are also true about $|John|$. We will here deviate from the specifications and focus on what matters for us. We will use the relation $|type\ of|$ to say which class(es) an individual belongs to, and the relation $|is\ a|$ to model parent/child relations between concepts (as we will soon define in section 2).

Barely any semantics is included in RDF. Some of the few things a computer can "know" from an RDF-graph is that when it sees a triple "s p o", then p must be of type "Property", and if the computer sees the triple "s p l" where l is a well-typed XML-literal (according to the XML Schema mentioned above), then l must be of type XML-Literal. This is defined in entailment rules in the W3C Recommandation *RDF Semantics* (Hayes, 2004). We will see more semantically rich frameworks, such as RDFS and OWL, in the next section.

4.2 Modelling Knowledge

We are here interested in how to represent general knowledge about the domain, i.e. knowledge the computer has access to prior to information extraction. We do this with the help of models. By defining classes or concepts (what type of things there are), properties (what kind of relationships that exist between concepts) and individuals (instances of a class/concept) we can describe the parts of the world we are interested in.

The most typical example of a relationship between two concepts is the $|is\ a|$ relationship, which creates a hierarchy of concepts and relations. For instance we could establish the hierarchical relationship between $|Man|$ and $|Person|$ with an expression like 2:

(2) $|Man|$ $|is\ a|$ $|Person|$

A useful distinction to make when modelling knowledge is that between assertional knowledge and theoretical knowledge. Theoretical knowledge is propositions describing relations between concepts, for instance the propositions "bronze diabetes is a type of disease" and "bronze diabetes is a kind of chromatosis"[3]. Assertional knowledge is specific knowledge describing individuals, and we can here equate it with the type of information we are to extract from the clinical notes, for instance the propositions "the patient has hemochromatosis" or "the patient does not have bronze diabetes".

Theoretical knowledge is typically represented in a taxonomy (a hierarchical terminology) or an ontology[4]. Creating taxonomies or ontologies is a task consisting of modelling information, i.e. capturing the relevant domain-specific knowledge. Doing this serves several purposes: The computer can reason on

[3]The relationship is taken from SNOMED-CT.

[4]Technically speaking, there is nothing wrong with expressing assertional knowledge in an ontology, and this is often done for well-known facts (Such as "The Beatles composed Lucy in the Sky with Diamonds").

the basis of the data, for instance if the computer knows that a patient has "bronze diabetes", it can infer that the patient also has "hemochromatosis". Having a common terminology also strengthens the interoperability between different parties, which means that information can be communicated more precisely (see section 4.4).

If the information we extract from the document is coupled with an ontology, the computer is also able to perform reasoning over the information, and thus draw new conclusions hidden in the text. One example of this is the parent/child relations in SNOMED-CT. If the computer extracts the disease $|$*Urethral caruncle*$|$ from a record, and links this to the corresponding SNOMED-CT Concept, the computer does not only "know" that the patient suffers from this particular disease. When one wants to retrieve all patients suffering from $|$*Polyp*$|$ the computer could also return this patient, because it knows from the ontology that $|$*Urethral caruncle*$|$ is a type of $|$*Polyp*$|$.

Terminologies

We will here define a *terminology* as a list of terms, often such that a terminology enlists all important terms used within a given domain. A term is a sequence of words used to name a concept, where each concept can be named by several terms (synonyms and abbreviations). Synonymity will here be understood thusly: Term A is synonymous with term B if they can be used to refer to the same concept. It may be the case that there exists some nuanced difference in the meaning of the synonymous terms in daily usage, but we will simplify this here and generally say that if two terms are synonymous they can represent the same class or concept.

A terminology, then, can typically be used to model how we use different terms to name a *concept*. It can also in some cases model antonyms (terms with opposite meanings). Additional information that could be found in a terminology is what language the term is written in and how frequently the word is being used (in some given set of example texts). Ambiguity could also be modelled in a terminology, by somehow stating that a term can refer to several concepts.

A terminology can, as we have seen in section 3.2, serve as a quite useful tool when extracting entities and events from running text. By providing each concept referred to by a term with a single and unique ID, they can also prove useful when referring to a concept in the computer, as well as distinguish the different concepts.

Modelling Taxonomical Relationships

A taxonomy builds up a hierarchy of the concepts which are defined in a model, and the typical relation is named $|$*is a*$|$ or $|$*is a child of*$|$. That a concept $|A|$ has an $|$*is a*$|$ relation to a concept $|B|$ means that every individual in the domain of the model which is an instance of $|A|$ is also an instance of $|B|$. This includes the fact that all properties of $|B|$ are also properties of $|A|$. The reverse relation of |*is a child of*| is, naturally, |*is a parent of*|, meaning if |*A*| |*is a parent of*| |*B*| then |*B*| |*is a child of*| |*A*|.

If a concept does not have any parents, we call this the *root* concept, and if a concept does not have any children we call it a *leaf* concept. There are some modelling rules which apply to taxonomical trees. There should only be a single root (which does not have any parents), and all concepts should have the root as an ancestor. Furthermore, there should be no cycles in such a hierarchy. Some taxonomies only allow one parent for each concept, and we call this *single inheritance*. When a taxonomy allows several parents for one concept it is called *multiple inheritance*. Similarly, in some taxonomies a concept is not allowed to have just one child, but must have zero or several children (Bodenreider et al., 2007).

As we have said, if "|*A*| |*is a*| |*B*|", then all properties of |*B*| are also properties of |*A*|. This also includes the fact that all *members* of |*B*| are members of |*A*|. |*is a*| is a so called *transitive* relation: If "|*A*| |*is a*| |*B*|" and "|*B*| |*is a*| |*C*|", then all properties of |*C*| are also properties of |*B*| and |*A*|. Some operate with a nuanced difference between the |*is a*| relation and the |*is a child of*| relation. For instance, when |*A*| |*is a*| |*B*| we only know that all instances of |*B*| also are instances of |*A*|, but it could be that |*A*| is identical to |*B*|. On the other hand, if |*A*| |*is a child of*| |*B*|, then |*A*| and |*B*| are not identical (a more formal specification is found in Bodenreider et al. (2007)). Here it is sufficient to treat these relations as equals.

Another kind of a hierarchical structure, called mereology, is built up by the |*part of*| relation[5]. The relation |*part of*| can for instance model that |*Hip region structure*| is a part of |*Entire lower limb*|[6]. In this way we can define localizations of the different body-parts within a human, as well as knowledge like that a |*Window*| is a part of a |*House*|. Similar principles apply here as in taxonomies, such that the hierarchy defined with the relation has a single root, that the relation is transitive, and that there should not exist any cycles in the hierarchy (Bodenreider et al., 2007).

Modelling with Ontologies

Often we want to define concepts (or classes) by other relations than hierarchical ones. More complex models can be designed as well by defining types of relations that can be used to capture even more in-domain knowledge. When making models like these we typically call them ontologies. We then need some way of defining the semantics of different relationships as well as the concepts involved in the ontology. One example of a computer language capable of building complex ontologies is the Web Ontology Language (OWL), based on the formalities in Description Logic (DL). We will here see what capabilities exists in OWL, following a standard point-wise approach by first introducing RDF Schema.

[5]The knowledge about part and wholes is called mereology.
[6]Taken from SNOMED CT.

We are not interested here in defining or describing the formal nature of ontologies, we are considering what kind of possibilities that lies within a full-fledged ontology. Later we will see how SNOMED-CT utilizes these possibilities (see section 5.1). Bear in mind that for the computer, to be able to calculate over the semantics, we need formal definitions of the constrains and relations used. Readers are here referred to the W3C standards on RDFS and OWL 2 (Hayes, 2004; OWL Working Group, 2009).

RDF(S) and OWL is designed around the open world assumption, as opposed to the closed world assumption used in the traditional databases. Simply put, in the open world assumption we assume that we cannot know that something is *not* the case, even if it is not stated in the knowledge base. Implicitly we assume that the knowledge is incomplete. The closed world assumption assumes that everything we know is in the data base, for instance if person x is not in the customer data base, we can assume that x is not a customer.

The open world perspective seems to be a healthy one when automatically extracting information: That we didn't extract some given information — say, our system do not say that patient x got allergy y — does not mean it is not the case. The patient could have the allergy; perhaps is wasn't mentioned, or that the software failed extracting this piece of information from the note.

RDF Schema (RDFS)

RDFS constrains RDF such that more precise, but still computable, propositions can be made. For instance, RDFS defines *domain* and *range* of predicates, specifying what types that are allowed as subject and object to the predicate. For instance, say that we have a relation $|has\ diagnosis|$, we could specify that the class $|Person|$ is the domain, meaning that only persons can be diagnosed[7]. Furthermore we could say that the *range* of the relation is $|Diagnosis|$ (which then means that a patient can only be diagnosed with a diagnosis).

With RDFS we can also define hierarchical relations between classes and properties/relations, such as those discussed above. As properties are hierarchical they will contain parent/child relations and the properties of a parent will always be inherited by the child. This means for instance that if a property $|p|$ has a domain $|C|$ and that property $|q|$ is child of property $|p|$, then $|q|$ also has the domain $|C|$. More generally, everything we say about a concept or a property is also true for its child. We can also explicitly say that a resource is a class, RDFS gives us such a vocabulary. Plain RDF already gives us the means to state that a resource is a property.

Web Ontology Language (OWL)

OWL 2 is divided into several sub-languages, which offer advantages with regard to computational complexity (how much time and space is needed to perform reasoning over the modelled knowledge) (OWL Working Group, 2009).

[7]For a veterinarian clinical setting this would, of course, be altered to $|Animal|$.

We will not consider these sub-languages here, but rather see in a more general way what kind of semantics OWL 2 is able to represent.

OWL 2 defines three basic notions used for making ontologies: *Axioms* which are the fundamental expressions of the ontology, *entities* which represent real-world objects and *expressions* which are formulated by combining the *entities*.

With OWL 2 we are able to, amongst other things, express that two classes are disjoint, which means that no members of one class is a member of another. A typical example would be that $|Man|$ is disjoint to $|Woman|$. Among other things possible to model with OWL is that a property does *not* hold between individuals (for instance that "$|Mary|$ is not married to $|John|$") and that two individuals are identical or not identical (for instance that "$|John|$ is not $|Billy|$" and that "$|Andy|$ is $|Andrew|$") (Hitzler et al., 2009).

With OWL 2 we can also define more complex classes. For instance, we can define that a class $|X|$ is an intersection of two other classes $|A|$ and $|B|$, meaning that every individual belonging to both classes $|A|$ and $|B|$ belongs to class $|X|$. A typical example would be that a $|Mom|$ is the intersection of $|Parent|$ and $|Women|$. A class could be defined as a union of two classes, such that individuals that are either a member of $|A|$ or a member of $|B|$ is a member of $|X|$, for instance a $|Parent|$ is a member of $|Mom|$ or $|Dad|$. A complement class of a class $|A|$ consists of every individual *not* in class $|A|$ (Hitzler et al., 2009).

Furthermore we can restrict properties as a way of defining classes. We can for instance express that "every person which has a child must be a parent". This is called *existential quantification*, where being in a particular relation to a type of individual includes membership to a class. A similar restriction is the *universal quantification*. For instance, we can model that a person is a member of the class $|HappyPerson|$ only if all his children are members of the class $|HappyPerson|$[8]. We can also add *cardinality*-restrictions, defining how many relations there can or must exist between individuals or classes (Hitzler et al., 2009).

4.3 Reasoning with Knowledge

Standardizing and formalizing the way humans reason gives us new computable capabilities to expand on previous knowledge. As we have seen, making hierarchical relations between concepts already gives the computer the means to deduce new information. For instance, when $|A|$ $|is\ a|$ $|B|$ and $|x|$ is a member of $|A|$ the computer can deduce that $|x|$ is a member of $|B|$. The same kind of logic applies to mereological hieararchies. If $|C|$ $|part\ of|$ $|D|$ and $|y|$ is located in $|C|$ the computer can deduce that $|y|$ also is located in $|D|$. These are safe, well-defined, ways of extending the knowledge in a computer database.

[8]Formally one needs both the universal and the existential constraints to model this, if not it would be sufficient to *not have any children* to meet the universal constraint.

If we have more complex ontologies, not restricted to merely describing the hierarchical (parent/child) relations between the concepts, even more new information can be automatically deduced by the computer. We can see this for instance in the SNOMED-CT ontology, where the relation |*CAUSATIVE AGENT*| links |*hemochromatosis*| with |*Iron AND/OR iron compound*|. The computer can then deduce that the patient which suffers from hemochromatosis has been exposed to iron. There is also an obvious hazard in deducing new information this way. If the algorithm we use to extract information from a clinical note fails, and the extracted information is incorrect, the consequence may be that not only is this piece of information wrong, but all further information the computer infers from it is wrong. We will not discuss these issues at length here, but it is important to bring them into attention. On the face of it, it seems that the more insecure the information extraction process is, the more hazardous it is to let the computer deduce new "knowledge" based on the extracted information.

Another, perhaps obvious, hazard is expanding negated propositions — such as "John does not have bronze diabetes" — with help of the hierarchical relations in the same manner we do with positive findings. We see that the parent of |*Bronze diabetes (disorder)*| in SNOMED CT is |*Hemochromatosis*|, but can we infer that John does not have hemochromasotis? No, for instance it could be that John has bronze cirrhosis, and |*Bronze cirrhosis*| is also the child of |*Hemochromatosis*|. But negated entities can be expanded the other way: If we know that John does not have hemochromasotis, we also know that he does not have bronze diabetes nor bronze cirrhosis.

The question of what kind of new information that can be deduced is closely related to the restrictions in the formal language we use for modelling knowledge. When two classes $|A|$ and $|B|$ are defined as disjoint, and we know that an individual $|a|$ belongs to $|A|$, the computer can infer that $|a|$ does not belong to $|B|$[9]. If a class $|C|$ is a intersection of two classes $|A|$ and $|B|$, and a individual $|a|$ is a member of both $|A|$ and $|B|$, the computer can deduce that $|a|$ is a member of $|C|$.

If we have an ontology defining diagnoses with respect to findings and symptoms, making use of complex ontological statements containing intersections and cardinality restrains, we imagine it possible to use the computer for automatically assigning diagnoses with respect to uncovered findings and symptoms.

With all the above as a background, it is fairly easy to explain how the computer deduces new information. We simply mean that we can state some rules explaining how new propositions can be inserted into a database, based on propositions already in it. In the example below, we have two statements already stated in a database (p1 and p2) and a piece of new knowledge that can be automatically inserted to the database by the computer (c).

p1 $|A|$ $|is\ a|$ $|B|$

[9]Remember that with the open-world assumption, we cannot determine that $|a|$ is not a member of $|B|$ merely on the ground that this is not explicitly stated.

p2 $|a|$ $|\textit{type of}|$ $|A|$

c $|a|$ $|\textit{type of}|$ $|B|$

The computer can iterate through the database, and see if there are is set of statements that constitute the deduction of a new statement, according to rules defined within RDF, RDFS or OWL. Each time the computer discovers new knowledge, it can then insert it into the database and see if this new knowledge again spawns even more knowledge. Some caution must be made, so we do not enter into a state where the computer never stop deducing more statements.

4.4 Exchanging Knowledge

If two or more parties are to communicate, they need three things: A link or medium to communicate through, a protocol (set of rules) for how the communication is to proceed, and the ability to understand the content of what is being communicated.

Ontologies bring us the ability to understand not only which specific terms are being used in the text, but also the underlying semantic content of the message. By linking each concept or entity, as well as every relation that can occur between concepts or entities, to a unique identifier and/or definition, we can interchange data in such a way that all parties understand what is being communicated. We call this semantic interoperability. Berges et al. (2010) explains semantic interoperability in the following way:

> In general, semantic interoperability is defined as the ability of one computer system to receive some information and interpret it in the same sense as intended by the sender system, without prior agreement on the nature of the exchanged data. (2010, page 11)

We will not consider semantic interoperability further, we are primarily concerned about extracting the information.

Chapter 5

Existing Resources

5.1 Ontologies and Terminologies

Using medical ontologies when representing the semantics in medical texts offers several advantages when reasoning over and extracting information from clinical documents. First of all it is a unique resource to clearly stated, language independent, non-ambiguous marking of the medical terms. Given that we find some concept or description from unstructured clinical text in a terminology, we attain partly in-depth knowledge of the subject-matter in the text. Secondly a precise ontology gives an exceptional ability to make high-level comparisons of symptoms and diseases due to the logical definition of the terms. For instance it is possible to automatically make comparisons between different procedures of *bone immobilizations*, both those recorded as *cast immobilization of fracture* as well as *closed* and *open reduction of fracture*, since they are likely both to be defined as types of *bone immobilizations* in the ontology.

We have already seen some examples of using terminologies and ontologies in information extraction (Oliver and Altman, 1994; Patrick et al., 2007). There are different resources to choose from, but one of the most promising alternatives seems to be SNOMED CT. There are other resources available as well, such as ICD9-CM, but SNOMED CT is in contrast far more granulated and complex. We will therefore here focus on SNOMED-CT, and also mention a few other existing ontological resources. In particular we will describe the UMLS Metathesaurus since we have used this in our experiments.

SNOMED CT

SNOMED CT is a comprehensive resource of terminology within the domain of health-care. The purpose is to have reliable clinical information within software applications, securing communication between different systems and in different languages. SNOMED CT can maintain input in electronic information systems in a standardized form, so that the data can be reused for different purposes. The development of SNOMED CT occurred within a native formalism of DL (Bodenreider et al., 2007). 311.000 *concepts* related to millions of *descriptions* (terms) ensure unambiguous documentation in electronic patient records and other health-related documents.

The building blocks of SNOMED CT are concepts, descriptions and relations. A *concept* has a unique clinical sense, coupled with a distinctive and static identifier, and each concept is logically defined by *relations* to other concepts within the SNOMED CT terminology. Each concept is represented by three kinds of *descriptions* (human readable names); fully specified name, preferred term and synonym (where a concept can have zero to several synonyms). All concepts must have one fully specified name, which is exclusive for the concept and one preferred term (IHTSDO, 2012). The extensive list of different descriptions of concepts makes SNOMED CT interesting when dealing with named entity recognition and term extraction. Having access to different ways of spelling out the term helps us when we are using NER methods which do not rely on machine-learning.

SNOMED CT Relations

Relations, also called *attributes*, are used to link concepts in SNOMED CT to each other. Four different kinds of relations exist (defining, qualifying, historical and additional) but we will here focus on the relations used to model and define the concepts. Some attributes have hierarchical relations to each other, such that one attribute can be more general or specific (granular) than another one (IHTSDO, 2012).

The most defining relation is the |*Is a*| relation, building a taxonomical tree over all concepts within the ontology. Mereological relations are used when defining the different parts of the human body. Other defining relations include |*Associated morphology*| and |*Finding site*|. 50 different attributive relations are also used, defined both with domain and range, and the attributes are also defined within a hierarchy in itself.

We know from chapter 4 that when a concept |*B*| is a child of another concept |*A*|, |*B*| inherits the definitions of |*A*|. But when the relations/attributes are defined within a taxonomical relationship we can constrain the definition of |*B*| to more granular versions of the attributes used to define |*A*|. This is also the case in SNOMED CT which defines several attribute hierarchies. For instance the relation |*Associated with*| has three children: |*Causative agent*|, |*Due to*| and |*After*|.

In figure 5.1 we see an incomplete graph of gout disorder[1]. We see that |*Gout*| is defined as a subtype of |*Disorder of purine metabolism*|. Gout also has the explicit qualifying relations |*Clinical Course*| and |*Severity*| (not depicted in the figure), which it inherits from its ancestor |*Clinical finding*|.

[1]This information was drawn from the UMLS Terminology Services, SNOMED CT Browser 2012

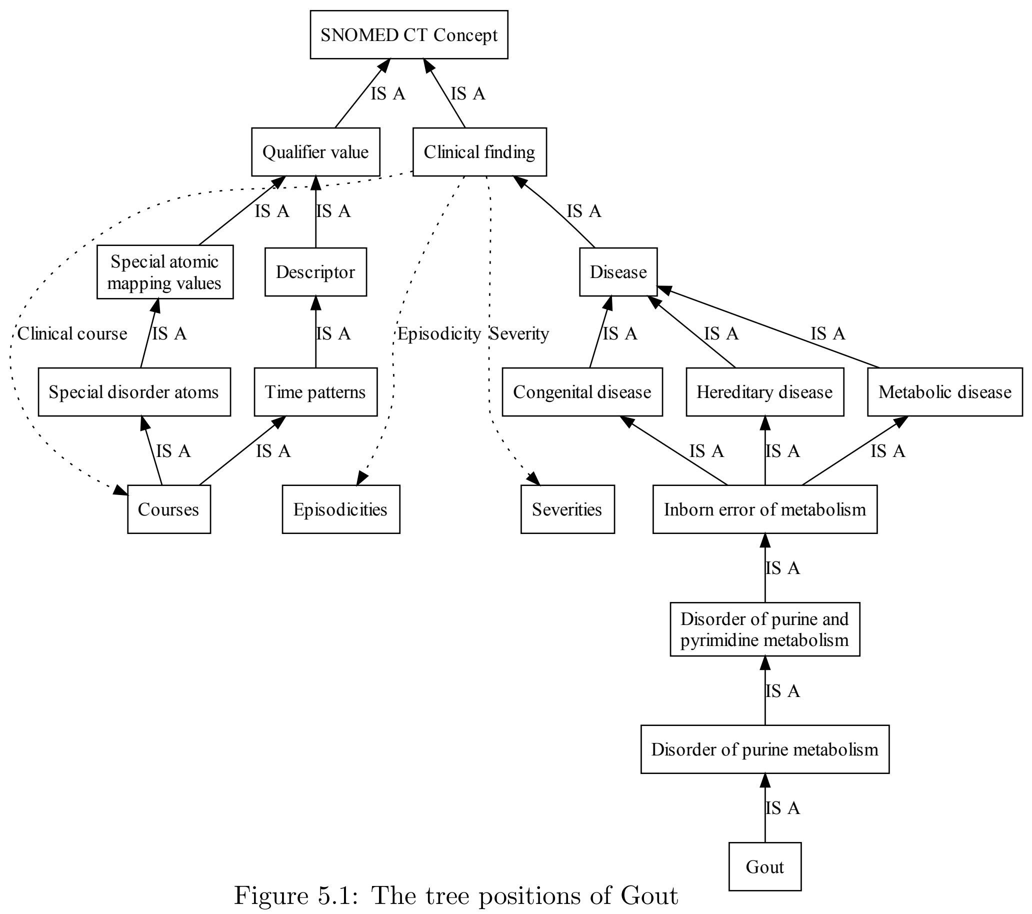

Figure 5.1: The tree positions of Gout

Concepts of SNOMED CT

The concepts are organized in a hierarchical manner, below 19 top level concepts. This hierarchy is defined with the taxonomical |*Is a*| relation. Each concept also has a formal DL-description, consisting of a listing of properties belonging to them. The properties could for instance be the unique identifier, parents (|*Is a*|) and names (Bodenreider et al., 2007). To get a nice overview of all the concepts, it is helpful to look at the top level concepts. These can be found, in lexicographical order, in figure 5.2 (IHTSDO, 2012).

Each concept in SNOMED CT is related to a *status*, revealing whether the concept is in active use or not. When a concept is found to be erroneous, ambiguous, etc. it is replaced with a new concept. The old one is kept, but changes its status accordingly.

Body structure Normal and abnormal anatomical structure.

Clinical finding Result of clinical observation, assessment or judgment

Environment or geographical location Environments (as intensive care units) and locations.

Event Occurrences, excluding procedures and interventions (flood, earthquake, etc.).

Linkage concept Concepts used to link two or more codes to express compositional meanings.

Observable entity A question or procedure which produces an answer or result.

Organism Animals, microorganisms, plants etc.

Pharmaceutical/biologic product Drug products (as opposed to their chemical constituents, substances).

Physical force Physical forces that can play a role in injury.

Physical object Natural and man-made objects.

Procedure Activities performed in the provision of health care.

Qualifier value Concepts used as values which are not a subtype of another top level concept.

Record artifact For instance patient records and also more fine-grained sections of records.

Situation with explicit context Conditions and procedures that have not yet occurred (future history), refer to someone else than the patient (family history), or that have already occurred (past history).

Social context Family status, economic status, ethnic and religious heritage, life style, occupations, etc.

Special concept Concepts no longer active in the terminology.

Specimen Entities obtained for examination or analysis.

Staging and scales Assessment scales, tumour stages, etc.

Substance Active chemical constituents of drugs, food and chemical allergens.

Figure 5.2: List of top level concepts in SNOMED CT (2012)

Unified Medical Language System

The Unified Medical Language System (UMLS) provides access to many controlled medical thesauri, ontologies, terminologies and classifications (we will refer to all of these as "vocabularies" from now). The collection of these vocabularies is designated as the UMLS Metathesaurus. All concepts within UMLS are also assigned one (or more) semantic category, this information/interface is provided through the Semantic Network. Further, UMLS provide NLP facilities through the SPECIALIST Lexicon.

The UMLS facilities is designed and maintained by the US National Library of Medicine. It is free to use the system, but licensing requirements of the source vocabularies still applies when accessing those through UMLS.

UMLS Metathesaurus

The UMLS Metathesaurus is a large multi-lingual vocabulary built up from several vocabularies from the medical domain. It contains over 1 million biomedical concepts which are collected from over 100 source vocabularies. This includes ICD-10, SNOMED CT, and RxNorm [2]

The Metathesaurus is focused around the concepts derived from the source vocabularies, with their attributes intact. Each concept belongs to a UMLS semantic type, as defined in the UMLS Semantic Network. Synonymous concepts across source vocabularies are also linked to each other. UMLS does not present a coherent ontology, but keeps all inconsistencies that may exist between different terminologies and ontologies encapsulated in the metathesaurus.

UMLS Semantic Network

The Semantic Network can be seen as a layer on top of the source vocabularies, grouping the Metathesaurus concepts into different semantic types and adding new relationships between concepts. Thus the Semantic Network categorizes the concepts and adds new formal semantics to them via the semantic types. The relationships are defined between the types, and there are 133 types and 54 relationships. All concepts are assigned at least one semantic type, but some concepts belong to several semantic types where appropriate (UMLS, 2009).

The different semantic types are hierarchically related to each other, for instance the semantic type |*Chemical*| is a subtype of |*Substance*|. The network is divided into ENTITIES and EVENTS. ENTITIES have two top-level types |*Physical object*| and |*Conceptual Entity*|. The |*Physical object*| type has four subtypes: |*Organism*|, |*Anatomical Structure*|, |*Manufactured Object*| and |*Substance*|. ENTITIES have the two top level types |*Activity*| and |*Phenomenon or Process*|.

[2]See a full list at `http://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/release/source_vocabularies.html` (last entered 08.17.12).

The division of types in the hierarchy continues in the same manner, where types important for the medical domain typically are more granular. This can be witnessed in the sub-types of |*Manufactured Object*|, which is |*Medical Device*|, |*Research Device*| and |*Clinical Drug*|. There exist, of course, many manufactured items which does not fit these more granular types. These would then belong directly under |*Manufactured Object*|.

The relations that can hold between the semantic types consist of the typical |*IS A*| relation, defining the taxonomical hierarchy plus five other different types of relations: |*physically related to*|, |*spatially related to*|, |*functionally related to*|, |*temporally related to*| and |*conceptually related to*|. These relations types function as relations in themselves. Amongst the relations we find for instance the mereological |*part of*| relation[3].

The relations are defined to hold across semantic types as high in the taxonomical structure as possible. As usual, the relations are inherited by the children of the semantic types, but in UMLS there is an added possibility of blocking inheritance. The UMLS Reference Manual exemplifies this with the fact that the type |*Mental Process*| would be linked to |*Plant*| via the |*process of*| relation by inheritance. Since plants are not sentient, this is specifically blocked.

UMLS Specialist Lexicon

The UML Specialist Lexicon (Browne et al., 2000) is a general English lexicon which includes terms used in the biomedical domain. It records both syntactic, morphological and orthographic information. With the lexicon one can gain access to all possible POS-tags of the given word, and for each POS-tag the lexicon includes information about the different inflectional variants of the word. For each word, it also includes spelling variants, for instance for the word "anaesthetic" the lexicon provides the alternative spelling "anesthetic".

The SPECIALIST lexicon tools have been used as a linguistic resource in everything from spell-cleaning (Tolentino et al., 2007)[4] to providing normalizations of the running tokens in a text[5] (Savova et al., 2010).

5.2 NLP and IE Systems

Several systems for information extraction from clinical free text already exist, and some of them will be described here. The descriptions are primarily based on the research papers written by those responsible for developing the systems. We will introduce four systems, two of historical interest, a modern open source project and a text-summarizing system for clinical free text in Japanese. Linguistic String Project and MedLEE are interesting systems because their creators pioneered the idea of using NLP technology to extract

[3]For a full list, see `http://www.nlm.nih.gov/research/umls/META3_current_relations.html` (last entered 07.17.12).

[4]Discussed in section 3.1 from page 3.1.

[5]Will be discussed in section 6.2.

and structure information in clinical free text. The newer open source project cTAKES is interesting because it is publicly available and serve as useful tool when investigating and developing methods and algorithms for IE in the clinical narrative. A notable example of a system we did not cover is the open source system Ontology Development & Information Extraction (ODIE)[6], which attempts to use ontologies to extract information from clinical records as well as using clinical records to enhance ontologies.

Linguistic String Project

One of the earliest attempts of capturing information in clinical free text to store in databases evolved around the Linguistic String Project. According to the homepage of Naomi Sager[7] — the project's initiator — the project took place in the period from 1965-2005. Sager et al. (1994) describe an early five-module system, where module one creates syntactical analyses over each input sentence. Module two filters out semantically incorrect syntactical analysis. Module three extends statements with conjuncts, such as "pain in arm and leg" to "pain in arm" and "pain in leg". The last two modules structure the extracted information so that it can be mapped into a database. This mapping was evaluated in accordance with information needs in asthma management, a pre-defined set of features was extracted. The precision was 82.1% and recall was 82.5% (they got higher figures when only counting major omissions). They also made early experiments of mapping free text into SNOMED III concepts (Sager et al., 1994).

The Linguistic String Project has a long history of research and development of NLP systems in the medical/clinical domain[8]. An interesting continuation of the Linguistic String Project is an experiment performed on patient records written in Dutch. Spyns et al. (1998) describe how they could use the output of a morpho-syntactic analyser for clinical Dutch text as input to the language independent modules of the now matured Medical Language Processor of the Linguistic String Project. They used this to build an information extraction system. This work effectively shows that it is feasible to couple language-specific NLP modules (analysing morphology and syntax) with domain specific information extraction modules (using formats to select information pieces such as "patient", "diagnosis", etc.).

MedLEE

MedLEE (Medical Language Extraction and Encoding System) is a text extraction system which extracts and encodes information from free text in patient

[6]`https://wiki.nci.nih.gov/display/VKC/Ontology+Development+and+Information+Extraction+%28ODIE%29` (last entered 31.07.12)

[7]`http://www.cs.nyu.edu/cs/faculty/sager/` (last entered 28.06.12).

[8]An overview of the project can be found at `http://www.cs.nyu.edu/cs/projects/lsp/pubs/MLPPubs_Annotated.html` (last entered 27.06.12).

records. Originally it was designed to process radiological reports at Columbia-Presbyterian Medical Center and was soon extended to handle mammography reports as well. The system maps extracted information into structured databases, where it is used by automated processes such as decision-support systems (Friedman et al., 1995). MedLEE is still available through licensing.

The system[9] consists of five modules arranged in a typical pipeline-manner. Module one is the pre-processing phase, which consists of discovering the free text sections in the report and prepares the free text for further processing. Module two parses the sentences with a semantic grammar. The third module regularizes the parsed sentences to fit the vocabulary better before encoding. For instance in the phrase "heart appears to be enlarged": The parser discovers "enlarged" as a central finding, and "heart" as a body-location modifier, and the regularizer transforms the phrase to "enlarged heart" to better fit the vocabulary used for encoding. The encoding is performed in the fourth module, which maps the regularized phrases into unique concepts in a clinical dictionary. A last module transforms the newly structured material into a standardized format (Friedman et al., 1995).

Three early initial evaluations were performed. The first evaluation considered the parsing-module. 3354 sentences, filtered in such a way that only sentences containing information of interest, were parsed. 86% of the sentences parsed correctly, which means that some analysis (possibly incorrect) could be given. Unsuccessful parses occurred mainly because the sentences contained spelling errors and out-of-vocabulary words (6%) or semantic patterns not recognizable by the parser (5%). Of the successfully parsed sentences, they saw that 2% were not structured correctly. The precision and recall for correctly structured sentences was 99% and 89 %(Friedman et al., 1995).

The second initial evaluation consisted in identifying four clinical diseases[10] from 230 randomly selected reports. The reports were hand-annotated by three medical experts for the diseases as well as processed with MedLee. They then made queries designed to retrieve the documents marked with the diseases. The average precision and recall figures were 87% and 70% respectively (Friedman et al., 1995).

A third evaluation was also performed in a similar manner, but with six diseases[11] and annotation done by twelve medical experts. Three of the diseases had a high precision and recall[12] while the other three had low precision and recall. They achieved better results when augmenting the queries used to retrieve the correct documents with more terms (at the risk of lower precision) (Friedman et al., 1995).

[9]We are here following a paper from 1995 (Friedman et al., 1995).

[10]These four diseases were: Neoplasm, chronic obstructive pulmonary disease, acute bacterial pneumonia, and congestive heart failure.

[11]The two extra diseases were: Pleural effusion without chf, and pneumothorax.

[12]This was not recorded in the paper, but they report that 26 reports were annotated for **congestive heart failure**. The system retrieved 25 of these, with an addition of two incorrect reports.

There has been ongoing development on MedLEE since the paper from 1995 was published and the above descriptions might not apply to the modern version of MedLEE. A more recent description can be found in Sevenster et al. (2012). They used the output of MedLEE to find correlations between clinical findings and body locations. From the structured output they wrote several rules, annotating uncovered body locations and clinical findings as correlating with each other. If they could not find the information straight forwardly from the MedLEE output they used an additional rule which annotated a correlation between a finding and a body location within sentences, given that only one body location was mentioned in the sentence. They also enhanced their system with frame filler software to better detect breast locations.

Sevenster et al. (2012) evaluated their system on 660 radiology reports and 119 breast cancer reports. They found that only using the output from MedLEE yielded a higher precision, but with a lower recall. Recall was overall quite low, ranging from 20-45 % while precision ranged from 83 - 100 %, depending on whether they evaluated on breast cancer reports or radiology reports, and whether they only used the output from MedLEE or also included their sentence-rule. The use of the frame filler also yielded better performance when working on the breast cancer reports (Sevenster et al., 2012).

TEXT2TABLE

TEXT2TABLE is a text-summarizing system for clinical free text, based on entity recognition and negation identification (Aramaki et al., 2009). The program extracts medical events and times from a clinical document, tries to detect whether the event actually has taken place and summarizes the findings for the end user. Even though the system is trained on clinical free text in Japanese, the methods described are independent of language.

Four steps are involved in TEXT2TABLE: The system first marks six different types of events (operation, test, disease, medication, patient action and other verb). This part resembles NERC. Secondly the discovered terms are normalized, for instance acronyms are disambiguated and extended. Thirdly, relations between events and date-times are established, and finally the negative events are located.

The negative events are divided into eight categories (modalities): Negation, future, purpose, suspected, necessity, intended by patient, possible and recommended. An event can be classified as several of these, for instance “No chemotherapy is planned” (example from the i2b2 2008 shared task, see page 57) is in this system regarded both as a negated event and as a future event. For evaluating negation detection 435 Japanese discharge summaries were collected and annotated for events and modality. All events are assumed correctly identified, and a SVM-classifier was used with a 10-fold cross validation for evaluation experiments. When treating all eight modalities as negated events, they reached an F-score of 85.81 (precision 89.40 and recall 82.50). When classifying the different modalities independently, results were considerable lower, mostly because of the small amount of training data.

clinical Text Analysis and Knowledge Extraction System (cTAKES)

cTAKES is an open-source NLP system for information extraction from clinical free text (Savova et al., 2010). It includes components for dealing with tokenizing, sentence segmentation, token-normalizing, POS-tagging, shallow parsing and a NER-annotator. The package comes with ready-made classifier models for the different tasks, developed with supervised training on hand-annotated clinical free texts. However, these annotated text-resources are in general not available due to copyright and patient confidence issues. Therefore, other resources are necessary for training new models and performing evaluations. Technical details about cTAKES are discussed in more detail in section 6.2.

cTAKES processes a clinical document incrementally through a pipeline, adding new information about the document for each step. The free text is first sentence segmented and tokenized, the tokens are then normalized, POS-tagged, shallowly parsed and then annotated with named entities, which are contextually annotated with negation (confirmed, negated or possible) and status (current, history or family history). Because of the modular structure of cTAKES, new modules can easily be developed and inserted by other parties.

The named entity recognition (NER) module was evaluated by Schuler et al. (2008). They used a corpus of 160 clinical notes manually annotated with 1957 discovered mentions of clinical entities. The NER module uses noun-phrases as look-up windows, and retrieves SNOMED-CT and RxNorm concepts via UMLS. When the look-up window exactly matched the span of the annotated term, they achieved a recall of 0.63, precision of 0.51 and F-score of 0.56[13]. Computing the same metrics with partial matched they got a recall of 0.76, precision of 0.88 and an F-score of 0.81. Schuler et al. (2008) found that spelling errors, incorrect assignment of POS-tags, incorrect chunking and ambiguous terms and abbreviations was typical sources of errors when finding named entities.

5.3 Shared Tasks and Corpora

Most of the publicly available clinical reports that can be freely used for scientific purposes are released via different shared tasks. These shared tasks have played a key role in the development of and research into information extraction in the clinical narrative. Naturally, hospitals and clinics are cautious about releasing sensitive data to researchers outside their institutions (Chapman et al., 2011), so getting hands on training and evaluation material is difficult in the clinical domain. The need of keeping patients data anonymous is addressed in the i2b2 shared task from 2007 (Uzuner et al., 2007), where one task of the shared task was automatic de-identification of clinical documents.

[13]These metrics are explained from page 72.

The lack of annotated data makes progress slow when developing NLP tools for the processing of clinical records. Annotated material makes it possible to evaluate and compare different systems and algorithms. The shared tasks presented below are the few resources available. With publicly available annotated material it is possible for scientists and other developers of IE systems for the clinical domain to measure their progress and compare with other systems and algorithms.

The CMC 2007 Challenge

In 2007 the Computational Medicine Center (CMC) held a challenge where the participants were to assign ICD-9-CM codes to clinical free text (Pestian et al., 2007). From several radiology reports they collected the two critical parts of the document for ICD-9-CM assignment; "clinical history" and "impression". Three different companies[14] annotated the documents with ICD-9-CM codes. In the available training material it is possible to see what codes the different companies assigned to the different documents. There is also an annotated code marked "MAJORITY", which simply means states the majority of the companies assigned. The majority is used for evaluation.

For evaluating the systems Pestian et al. (2007) used a micro-averaged F-score[15]. The winner achieved an F-score of 0.8908% and the top 21 systems had scores from 0.81% and up. The common approach of the participants was to find representative features in the records and use these in some kind of machine-learning. The best system used C4.5 decision trees, outperforming newer and mathematically heavier algorithms.

The BioScope Corpus

The BioScope corpus (Vincze et al., 2008) consists of documents from the biomedical domain annotated with negations and speculations. There are three different types of text: Medical free text, biological full papers and abstracts of such papers. We are here interested in the medical free text, which more specifically consists of radiology reports. This is the same material as used in the CMC 2007 Challenge.

1954 radiology reports with 6383 sentences are marked with 877 negation cues and 1189 "hedge" cues (speculation). Vincze et al. (2008) report that the "impression" part of the document is denser with respect to negations and speculations. The corpus was used at the CoNLL-2010 shared task (Farkas et al., 2010), but the clinical data was not used as far as we can tell.

[14]The three companies providing annotations were Cincinnati Childrens Hospital Medical Center and two anonymous parties (likely domain experts).

[15]We describe the evaluation metric F-measure from page 72.

i2b2 Shared Tasks

i2b2 2007 Shared Task

There were two i2b2 challenges in the first shared task of i2b2. In the first challenge the participants were to build systems for automatic de-identification (Uzuner et al., 2007). In the second challenge the participants were to build systems which could identify whether the patient was a former or current smoker, or non-smoker based on medical records (Uzuner et al., 2008).

De-identification is an important aspect of using and sharing clinical records for research purposes. The removal of private health information (PHI) is necessary before sharing data, and the building of systems which could do this automatically would be beneficial in this regard. In the i2b2 shared task, eight categories of PHI were identified and annotated in 889 discharge summaries. The annotated PHI was also replaced by surrogate-pairs, for instance every name-mention was replaced with another name. This means that the records were de-identified before they were released (Uzuner et al., 2007).

Of the 889 discharge summaries in the data-set, 669 were released to the participants to use in development. The remaining 220 records were used to evaluate submitted systems. After evaluating Uzuner et al. (2007) concluded that the best performing systems were able to locate almost all PHI instances in the data. But since all records were drawn from the same hospital, and the regularity of names was artificially high, these systems can not directly be used to de-identify data from other sources. Hence, larger and more heterogeneous data sets are needed for the further development of de-identification systems.

The smoking challenge (Uzuner et al., 2008) is a classification challenge, where systems are built to identify the smoking status of a patient based on the patient's discharge record. The motivation was to inspire new research within MLP by facilitating evaluation and comparisons on the same data-set. By letting domain experts annotate the smoking status of a patient based on their medical intuitions, one can also evaluate the system's ability to infer new knowledge that is not spelled out in the narrative text.

The discharge records were annotated on *document level*, and a document belongs to one of the five categories: Past smoker, Current smoker, Smoker (either current or past), Non-Smoker and Unknown. This was done for both textual information, meaning that the information was spelled out in the narrative, and intuitive, meaning that the experts annotating the reports had to infer the smoking status based on other clues in the text. In the challenge the participants were only using the textually annotated material, in other words the "intuitive" annotations were not used to score the submitted systems.

The winning team got a micro-averaged F-measure[16] of 0.90, closely followed by 12 other system runs which all had a micro-averaged F-measure above 0.84. The systems used different techniques for uncovering smoking status, but most of them made use of explicitly textual features, such as whether the document contained sub-strings like "smok", "tobac" etc. (Uzuner et al., 2008).

[16]See section about evaluation metrics from page 72.

i2b2 2008 Shared Task

The second i2b2 shared task was centred around detecting obesity and co-morbidities in clinical records. This is a multi-label classification task involving 16 morbidities. The motivation for this task is the development of "indexing, classifying, and summarizing obesity-related facts found in discharge summaries." (Uzuner, 2008, p. 562) The annotated data released with this shared task was used in our experiments (see section 7.2).

The data for the Obesity Challenge consists of 1237 discharge summaries for patients which recently had problems with diabetes or obesity. These documents were annotated by hand by two medical experts with textual and intuitive judgements for each of the 16 morbidities. A textual judgement means that the diagnosis was spelled right out in the discharge summary. Intuitive judgements, on the other hand, means that the experts had to perform some kind of reasoning in order to uncover the diagnosis, for instance classifying the patient as obese or not by looking at statements about weight and height. This is similar to how the smoking-status documents were annotated. In this challenge, the participants were competing on both the textual and the intuitive annotations.

The experts would annotate a discharge summary for each of the 16 morbidities as present (Y), absent (N) questionable (Q) or unmentioned (U) for textual judgements and present (Y), absent (N) or questionable (Q) for intuitive judgements. The challenge is to develop a system making the same judgements as the human annotators. In the textual task the rule-based systems performed best, while systems based on machine-learning competed fairly in the intuitive task.

i2b2 2009 Shared Task

In the third i2b2 shared task the participants extracted information about medications, dosages, frequencies, durations, reasons for prescribing the drug and how the drug is used (modes of administration). 1243 discharge summaries, of which 696 were released during the development period (Uzuner, 2009).

An interesting aspect of this medication challenge was the community-driven annotation of the discharge summaries. When participants entered the contest, they committed to partake in the annotation process. The i2b2 team annotated 17 "gold-standard" records of the 696 training records released. The evaluation set of annotated test-documents was then collectively annotated by the contestants. The community annotated 251 records for evaluating the systems after the system results were submitted to the challenge (Uzuner et al., 2010).

Evaluating the performance Uzuner (2009) found that the contributed systems performed well on detecting drug names, dosages, modes and frequencies but that detecting duration and reason for medication proved to be more challenging.

i2b2 2010 Shared Task

The fourth shared task focused on extracting medical concepts. The participants were to build systems which identified concepts in the clinical record, medical assertions indicating existence, absence or uncertainty of medical problems and relations between medical problems, tests and treatments (Uzuner et al., 2011).

As usual the annotated data used for the shared tasks was split in a training set used for development, and testing set used for evaluation. A set of 349 documents was used for training, with 27,837 concepts and 5,264 relations. The test set consisted of 477 documents with 45,009 concepts and 9,069 relations. Additionally the medical problem concepts were annotated so that they indicating whether the problem was present, absent, hypothetical, conditional, possible or associated with anybody else than the patient.

i2b2 2011 Shared Task

The fifth i2b2 shared task focused on co-reference resolution. In this task i2b2 gathered data from several resources, and used different kinds of clinical records, such as pathology-reports, discharge records, radiology reports, etc. Three tasks in the shared task were given: Identify mentions and co-reference in the ODIE corpus was the first task. Task two focused on the co-reference resolution part, that is, the mentions were pre-annotated. Task three was similar, but on a different set of documents (Uzuner et al., 2012).

Evaluating the contributed systems developed by the 20 participant teams, Uzuner et al. (2012) concludes that the co-reference resolution systems performed well in finding co-references, but that it seems to be more difficult when it comes to cases that require domain knowledge.

Chapter 6

UIMA and cTAKES

This chapter is devoted to the UIMA framework and the technicalities of cTAKES. These systems were used in our experiments, with some additional custom-made modules explained in section 7.1.

6.1 Apache UIMA

Unstructured Information Management Architecture (UIMA) is an open source scalable and extensible platform and an OASIS[1] standard for creating software for managing and analyzing unstructured information such as free text, audio-streams and images. The focus here will be on the text processing part. One of the ideas behind UIMA is that if NLP developers were to agree on a mutual standard for managing the work flow between components handling unstructured free text, it would be easier to share and re-use resources developed by other parties. Many NLP projects follow a similar pattern: I) Read a number of documents from a collection. II) Run several NLP algorithms in a specific order, analyzing and annotating each document, and III) Serve the resulting uncovered and structured information to its final destination. The UIMA standard defines this workflow and how the different components should communicate.

These steps could include using machine learning algorithms for training models or clustering the documents, classifying the documents or parts of the documents, storing and visualizing the data or serving it to a search engine, etc. Conforming to a standardization such as UIMA could be a huge benefit for both researchers and companies dealing with NLP tasks such as information retrieval and extraction.

UIMA was first developed by IBM (Ferrucci and Lally, 2004), but is now maintained as open source software by the Apache foundation[2]. It functions as a framework for developing information management tools, where other parties can contribute modules for information extraction and management them-

[1]https://www.oasis-open.org/
[2]http://uima.apache.org/

selves. Apache UIMA conforms to the Unstructured Information Management Architecture (UIMA) Version 1.0 standard approved by OASIS, which provides guidelines and methods for accessing unstructured information.

The document and annotations (meta-data) are handled by a component called Common Analysis Structure (CAS), which is being sent through the UIMA pipeline system. The standard defines a common work flow of managing components, where a Collection Reader module is in charge of loading the documents of interest into a pipeline, Analysis Engines enrich and structure the document and CAS Consumers post-process the annotated documents for end users. The UIMA Collection Processing Manager is responsible for initializing and running all the components in an UIMA pipeline.

The Common Analysis Structure

The Common Analysis Structure (CAS) is responsible for managing both the original content of the document as well as the additional annotated content added by the analysis engines (such as tokens, POS-tags, sentence annotations, parse trees, NEs, etc). As such, the CAS gives the analysis engines an interface access to the (original and extracted) content of the analyzed document. The information extracted from a document is stored as attributive meta-information about (parts of) the document. As the CAS object is passed from one AE to another, it is successively enriched with more information about the document, so that the AEs later in the pipeline can utilize information extracted from earlier modules. For instance, when a POS-tagger AE has populated a CAS object with its findings, the CAS can be sent to the next step in the pipeline, for example a chunker (Götz and Suhre, 2004).

The Type System

Formally the Type System is a Typed Feature Structure, supporting hierarchical types and single inheritance. Some standard base-types are already defined within the UIMA system, like string, integer, Boolean and float as well as feature structure lists and arrays. Each type belongs to a name space, similar in form to Java name spaces, in order not to confuse two types which accidentally share the same name.

There is no limitation as to what type of information can be stored in a CAS object. The developer is free to engineer a suitable Type System for the CAS, which defines the different types of information he wants to extract from the document. One can for instance define the type “Person”, which has features like “name”, “age”, etc. An analysis engine designed to extract mentions of persons in free text can then store its findings in the CAS object as “Person” and fill out the uncovered features.

The Type System, then, defines the input and output of an analysis engine, and one typically defines types belonging to a specific collection of analysis engines. When the Collection Processing Manager initializes the analysis engines, it creates CAS objects containing the definition of the different types defined for the different AEs (Hahn et al., 2007).

The Collection Reader

The Collection Reader ensures uniform access to the data for all components within the UIMA pipeline. The reader knows how to iterate over the input documents, and stores the raw data in the document into the CAS Object. Within the CAS the document is referred to as the subject of analysis (sofa), and it is possible to have several sofas within one CAS object (for instance different perspectives on the same document).

Creating a Collection Reader is done in two steps. One must first define a description of the reader, to be read by the Collection Processing Manager. This defines the input parameters that should be set by the end user, for instance in what directory the documents are located, and the Type System that the Collection Reader should use[3]. The second step is to write the actual code which reads the document collection. The code consists of an initializing method for reading user emitted parameters, the iterator components **getNext** and **hasNext** used by the pipeline system to iterate through the documents when populating CAS objects, and a **close**-method for closing the resources used for reading the documents.

The Analysis Engine

An analysis engine (AE) typically analyses the documents and infers new information. This includes simple AEs such as tokenizers and sentence detectors as well as sophisticated machinery which parses the sentences, detects named entities, etc. The AE can utilize annotations inserted to the CAS by previous AEs as well as the content of the document itself, and it stores the new inferred information into the CAS objects.

When implementing an AE, one must provide a description of it, in the same manner as for the Collection Reader. This includes defining both the types that the AE retrieves from the CAS object and the new annotations with which the AE populates the object. Each AE also has a process-method, which performs the actual analysis, or calls on a third party tool (such as an OpenNLP module) for doing the actual analysis. When using third party tools for analyzing, this is typically wrapped inside an AE.

The CAS consumer

The CAS consumer iterates over all CASes and utilizes the new structured information which is populated by the AEs. The consumer can typically store the structured information into a database, use the annotated information

[3]There is a possibility that the reader wants to populate the CAS object with annotations, and not just the raw document, for instance if the documents are pre-annotated with information of interest.

as features for a machine-learning algorithm, or index the documents for a search engine. The CAS consumer is, technically, not different from an AE, the only difference is the intended use of the component and the fact that the Collection Processing Manager runs the consumers after the analysis engines. A consumer iterates over the CAS objects like an AE. When all the documents in a collection are processed, a method named **collectionProcessComplete** is called (which also exists for AEs). This method performs necessary tasks after all CAS-objects are read, such as computing statistics across documents. This is very convenient for tasks that require all analysis to be completed.

Collection Processing Manager

The Collection Processing Manager manages the flow in the UIMA pipeline. It reads the descriptions of the Collection Reader, the AEs and the CAS consumers, initializes a CAS-object for each document, and manages the pipeline flow from the document collection phase to the consumer phase. Developing a Collection Processing Manager consists of making a description of which Collection Reader, AEs and Consumers to use.

Summary

To summarize; when a collection of documents is processed through the UIMA architecture, they are first read by a Collection Reader, then a CAS object with the appropriate Type System is initialized for each document. The documents are then processed and analyzed through an AE pipeline, before entering a CAS Consumer. Then the extracted and structured meta-information is made available to a front-end application, such as a search engine or an information browser. The component taking care of all these parts is called a Collection Processing Manager.

6.2 cTAKES

cTAKES is a framework for processing clinical free text developed within the UIMA framework by the Mayo clinic and IBM (Savova et al., 2008; Schuler et al., 2008). It is released as a part of the Open Health Natural Language Processing Consortium[4]. It includes several modules which process patient records for uncovering tokens, sentences, POS-tags, chunks, named entities and more. Many of the components are trained on in-domain text or a combination of in-domain text and general corpora. We will here look into the main components of cTAKES, especially those we will apply in our experiments (see section 7.2). In our experiments we used cTAKES version 1.3.1.

When a clinical document travels through the typical cTAKES clinical document pipeline, the text gets tokenized, sentence segmented, normalized, POS-tagged, chunked and annotated for named entities, including context annota-

[4]http://ohnlp.org/

$\boxed{\text{The}}\ \boxed{\text{patient}}\ \left[_{RomanNumeral}\ \boxed{\text{did}}\right]\ \boxed{\text{not}}\ \boxed{\text{feel}}\ \boxed{\text{any}}\ \boxed{\text{chest}}\ \boxed{\text{pain}}|,|\ \boxed{\text{but}}$
$\left[_{PersonTitle}\ \boxed{\text{Dr}}\ |.|\right]\ \boxed{\text{Joe}}\ \boxed{\text{confirmed}}\ \boxed{\text{myocardial}}\ \boxed{\text{infarction}}|.|$

Figure 6.1: Base tokens and contextual tokens.

tions (negation and status). Before tokenizing, the pipeline uncovers sentences. The sentence detection module is a wrapper around the OpenNLP tools (Savova et al., 2010), in the same way as several others of the cTAKES pipeline modules. The OpenNLP sentence models used for identifying sentences are trained on annotated clinical data, and serves in this regard as a unique source when handling English clinical free text. The sentence detector annotates sentences both by looking at the OpenNLP sentence annotator output, and by explicitly declaring a new sentence at end-of-line (EOL) characters[5]. Postulating that EOL characters always mark the end of a sentence produces some difficulties using cTAKES on clinical documents. A widely used source of research within clinical text processing – the I2B2 shared tasks – involves documents where EOL characters occur inside sentences on a regular basis. When using cTAKES as an information management framework on these documents (as well as other documents with the same property) one has to deal with this. One option is to replace EOL-characters in these documents with white space, but this reduces the amount of information we have about the document. For instance may the lack of new-line characters make the section segmentation process harder. Another option is to not consider EOL-characters when identifying sentences, but this produces noise for other components such as the POS-tagger. The POS-tagger is, naturally, not trained on material where EOL occurs inside a sentence. For instance it classifies "EOL" as an NN in sentence 1. We therefore chose to replace EOL with a space character (" ").

(1) $\boxed{\text{The}}\ \boxed{\text{patient}}\ \boxed{\text{did}}\ \boxed{\text{not}}\ \boxed{\text{feel}}\ \boxed{\text{any}}\ \boxed{\text{chest}}\ \boxed{\text{pain}}\ \boxed{,}\ \boxed{\text{EOL}}\ \boxed{\text{but}}\ \boxed{\text{Dr.}}$
$\boxed{\text{Joe}}\ \boxed{\text{confirmed}}\ \boxed{\text{myocardial}}\ \boxed{\text{infarction}}\ \boxed{.}$

The tokenizer works in two steps: First, the text is split into basic tokens, by space and punctuation characters. Second, some special tokens may be detected and possibly merged by several Final State Automata (FSA) which identify entities like Roman numerals, dates, titles and lists. We see an example of a tokenized sentence in Figure 6.1, where base word (e.g. $\boxed{\text{word}}$) and punctuation tokens (e.g. |.|) are annotated as well as the person title "Dr." and the pseudo Roman numeral "did".

Some fine-tuning could have helped these FSAs, we see for instance that the Roman numeral FSA identifies **did** as a Roman numeral. It is trivial to build an FSA only recognizing Roman numerals in correct form. However, that

[5]An EOL character is the symbol for "new line", which is inserted in text when the Enter-key on the keyboard is punched.

$$FSList \begin{bmatrix} head & Lemma \begin{bmatrix} key & “patients'' \\ posTag & “NN'' \end{bmatrix} \\ tail & FSList \begin{bmatrix} head & Lemma \begin{bmatrix} key & “patient'' \\ posTag & “JJ'' \end{bmatrix} \\ tail & FSList\,[...] \end{bmatrix} \end{bmatrix}$$

Figure 6.2: Alternate spelling variants and POS-tags of “patient”.

The	patient	did	not	feel	any	chest	pain	,	but	Dr
DT	NN	VBD	RB	VB	DT	NN	NN	,	CC	NNP
.	Joe	confirmed	myocardial	infarction	.					
.	NNP	VBD	JJ	NN	.					

Figure 6.3: POS-tags applied by cTAKES.

did is identified as a Roman numeral does not mean that **did** is not treated as a regular word by the system. This could both be a strength and a weakness. For instance, the POS-tagger could benefit from knowing that a sequence of letters actually is a number, but as long the Roman numeral FSA returns false positive hits this could potentially lead to the result that AEs later in the pipeline miss out on important information.

After tokenization each word token gets normalized. The normalizer in cTAKES is a wrapper around the SPECIALIST lexical tools, and produce alternative spellings such as alphabetic case, inflection, spelling variants, punctuation and genitive markers. The normalizer AE inserts a feature structure list into the CAS-object, coupled to the token, including information about the POS-tag of every alternative inflection. An example of a feature structure list can be seen in Figure 6.2, which is the output from cTAKES for the word token $\boxed{\text{patient}}$[6].

The cTAKES POS-tagger and chunker AEs are also wrappers around OpenNLP modules, trained on clinical data. Inspecting the output of the POS-tagger, we see that it is the basic word tokens that get POS-tagged, and not the context depended tokens. For instance, “Dr.” is treated as two, and not one, tokens by the POS-tagger as we can see in Figure 6.3. On this stage, when cTAKES is finished tokenizing, normalizing and POS-tagging each sentence, each token is enriched with an annotation similar to the one in figure 6.4.

The chunker (shallow parser) actually exists of three modules, the chunker itself and two extensions which adjusts noun phrases to first include following noun phrases, and then to include following preposition phrases, then noun phrases (within the same sentence)[7]. The chunker itself is trained on annotated

[6]Due to space limitations, the last item on the list is removed in figure 6.2, which consists of the spelling “patient” with the tag “NN”

[7]These chunking modules is witnessed in the source code.

$$WordToken \begin{bmatrix} begin & 4 \\ end & 11 \\ tokenNumber & 1 \\ normalizedForm & \text{“}patient'' \\ partOfSPeech & \text{“}NN'' \\ canonicalForm & \text{“}patient'' \\ lemmaEntries & FSList\,[...] \end{bmatrix}$$

The content of lemmaEntries is figure 6.2.

Figure 6.4: The WordToken fully annotated through the cTAKES pipeline.

$\Big[_{NP}$ The patient$\Big]$ $\Big[_{VP}$ did not feel$\Big]$ $\Big[_{NP}$ any chest pain, but $\Big[_{NP}$ Dr$\Big]$. $\Big[_{NP}$ Joe$\Big]\Big]$ $\Big[_{VP}$ confirmed$\Big]$ $\Big[_{NP}$ myocardial infarction$\Big]$.

Figure 6.5: The identified chunks.

$\Big[_{LW}$ The patient$\Big]$ dit not feel $\Big[_{LW}$ any chest pain, but Dr. Joe$\Big]$ confirmed $\Big[_{LW}$ myocardial infarction$\Big]$.

Figure 6.6: Lookup Window annotations.

clinical data (Savova et al., 2010). An example output can be seen in figure 6.5.

Perhaps the most important tool in cTAKES is the Dictionary lookup module, which identifies medical named entities. This rely on the Lookup Window Annotator, which marks all NP-phrases as lookup-windows, but deletes subframes of lookup windows. This means that if window A fully overlaps window B, window B is disregarded. An example output of lookup windows can be seen in figure 6.6.

After finding the lookup windows, the Dictionary Lookup Annotator uses the normalized variants of the tokens inside this window as a query in a dictionary. The dictionary consists of SNOMED-CT and RxNorms terms, through UMLS (Savova et al., 2010). The dictionary lookup does not distinguish between SNOMED CT concepts with different statuses. For instance, when looking up the window “myocardial infarction” six different SNOMED CT concepts are returned, only one of them having the status “current”. The other five have the status “ambiguous”. Information about concept status is not saved in the CAS, so if one wants to only use current concepts, one has to implement a filtering AE (or modify the original AE) which queries the UMLS database with the dictionaries a second time uncovering the concept status and deleting non-current concepts. An example output of the Dictionary Annotator can be seen in figure 6.7, for the substring “chest pain”.

cTAKES also includes a Context Annotator, based on an algorithm similar

$$IdentifiedAnnotation \begin{bmatrix} begin & 29 \\ end & 39 \\ status & 0 \\ certainty & -1 \\ ontologyConceptArr & < UmlsConcept\,[...]\,, \\ & [...]\,, ... > \end{bmatrix}$$

Figure 6.7: An annotated SNOMED CT entity. *ontologyConceptArr* is an array containing the UMLS IDs for this entity.

to NegEx. NegEx is — as we have looked at briefly in section 3.2 — a simple, but effective, method of discovering negations based on regular expressions built specifically for clinical data. Two annotators implement the Context Annotator; Negation Annotator and Status Annotator. For each annotated named entity, they search for negation and status clues on each side of the entity. There is one regular expression which detects negation cues that negate entities to the left of the cue, and one which detects negation cues that negate entities to the right of the cue. In cTAKES, if there is a match of negation cue and the focused entity, the entity is marked with a certainty value of "-1". We see an example of this in figure 6.7. The Status Annotator works in a similar manner, and searches for instance of clues for whether the entity belongs in the past ("the patient *had* chest pain"), then annotates the entity accordingly.

Chapter 7

Experiments

In this chapter we will perform and discuss our experiments. What we want to explore is how well one of the up-to-date systems performs when extracting information. This is not easily evaluated directly, since there are few, if any, available gold standards that can be used to evaluate the system performance. Instead we have used the indirect approach by using the output of cTAKES as input to a machine-learning algorithm to classify the documents in the i2b2 2008 shared task challenge. In the experiments we make use of machine-learned decision trees to annotate the morbidities found in the shared task data. The uncovered information extracted from cTAKES is used as clues, or features, that the decision tree uses to decide whether a patient has one or more of the 16 morbidities which the shared task data is annotated with. The use of decision trees might not give us optimal performance, but it gives us a unique insight into how the computer decides on a given morbidity.

Before we could begin our experiments, some adjustments had to be made. We built extensions to the system so it could read the i2b2 documents and use the extracted information to train and use decision trees for classifying documents. So we extended the cTAKES framework with modules that can read the i2b2 data and modules that use the output from cTAKES to train and classify the documents. These extensions are described first, before we cover our experiments. The chapter ends with a discussion of our results.

7.1 Extensions of UIMA/cTAKES

Some UIMA modules had to be built for this thesis. Specifically new Collection Readers, a simple Section Segmentation AE, AEs to extract features used in several machine learning algorithms, and some AE wrappers for the Weka Data Mining Java tools (Hall et al., 2009). We will briefly discuss all the components that we needed to develop before experimenting could begin.

A simplified overview of the extended pipeline is shown in figure 7.1. Typically one has to make extensions in the front and the end of a pipeline to adjust it to specific needs: First we needed to adapt the i2b2 data to the

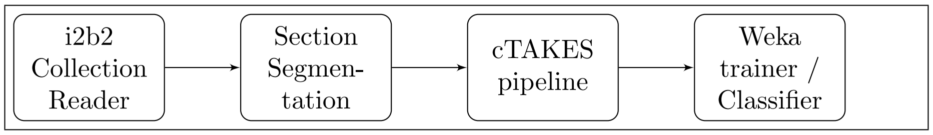

Figure 7.1: The complete pipeline of our system.

original pipeline, which included reading the XML formatted i2b2 documents and uncovering the sections in each document. At the end of the pipeline, we wanted to adapt the extracted information into a setting where we either trained weka models for classification, or used previously trained models to classify documents.

The i2b2 2008 Collection Reader

A UIMA Collection Reader for reading the i2b2 2008 shared task data was developed. These documents are stored in XML-format with annotations stored in a separate XML-document. Three parameters, intended to be emitted by the end-user when building a pipeline handling i2b2 2008 documents, were set. The user has to tell the Collection Reader where to find the XML-document containing the clinical documents (obligatory), where to find the XML-document with annotations (optional) and whether the Collection Reader should replace new-line characters with a white space character. The last parameter was needed because of how cTAKES handles new-lines when annotating sentences or how the POS-tagger behaves when it sees a new-line token (as discussed in the previous chapter).

This Collection Reader initializes by parsing a XML DOM-tree over both the documents-file (DOC) and the annotations-file (ANN). For this, the standard *javax.xml* library was used. Then, when the Collection Reader is iterated by the Collection Processing Manager, it climbs the DOC and returns one document at the time. If the user has set the "replace new-line characters with white space" parameter, the Collection Reader does this before populating the CAS object with the clinical text. (Otherwise the text is not altered in any respect.) If the user has specified an annotation file, the Collection Reader also extracts the annotations from the ANN (for the document at hand) and inserts it into the CAS object.

A UIMA Type System was also introduced, where a MorbidityAnnotation extends the uima.tcas.Annotation type. It includes three features, first the Morbidity, with a string value indicating one of the sixteen morbidities, second a Certainty feature, with a string value being either "Y", "N", "Q" or "U", lastly a feature named AnnotationType, with a string value of either "intuitive" or "textual". In this case, it is the Collection Reader, and not an Analysis Engine that populates the CAS with annotations, because the documents are pre-annotated. The classification module (explained later) uses the same Type System when populating the CAS with predicted annotations based on the decision tree models.

New Analysis Engines

We developed a Section Segmentation AE, a Weka Classifier AE and several AEs used to extract CAS annotations for using them as features when training models with Weka. The Section Segmentation AE is a simple and naïve section annotator built specifically for the i2b2 2008 documents. It employs a regular expression in order to identify headlines of each section. It marks the start of a section at the start of a headline, and the end of a section at the start of the next headline. Exceptions are the start of the first section, which starts at the beginning of the document, and the last section which ends at the end of the document.

The sections are annotated by section type. To find the type of a section, a rule-based system is used which looks at the content of the headline. For instance, if the headline consists of the uppercase letters "DIAGN" the section is annotated as the type "diagnose". As said, this is a simple section segmentation, and investing in a more sophisticated one could be of value in a clinical document pipeline. The output of this section segmentation method is explained in more detail from page 76.

The weka classifier AE is a wrapper around the Weka tools. This supports multi-classification tasks by loading several weka models at once, and filters away features unknown to the model by coupling each model to a Weka arff-file. There are defined two user parameters: One to specify the directory in which the model files are located, and one to specify the directory where the weka arff-files are located. The AE simply couples each arff-file with a model-file by matching the file-names.

Each Weka model file is a byte-serialized Java object implementing the Classifier interface defined in Weka. Each model file is deserialized as a Classifier, and used for classifying a set of feature-values. When classifying a document, the relevant features are extracted from the CAS object. Features or feature values that are not defined in the arff header file are filtered out, and a new in-memory arff-file representing the single document is created. This document is then served to the Classifier object, and the predicted class is extracted. The class is then inserted into the CAS object again, linked with the name of the classifier.

Some Weka Types were also defined in the UIMA Type System, signifying the different Weka attribute types. The root Weka attribute type is called Attribute, with the feature "name". Three subtypes were defined, NominalAttribute, StringAttribute and NumericalAttribute. All three have a feature "value" which is either a string (for nominal and string attributes) or a Double (for the numerical attribute). NominalAttribute also has an extra Boolean feature named classificationClass. If this is set to true, it is used as the *class* when training Weka models.

The Weka Consumer

A CAS Consumer which creates Weka-files was also needed. It employs the same weka types as the Weka Classifier AE, and creates files in arff format out of the annotations in the CAS Object that uses the Weka features. Two parameters are defined for this consumer, an optional arff input file, and an output directory for storing the new arff files. The arff input file is used in the same manner as the Weka Classification AE; it filters out attributes and attribute values not mentioned in the arff-file, and ensures that the order of the attributes and attribute-values is the same. This is necessary when creating both a training set and a test set of arff-files, because the trained Weka models expect their input to be in the same order when classifying new documents.

This Consumer can output several arff-files, depending on how many different nominal attributes are used as a classification class. There is only one nominal attribute marked as a classification class which is saved in each arff-file. This way, one can effectively create several classifiers at once in Weka for the same set of documents. The actual training and creation of the weka model files was not done in UIMA, but a simple consumer running after this consumer could easily be created if one was to make a pipeline that finishes the whole job.

7.2 Experiments with the i2b2 2008 Shared Task

As said we want to, in one way or another, evaluate how well a modern IE-system for clinical data performs. Our choice of IE-system fell on cTAKES, because as an open-source project it is freely available for scientific purposes and because the system utilizes ontologies such as SNOMED CT and RxNorm through the UMLS Metathesaurus. As evaluation data we used the documents and annotations released in connection with the i2b2 2008 shared task, as described from page 57.

The nearest thing to compare our results with are the contestants in the i2b2 2008 contest, but the reader should be aware that they developed systems designated to compete in this task. We, on the other hand, have not altered the cTAKES pipeline in any way to gear it towards better scores in the given task. To encompass this we have instead coupled the metadata extracted with cTAKES and the annotations from the shared task to train a classifier for each morbidity we are interested in. These classifiers should then, based on the output from cTAKES, be able to decide whether a patient belonging to a given discharge summary has a given morbidity or not.

We used the uncovered Named Entities from the cTAKES system as features for classifying the documents. We also utilized the simple section-segmentation module developed for these experiments as well as the negation detection algorithm within cTAKES. These features were used to train several sets of classifiers in accordance with the shared task at hand.

The Corpus Data

The i2b2 2008 data consists of 1237 discharge summaries, each annotated for 16 different morbidities. The annotation is done both for textual judgements (whether or not it is spelled right out in the document that the patient suffers from the morbidity) and intuitive judgements (if the annotator has to do some thinking before annotating the summary). This totals in 32 different annotation-types, 16 morbidities and 2 different types of judgements.

For each morbidity and for each annotation-type the document is annotated with a value **Y**, **N**, **Q** and **U**. **Y**es, the patient has the morbidity. **N**o, the patient does not have the morbidity. It is **Q**uestionable whether the patient has the morbidity or not, and for the textual task, it is **U**nmentioned whether the patient has the morbidity or not. We will consider these values (Y, N, Q and U) as classes, and classify each document for each of the 16 morbidities, for both textual and intuitive judgements. This sums up to 32 different classifiers, with three or four classes for each classifier.

The 1237 documents are divided into three sets, 611 documents as a *training* set, 119 documents as a *held-out* set. The rest of the documents are used as a final *test set*. The held-out set is used as a temporary test set while developing the classifiers. The test-set was not available for the participants in the shared task when developing their systems. It is not kosher to evaluate a system on the same documents that are used to train it, because the system can be "overfitted" to these documents. This means that the evaluation figures do not reflect how the system would perform in a realistic setting, where it is used for classifying unseen documents. Therefore, we used the same strategy here, and did not look at the test set before evaluating different settings on the held-out set. At the end we decided on a final configuration and evaluated this setting on the test set. We found that the best setting was to count the occurrences of each concept in the discharge summary, and use this as features, while differentiating between negated and non-negated concepts and ignoring concepts occurring in the family history section of the document.

Methods

cTAKES was used, with its default configurations, for extracting metadata from the documents. The Weka data-mining tool was used for creating the classifiers, with the help of the components described in section 7.1 (page 70). Named entities (from the UMLS Metathesaurus) were extracted as features, and different settings were tested: In some settings we only looked at whether the entities were mentioned in the current document or not, in others we also included information about whether the entity was negated or not. The simple section segmentation component (technical descriptions in section 7.1 on page 69) was also used, to indicate in which part of the document the clinical named entities were located or to filter out uninteresting parts of the document.

A similar approach can be found in Meystre (2009), which used MMTx to extract entities from the text, in combination with ConText to analyze negation, temporality and experiencer[1] and a method for dealing with ambiguous terms. The paper also used the semantics in the UMLS Metathesaurus actively by (manually) linking the 16 morbidities to related concepts. For instance was hypertriglyceridemia linked to primary hypertriglyceridemia, secondary hypertriglyceridemia, blood triglycerides increased, and so on.

To cover the intuitive annotations better, Meystre (2009) also extracted a set of keywords for each morbidity which gave an indication of whether the morbidity was present. This set includes medicines and therapeutic procedures only used to treat the disease in question (like the drug allopurinol to treat gout and the procedure fundoplication to treat gastro-esophageal flux). Their system also recognized specific biomarker values used when detecting specific diseases.

In this experiment we have, instead of using the semantics of the Metathesaurus and specific keywords, depended on machine learning algorithms to uncover which entities are linked to the different diseases. In the 611 training documents a classifier is trained to uncover the diagnosis given the set of features. The classifier does not know in advance which clinical entity that corresponds to the disease it is trying to classify, but learns this only indirectly. In the process, the classifier should also become sensitive to other clues indicating the disease, such as that the presence of |*allopurinol*| indicates that the patient has gout.

This was done in the following manner: In the training phase we extracted so-called *features* from the document, which is a set of properties of the item we want to classify[2]. The model trained and used for classification only see these features, it does not see the document as a whole, nor all extracted metadata. It is given, then, that the system performance depends on what kind of features we choose to extract.

The machine learning algorithms explored are Naïve Bayes, and a decision tree[3], both as implemented in Weka. With the decision tree we get direct information about why the classifier assigns a class to a given document, which may reveal interesting data. We also found that we got better results using decision trees at an early stage in our experiments.

Evaluation Metrics

For evaluating our system performance we use *Precision*, *Recall* and the harmonic mean (F_1) of these. Precision is the fraction of the classified instances that are correct, while recall is the fraction of the instances that are correctly classified. For instance, if we are measuring the precision when classifying the class Y, we count two things: How many instances that the classifier says belongs to Y and which actually belongs to Y (true positives, TP), and how

[1]The experiencer as in, does the entity describe something about the patient or someone else?

[2]The "item" we want to classify is really the patient, which is represented by a discharge summary.

[3]The J48 implementation of the C4.5 algorithm.

		Annotation	
		TRUE	FALSE
System	TRUE	TP	FP
	FALSE	FN	TN

Table 7.1: Counting true and false positives and true and false negatives.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Figure 7.2: Formulas for precision, recall and F-score.

many instances the classifier says belongs to Y but does not (false positives, FP). (See table 7.1.) We find the precision by the equation given in figure 7.2. Notice that saying that no instances belong to Y gives a maximal precision of 1.

In a similar manner we find the system recall (also called sensitivity) by counting true positives and false negatives. For the class Y, false negatives (FN) are instances of Y that are not detected by the classifier. We find recall by the equation in figure 7.2. Notice that we gain a maximum recall simply by saying that all instances are members of the class, much in the same way we get a maximum precision by classifying no instances as Y. Because of this we use the harmonic mean between precision and recall, called F_1.

The evaluation software following the i2b2 2008 shared task is used when evaluating the different sets of classifiers. Given a system output and a gold standard (the correct answers) the software computes precision, recall and the harmonic mean (F_1) between these, both micro- and macro-averaged. The metrics are described in detail in Uzuner (2008). The participants in the shared task competed for highest macro-averaged F-measure because i2b2 wanted to focus on the less well-known classes: Macro-averaging the scores gives an equal weight for each class, regardless of the class frequency. As we can see in Table 7.2 some classes are very rare, for instance missing the one member of class Q for the intuitive judgements of the PVD classifier will give a precision and recall of zero for this class.

It is difficult, if not impossible, to learn such low frequent classes with machine learning algorithms because there is no way to separate features which are unique to the class by chance with features which naturally only occur inside the class. As such, it has no purpose here to compete with the highest ranking scores of the i2b2 2008 shared task participants. The macro-averaged scores are still included for convenience of comparison.

When macro-averaging the scores, we are averaging over each class and not over the result for each morbidity. Therefore, we see in the evaluation tables in appendix A that the macro-averaged overall system score can be lower than if we average the score for each of the morbidity classifiers. For instance, when calculating the macro-averaged precision for textual judgements, the i2b2 evaluation script computes the precision of class Y over all morbidities. This is done by summing up TP and FP for all morbidities, and then computing the precision. The other classes N, U and Q are computed similarly. The macro-averaged F-score over all morbidities for textual judgements is then the average over these four classes and we are penalized even further for misclassifying the low-frequent classes.

When micro-averaging the scores, precision, recall and F-measure will be identical to each other. Instead of talking about three different measurements which in effect gives us the same number, we will from now on use the term "accuracy" instead of micro-averaged metrics. Accuracy, then, is simply the fraction of how many documents the system has classified correctly (a score of 1 would mean that all documents are classified correctly). Macro-averaged precision, recall and F-measures will be denoted as P-Macro, R-Macro and F-Macro respectively.

Baseline

The baseline gives us an idea of what we could, at least, expect from a system, by using a simple heuristic for classifying each instance. For instance, a baseline classifier could pick out a random class, or it could classify all instances as the most frequent class from the training material. A natural baseline for this classification task is to classify all instances as the most frequent class in the classifier task. Table 7.2 gives us the frequency distribution for each class, for each morbidity on the held-out data. The baseline intuitive judgement Asthma classifier then, would say that all documents are an instance of the class N, whilst the intuitive CAD classifier says that all documents belong to class Y, etc.

The result of this simple classification procedure is reported in Table 7.3, the resulting macro-averaging f-measures for intuitive and textual classification being 0.4896 and 0.3550 respectively. It proves that even a simple baseline such as this can be hard to beat; the worst macro-averaged F-measures of the submitted systems in the i2b2 2008 shared task were 0.3358 for intuitive and 0.2237 for textual judgements[4] (Uzuner, 2008).

[4]Note that the shared task numbers are for the test-data, while our baseline-numbers are for the held-out data-set

	Intuitive			Textual			
Morbidity	Q	N	Y	U	Q	N	Y
Asthma	0	502	70	529	1	1	75
CAD	4	223	325	241	4	16	333
CHF	1	280	243	345	0	7	239
Depression	0	460	122	519	0	0	86
Diabetes	5	171	396	184	6	12	399
GERD	1	372	115	501	3	1	98
Gallstones	0	506	87	513	0	3	91
Gout	2	518	78	534	2	0	71
Hypercholesterolemia	1	240	262	342	1	9	246
Hypertension	0	103	428	149	0	10	442
Hypertriglyceridemia	0	554	33	594	0	0	15
OA	1	467	98	514	0	0	89
OSA	7	506	84	514	7	0	88
Obesity	1	314	239	354	4	3	245
PVD	1	469	87	526	0	0	83
Venous Insufficiency	0	482	44	592	0	0	14

Table 7.2: Class frequencies for textual and intuitive judgements over each morbidity from the training set.

System	Accuracy	P-Macro	R-Macro	F-Macro
Intuitive	0.7724	0.8250	0.4880	0.4896
Textual	0.7686	0.8578	0.3537	0.3550

Table 7.3: Baseline precision, recall and harmonic mean, micro- and macro-averaged on held-out data.

Features and models

In order to classify each document we extracted pre-defined annotated information, assembled it as features of each document and used a training algorithm for teaching the computer which features are significant for a given class. We used three different types of information pieces to create the features: Named Entities, document sections and negation of the Named Entities. These are put together in different ways to form features describing each document. We here operate with features and feature-values. For a set of classifiers, we have a fixed number of features. For each feature, there is a *value*, and it is the value that we use for separating the different documents. We can, for instance, have three features named "Asthma", "Diabetes" and "Gout" indicating whether one of these diagnoses are mentioned in the document. The feature-values could then for instance be *present* if the given feature exists in the document, *null* if it does not and *negated* if it is present but negated.

Since Named Entities can occur several times within a document, we also used the *frequency count* (how many times the entity was mentioned in the document) as feature values. To separate negated and non-negated Named Entities we created two features for each entity name, for instance "Asthma:Present" and "Asthma:Negated" for the named entity $|Asthma|$.

We have used two different kinds of supervised machine learning algorithms, Naïve Bayes and C4.5 decision tree algorithm. Both are taken from the Weka package (Hall et al., 2009). Naïve Bayes is a stochastic method, multiplying up probabilities for the different classes given each feature present in the document. The decision tree tries to discover unique combinations of features for each class. An example of a decision tree is given in figure 7.3 (page 84). One reason for sticking with decision trees is the issue of transparency: We always know how the computer decides which class the document supposedly belongs to.

NE Only

The NE classifier simply uses the uncovered named entities from the Clinical Document Pipeline in cTAKES as features. In this first iteration of building a set of classifiers, no effort has been made to uncover different sections of the document. Hence the system does not differentiate between mentions of diagnoses from a diagnose section and mentions of diagnoses from a family history section. Entities which are negated are also disregarded in this first run, which is to say that if the Named Entity $|Asthma|$ only occurs negated in a document the feature "Asthma" of that document gets the feature value *null*. Each document is thus represented by a set of features with the value *present* or *null* according to whether the NE exists in the document or not. In the first iteration these features are fed to the machine learning algorithm in order to build models which can recognize the classes given the features. In the second iteration we evaluate models by comparing its judgements to the correct answers in the held-out set of documents. As said, one classifier model is made for each of the sixteen morbidities for both textual and intuitive annotations, resulting in a total of 32 classifiers.

When only using Named Entities we already achieve a high accuracy. For the Naïve Bayes classifiers, the total accuracy over all 16 classifiers for textual judgements is 81.97 % while the accuracy for intuitive judgements is 81.74 %. Classifiers built over the decision tree perform even better, with an over-all accuracy of 90.98 % and 90.64 % for textual and intuitive judgements respectively.

We see that Naïve Bayes outperforms the Decision Three when it comes to P-Macro for textual judgements, but gets a lower F-Macro due to lower R-Macro. For a full report on how these two sets of classifiers performed on each morbidity, see tables A.1 and A.2 on page 102 and 103 in the appendix.

Section Segmentation

Not all parts of the document are necessarily valuable when trying to classify diagnoses of the patient, in particular this holds true for the family history section. Therefore, a very simple section segmentation was developed, as pre-

Intuitive judgements

Algorithm	Accuracy	P-Macro	R-Macro	F-Macro
Naïve Bayes	0.8175	0.8646	0.5129	0.5200
Decision Tree	0.9064	0.9286	0.5942	0.5947

Table 7.4: Accuracy and macro-averaged P, R and F scores for NE-only features on intuitive judgements.

Textual judgements

Algorithm	Accuracy	P-Macro	R-Macro	F-Macro
Naïve Bayes	0.8197	0.8941	0.3834	0.3873
Decision Tree	0.9099	0.4497	0.4503	0.4500

Table 7.5: Accuracy and macro-averaged P, R and F scores for NE-only features on textual judgements.

viously described, to detect many of the sections in a document. This consisted of defining a greedy pattern which recognized most of the headlines in the document. The pattern for detecting headlines was *"([A-Z -][A-Z -]*):"*, where the parenthesis marks the headline start and end. This expression captures all sub-strings consisting of capitalized letters, space (" ") and hyphen ("-") that end with a colon (":").

This is perhaps a naïve method of finding and annotating sections, but the classifiers should now nevertheless have some more information about the document. Developing more sophisticated methods for finding sections should yield even better results. Based on literature around the i2b2 2008 shared task (see for instance Childs et al. (2009)) we defined these following section types: Diagnosis, family history, laboratory data, comments, medications, allergies, past and other. The content of the headline decides what kind of section we are annotating. For instance, if the headline contains the letters "DIAGN" the following section is marked as "diagnosis". The reason for only using "DIAGN" to uncover the diagnose-part of a document is that no other tokens in non-related headlines contains the sub string "DIAGN", the fact that some headlines spell it out "DIAGNOSES " while other spell out "DIAGNOSE" or "DIAGNOSIS". This also works as a recovery for some spelling errors. Similar sub-strings for uncovering the section-type were identified for all types, based on the most frequent occurrences of headlines. If any identified headline is not covered by one of these rules, it is regarded as the type "other".

In table 7.6 we have included some statistics over how many sections, and what kind of sections, that was uncovered in the training set with this method. We see that the "comments" sections and the "Other" sections are the most numerous, with an average appearance of over 9 times in a discharge summary. This is because in the "Comments" sections we tried to detect all sections that seemed interesting, while the "Other" sections is not identified as any of the other types. We have also included a list over the sub-strings used to identify

Section type	Freq.	Rate	Avg. pr doc.
Allergies	609	0.0396	0.99
Comments	5873	0.3819	9.61
Diagnosis	782	0.0509	1.28
Family history	437	0.0308	0.71
Lab data	318	0.0207	0.52
Medications	1003	0.0652	1.64
Other	5888	0.3829	9.64
Past	468	0.0304	0.76

Table 7.6: Frequency, rate and average occurrence of the sections types in the 611 training documents.

a section in Appendix C.

The first classifier which took sections into account simply labelled all NEs with the section the NE was found in. So the attribute was named for instance diagnose9999999 instead of just 9999999, if the concept ID 9999999 was found in a diagnoses section. When testing a Naïve Bayes and decision tree classifier with these features the accuracy went down compared to only using NE as feature. For Naïve Bayes the total accuracy went down from 81.97 % and 81.74 % to 80.60 % and 80.60 % for textual and intuitive judgements respectively. For the decision tree the corresponding numbers were from 90.98 % and 90.64 % down to 86.93 % and 83.51 %. Also the macro-averaged P, R and F_1 scores dropped with respect to the decision tree. F-Macro dropped from 0.5947 to 0.5380 for intuitive judgements and from 0.4500 to 0.4432 for textual judgements, as can be seen in table 7.7 page 79[5].

It seems that knowing which section the different named entities are extracted from does not help us much. This could be due to a number of factors, for instance that the section segmentation method we have developed is not sophisticated enough or that the features get to sparse to be used efficiently with the machine learning algorithms. Another solution is to simply disregard sections which seem uninformative, instead of annotating each NE with the document section the NE was extracted from. This will solve the sparseness problem. So instead of marking each NE feature with section, new feature values were extracted by simply disregarding some sections when collecting NEs. We decided to filter out the sections labelled as “family”, “allergies” and “past” in our first go. The result of this can be seen in table 7.7 on the line “NE+MINUS-ALL” (meaning that we use NEs as features, but not including the three mentioned types of sections). This procedure performs better than when separating features according to which section we find them in, but disregarding sections at all still performs the best. The only exception is F-Micro for textual judgements, going even further down from 0.8693 to 0.8635.

[5]All tables with micro and macro-averaged P, R and F-scores for all sets of classifiers are included in appendix A.

	Intuitive			
	Accuracy	P-macro	R-macro	F-macro
Baseline	0.7724	0.8250	0.4880	0.4896
NE	0.9064	0.9286	0.5942	0.5947
NE+SEGMENT	0.8351	0.5451	0.5324	0.5380
NE+MINUS-ALL	0.8768	0.5749	0.5679	0.5711
NE+MINUS-FAMILY	0.9064	0.9288	0.5939	0.5946
NE+NEG+MINUS-FAMILY	0.9122	**0.9360**	0.5948	0.5984
FREQ	**0.9384**	0.6265	**0.6137**	**0.6196**
	Textual			
	Accuracy	P-macro	R-macro	F-macro
Baseline	0.7686	**0.8578**	0.3537	0.3550
NE	0.9099	0.4497	0.4503	0.4500
NE+SEGMENT	0.8693	0.4495	0.4401	0.4432
NE+MINUS-ALL	0.8635	0.4952	0.4424	0.4584
NE+MINUS-FAMILY	0.9088	0.6989	0.4508	0.4499
NE+NEG+MINUS-FAMILY	0.9083	0.7802	0.4895	0.5029
FREQ	**0.9425**	0.6488	**0.5808**	**0.6063**

Table 7.7: Micro and macro-averaged F_1-scores. Decision tree classification of held-out material.

It seems intuitive that the sections of the records mentioning family history should not contain information about the patients morbidity, at least not in such a way that this can be discovered with the tools we are employing. One last attempt of filtering out sections was therefore investigated. We only disregarded the family-section of the documents, and collected NE features as usual. The result of this was slightly smaller scores than the original NE-setting, but these differences are probably not significant (though we did not really assess whether the improvement were significant or not): F-Micro remained the same for intuitive judgements compared with NE only, while for textual judgements it went down from 0.9099 to 0.9088. F-Macro went down from 0.5947 to 0.5946 (intuitive task) and from 0.4500 to 0.4499 (textual task).

Since there was almost no difference between using NEs from the whole document and using NEs from the whole document minus family history-sections and since we have the intuitive notion that the family history sections do not include much information about the patients morbidities, we will continue to disregard the family section.

Negated Named Entities

As mentioned earlier, and as showed in Table 7.2, some classes are very rare. For the intuitive judgements these are questionable diagnoses for CAD, CHF, Diabetes, GERD, Gout, OA, OSA, Obesity and PVD. For the textual judgments these are the questionable and/or negative diagnoses of all the morbidities except Hypertriglyceridemia, OA, PVD and Venous Insufficiency. In the i2b2 2008 shared task they wanted to focus on the correct classification of these rare cases, and therefore used an unweighted macro-average for the F_1-measure. This means that missing one instance in one of these rare classes will have a large impact on the end result.

More sophisticated analysis of the text is most likely necessary in order to get the questionable diagnoses correctly classified. The few negative judgements in the textual classifications, on the other hand, could be easier to detect if we recorded information about negated named entities. Therefore, instead of disregarding the negative entities, it could boost system performance to use those as features as well.

There are two ways of doing this. Either we could have negated concepts as new features, with two values "present" and "not present". Or we could use the original features, with the extra possible value *negated*. These two scenarios seem to give the training algorithm the same information, but the last one will not capture the cases when a document contains both negated and non-negated mentions of the same entity. Since we are already using nominative feature-values the natural extension is to use the value *negated* instead of *null* if the document contains a negated mention of an uncovered entity. Later, when we use numeric values instead, we will go for the first alternative, which basically consists of adding more features rather than more feature values.

The "feature space" for each document now consists of a fixed set of features representing given SNOMED CT and RxNorm concepts. Each of these have either the value "present" if the concept is present in the document and not negated, "null" if the concept is not found in the document or "negated" if the concept is found in the document but is negated. This setting gave us better results than any previous scheme. Using the Weka J48 decision tree, we outperform the simple NE classifiers, with exception of micro-averaged scores for textual judgements. The biggest leap upwards is nevertheless the macro-averaged scores on textual judgements — F-Macro goes from 0.4500 to 0.5029 — as we hoped.

Counting frequency

We are still trying to increase performance. At this point we found an obvious but overlooked flaw in our scheme. We are representing each document with a nominal value indicating whether a concept has occurred in the document or not. When neglecting negated entities this could possibly be defended, but what now, when we are representing negated mentions of the entities as well? The obvious problem is that a document can contain both negated and non-negated occurrences of a given entity, and we have no way to capture this with the feature engineering performed until now.

We solved this by adding features for negated entities. Instead of using features like "99999999" representing the unique identifier for a SNOMED CT or RxNorms concept, we use the features "99999999:present" and "99999999:negated". We are also moving away from using nominal values, and instead count the number of times any concept is mentioned in the document (according to cTAKES). This way we are also giving the training algorithm more information about the document.

Comparing the evaluation metrics when using frequency as feature values (FREQ) in table 7.7 we see a significant performance boost. The biggest leap is seen in F-Macro for textual judgements, which goes up more than 0.1 points. Surprisingly we also see that P-Macro goes *down* quite a bit both for textual and intuitive judgements. If we compare the two tables A.6 and A.7 (page 107 and 108) we see that the precision drops significantly for OSA, both in textual and intuitive judgements. Since R-Macro also is below average this affects the F measure as well. But we see a profitable increase of macro-averaged precision, recall and F measure in Hypertension. So it seems that using the frequency count of each entity gives the computer more insight with respect to particular morbidities, but not all. When examining the evaluation metrics for each morbidity in the two aforementioned tables, we see increased performance for some morbidities, and decreased performance for others.

A peculiarity seems to be that, for intuitive judgements, the performance of PVD is identical when using frequency counts feature values and nominal feature values. This was a surprising find, and had to be investigated further. That these values are identical seems to indicate that the decision trees contain the same information or that some kind of error has occurred in the transition from using nominal values to using frequency values. We examined the model outputs and witnessed that the decision trees used different information. We then ran the experiments several times to rule out any errors. That we get identical results seems to us a mere coincidence.

Results

We have now examined how using discovered SNOMED CT and RxNorm concepts, in combination with section identification and negation, has performed when classifying morbidities of patients given their discharge summaries. Despite the fact that using frequency values made the precision of the system decrease notably, this gave us the best macro-averaged F-score. We therefore allowed ourselves two final system runs, one with the nominal values and one with frequency values.

The final evaluation is performed on the same test-set as the i2b2 2008 participants used. In this way, we get comparable evaluation values. This time we include both the training set and the held-out set in the training phase, which gives the decision tree machine learning algorithm more data during training. Hopefully this extension of the training material will boost performance even more.

When comparing the performance with nominal values on the final test set (table A.9, page 110) to the similar system on the held-out test set (table A.6, page 107) we see that the P-Macro drops, but that the F-Macro values increase, in much the same way as when going from nominal values to frequency values on the same training and evaluation set. The difference between these settings

	Intuitive		Textual	
	F-micro	F-macro	F-micro	F-macro
Nominal values	0.9208	0.6053	0.9158	0.5433
Frequency values	0.9324	0.6142	0.9490	0.6052
Solt et al.	0.9590	0.6745	0.9756	0.8000
Yang et al.	0.9572	0.6336	0.9723	0.8052

Table 7.8: Comparison of our system with the top two contestants in the i2b2 2008 shared task.

is the amount of training and evaluation documents, the latter one having both more material for training and for evaluation. Having more training material then, seems to give us better recall and overall system performance, at the expense of precision.

If we compare the two final system runs, the one using nominal values (table A.9, page 110) and the other using frequency values (A.10, page 111) we see that overall performance increases when using frequency values. This can particularly be seen in the overall F-macro score on textual judgements which goes up from 0.5433 to 0.6052.

A comparison of our system and the top-performing contributors in the shared task is given in table 7.8. The achievements of the systems reported are taken from Uzuner (2008). From this source we also learn that our system would not have made it to the top-ten list in this shared task. In the textual judgements 10th place went to DeShazo et al. with an F-macro of 0.6140, while our best score was 0.6052. In the intuitive judgements 10th place went to Jazayeri et al. with an F-macro score of 0.6287 while our score was 0.6142. Considering that we did not use a system tuned for this task (except the fact that our models, of course, were trained and evaluated on the same material), but used a general clinical IE system coupled with a rather naïve section segmentation method, these results are perhaps not so bad. We did not have domain experts to help us engineer rules, symptom-lists etc. for the task, like many of the shared task participants had. Finally, the decision to use decision trees is also very likely to have altered our performance in a negative way. Using a more sophisticated training/classifying algorithm would most likely alter the final results, but unfortunately we did not have the time to investigate this further.

Decision Trees

It is interesting and enlightening to see some examples of the decision trees used to classify the documents. The advantage of using decision trees is that we can directly see *why* the computer has classified the document/patient the way it has. The choices the computer makes are transparent. This is not the case with more sophisticated machine learning algorithms.

The decision tree created by Weka for the textual classifications of gout is perhaps the easiest example to run through due to its small number of nodes (decisions). Figure 7.3 is a graphical representation of the decision tree we get when using identified NEs as features, with the nominal feature values *present*, *negated* and *null* (meaning that the NE was found, the NE was found but was negated, or the NE was not found in a document) when ignoring "Family history" sections of the documents. The circles (nodes) represent the feature-names, and in this tree we have only three of those: |*Gout*|, |*Gout NOS*| and |*Entire right lower leg*|, all concepts from SNOMED-CT. This means that the training algorithm only found these three features to be of any value for separating patients with gout, patients unknown to have gout and cases where it is questionable whether the patient have gout or not.

The arrows (arcs) represent feature-values and what choice the classifier makes when encountering a document with this feature value. For instance, if the classifier sees that the concept "Gout" has been negated for this document, the patient is classified as having gout: The arc labelled *negated* points to the quadratic box (leaf node, representing the selected class) containing an "Y" (Yes, the patient has gout). That the classifier says "Yes, the patient has gout" may seem strange when the concept "gout" is negated. However, the number under the "Y" indicates that only one document in the entire test set had a negated instance of |*Gout*|[6], and the patient did actually have Gout. The reason for this could for instance be that the negation detection algorithm erred on this concept.

If |*Gout*| is present in the document, the classifier enters the |*Entire right lower leg*| node to separate the classes "Q" and "Y". If this concept is present, the classifier says it is questionable that the patient has gout. Only two questionable instances are detected with this rule, and it is likely that this decision does not reflect facts from reality: It happens to be that out of the 68 annotated documents where |*Gout*| is present, two of those classified as "Q" also have a present (and non-negated) |*Entire right lower leg*| in common, so the algorithm selected this feature to separate these instances.

We also see that if |*Entire right lower leg*| is negated, it chooses the class "Y" despite the fact that no such documents exist: This decision is based upon the class frequency of all the instances in the |*Entire right lower leg*| node. That is, in all the documents where |*Gout*| is present, the most frequent class is "Y". In the 68 documents where |*Gout*| is present, we see that in 66 of these |*Entire right lower leg*| is *not* present. 65 of these documents are classified as "Y", and the algorithm did not find a feature which separates the one document not classified as "Y", hence we have a document in the training set that at this stage is misclassified by the decision tree. (This is the reason that we see "65/1" in the leaf node, meaning that 65 documents of the training set gets classified correctly and 1 incorrectly by this decision tree).

In figure 7.4 we see a similar decision tree for Gout that is made when

[6]This is not entirely true. Because of our naïve assumption that a concept only occurs once in a document, we do not need to count occurrences of each document. This means that there is only one document where the last mentioning of |*Gout*| is negated.

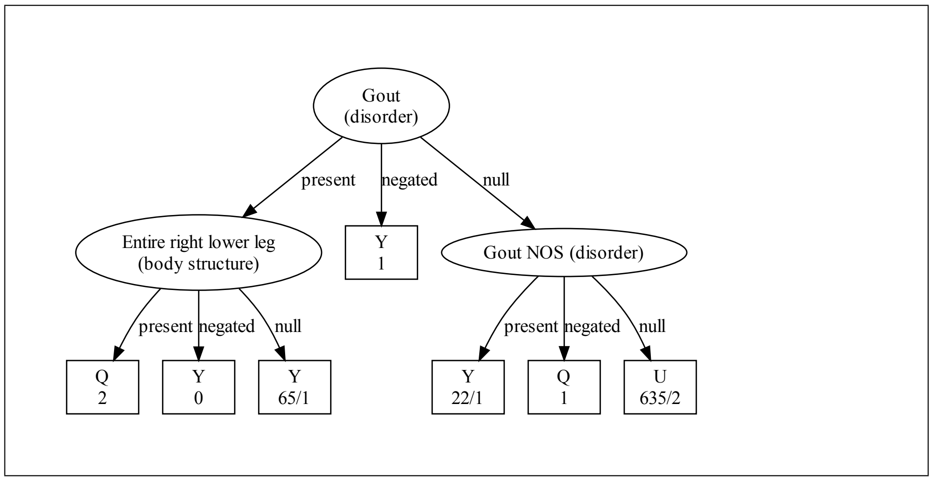

Figure 7.3: Decision tree for textual classification of gout. Nominal feature values.

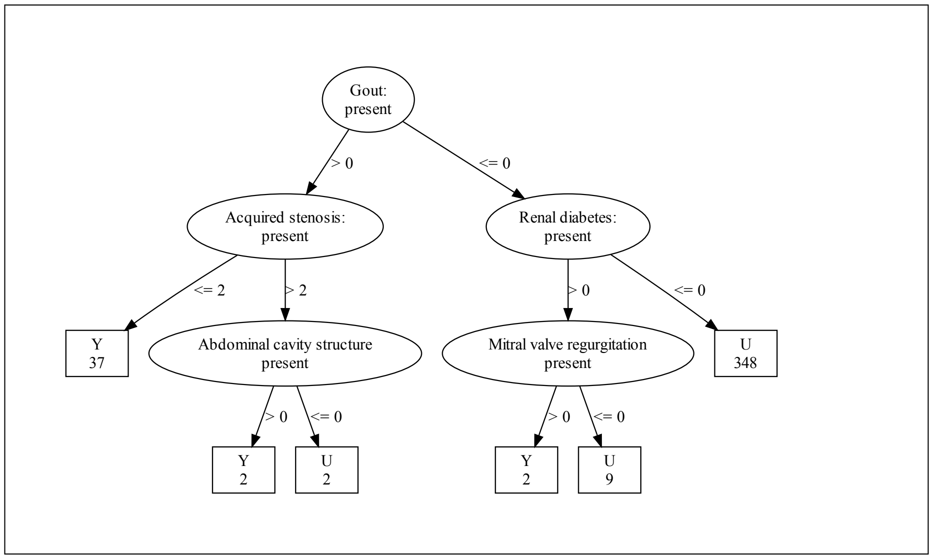
Gout: present
> 0
<= 0
Acquired stenosis: present
Renal diabetes: present
<= 2
> 2
> 0
<= 0
Y 37
Abdominal cavity structure present
Mitral valve regurgitation present
U 348
> 0
<= 0
> 0
<= 0
Y 2
U 2
Y 2
U 9

Figure 7.4: Decision tree for textual classification of gout. Frequency feature values.

using frequency feature values. The decision tree works in a similar manner, but instead of making a decision on a particular feature value (i.e. an integer) it uses "bigger than" and "smaller or equal than" operators. For instance, if a document mentions the term $|Gout|$ more than 0 times and $|$*Acquired stenosis: present*$|$ is mentioned more than 2 times, it decides that the patient must have gout.

7.3 Discussion

We have now explored how we can use the output from up-to-date IE software to classify patient records with respect to obesity and 15 co-morbidities. We have done this in a fairly straightforward fashion, which is easily implementable for other parties. The standard cTAKES pipeline was used in combination with a homemade section segmentation, and the extracted information served as input to a machine learning algorithm based on decision trees. With this system we got an average score comparing our scores with the shared task participants. Given that we had no input from domain experts (i.e. medical personnel) to create rules or key-word lists and the fact that we used a general IE system (as opposed to engineering a system geared to this particular task), we conclude that our system performed decently.

When comparing our results to the results of the participants in the same shared task we got the training and evaluation data from, we find that, at least for the textual judgements, we got better than average scores: According to Mishra et al. (2009) the average F-macro score of the participants in the textual part of the shared task was 0.56. Comparably, we got a F-macro of 0.6052. So even if we did not beat the top ten performances as described by (Uzuner, 2008) it seems that our system is performing well.

We did not find similar average scores for the intuitive task in the literature, and we are hence not able to compare our scores on the intuitive judgement part of the shared task with the participants. In Uzuner (2008) we learn that participants utilizing machine learning in their classification systems (as opposed to rule-based) did a better job on the intuitive task compared to the textual. Also, the top ten participants in the textual task had F-macro scores in the range 0.6140 - 0.8052, while the range of the top ten participants in the intuitive task was 0.6287 - 0.6745. Therefore, there is reason to believe that our system performed decently also for the intuitive task; we also used machine learning algorithms and the top ten participants in the intuitive task had generally lower scores with respect to the textual task. Our best F-macro for the intuitive task was 0.6142.

Many of the participants took advantage of domain experts when creating their systems, either for writing detection rules used for the classification or for writing a word-list for the different morbidities over which words and/or measures signify a given morbidity (Uzuner, 2008). We did not use input from domain experts, but instead used software utilizing an ontology and a terminology (SNOMED-CT and RxNorms) to detect bio-medical entities mentioned in the text. This software structures the information in the text by adding meta-data about the clinical terms used in the document. The structured information serve as a computable insight into the content of each document.

In classification tasks such as these, it is not uncommon to use complicated queries to retrieve the possible relevant documents for a given disease (see for instance (Sager et al., 1994)). At least for the textual part of the task, this method can attain a high recall (finding all the relevant documents), but engineering a query which recognises all relevant entities (medications, more granular variants of the morbidity, etc.) needs the assistance from skilful medical personnel who knows which entities are likely to be associated with what morbidities. A part of this task could perhaps be performed automatically by using the taxonomical knowledge within UMLS or SNOMED-CT. This was for instance done by Meystre (2009), but not without input from experts. Instead we went for a machine learning algorithm, teaching the computer which medical entities are likely to be seen in the case of a given morbidity.

Machine learning algorithms could be used because we had access to training material prior to the evaluation of our system, something which is likely not a resource in clinical settings in general. The algorithm we chose was the Weka-implementation of the well-used C4.5 decision tree (Hall et al., 2009). The positive aspects of using a decision tree are that we gain a unique view of why the computer selects a given class. We can, as we have seen, read the highly readable tree and see what specific feature values are used to classify a document/patient. The negative aspect is a possible lack of performance, but this needs an in-depth study.

We saw in figure 7.4 that the decision trees use entities that might be unrelated to the given morbidity when deciding whether or not a patient has the morbidity. Whether these entities really are unrelated to the disease has to be, of course, evaluated by domain experts. Either way it is well known that sparseness in the training data leads to low performance, and the decision trees would possibly find better clues for a given morbidity if more training material were available.

Sparseness is indeed a problem for the very low-frequent classes Q in the intuitive task and Q and N in the textual task. To illustrate how the performance decreases due to these low frequent classes, we can eliminate the classes Q and U and calculate new scores. For the intuitive task we eliminated the class Q, and said that every instance of Q really is an instance of Y. Without training any new classifiers, but merely by considering Q as Y using the best final classifier, we get the results as reported in table A.11 on page 112. We see that the system both get a much higher P-Macro (0.9294), R-Macro (0.9137) and F-Macro (0.9210). In a similar manner we can replace the classes U and Q in the textual task, and consider U as N and Q as Y. Here as well we get much higher P-Macro (0.9499), R-Macro (0.9397) and F-Macro (0.9446) scores.

These new numbers look, of course, much more promising, but what do they mean? In the intuitive task we are simply regarding all questionable cases as cases where the patient has the diagnose. In many use-cases this is indeed what we want. For instance, if we want to find all patients with a given diagnose it is often better to have false positives than false negatives, i.e. it is in these settings better to falsely report a patient having a disease than ignoring a patient that actually has the disease. In use-cases such as these, it is also important that we get a (near) perfect intuitive recall for the class Y: We want a perfect recall because we want to include every patient that might have the disease in question. We are interested in the intuitive task because we can then also identify patients with the disease without it being spelled out in one of the patients records. The R-Macro for the intuitive task reported above was 0.9137 %, but this is averaged between the two classes Y and N. Since we, in this, scenario do not mind including patients without the disease, but are more concerned with finding all patients with the disease, it is more interesting to compute the recall for the class Y in itself (over all the 16 morbidities). By using the (undocumented) method which prints the confusion matrix in the i2b2 2008 shared task evaluation script, we find that the number of true positives is 1980 and the number of false negatives is 173 (adding up over all morbidities), giving us a recall of 0.9195. This means that around 92 % of the patients which have or might have the disease are discovered. The recall of the class Y in the textual task was 0.9361.

Whether these are good enough results to make use of such a system in a clinical setting is not up to us to consider. We must also have in mind that these figures are not "natural". Our sample documents are not randomly drawn from a EHR/EMR, but include only patients with obesity or one of its 15 co-morbidities as defined in the shared task. These numbers may therefore be artificially high.

Chapter 8

Conclusions and the Road Ahead

Throughout this thesis we have seen how one can build an IE system from the ground up, focusing on algorithms and systems developed for the clinical domain. This has been done in such a manner that prior knowledge in the field was not required by the reader. We then evaluated how well an off-the-shelf system performed on a shared task, coupling it with a decision tree algorithm. Comparing our general-purpose IE system with systems geared towards a specific IE task, we found that our system performed quite well. We learned that the main resource needed to strengthen such systems is annotated clinical data which can be used to develop more precise NLP and IE modules.

In this final chapter we want to discuss all of the above with a close eye on our problem formulations given in section 2.5. Then we want to discuss what should be done if one is to develop similar IE systems for a different language.

8.1 Conclusions

This thesis includes a diversity of topics, geared towards attaining a deep understanding of how one can capture information from unstructured clinical text. A chief contribution has been to clarify the picture of this inter-disciplinary research field thus removing the magic of AI in natural language processing. We have often glanced towards the role of ontologies in such a task; In section 3.2 we saw how ontologies can be used as a tool when uncovering clinical named entities in the text, by using the textual definitions of concepts for finding the entities. In chapter 4 we explained how an ontology can expand the uncovered information, thus enabling the comparison and retrieval of patient information not explicitly stated in clinical text. We have also seen how an ontology can function as a target for entity extraction, by mapping the extracted entities to the ontology. Finally we evaluated a system which uses SNOMED CT and RxNorms to extract entities in clinical text. Comparing our general IE system with systems the participants of the shared task, we found that the performance of our system was quite decent.

We have predominantly focused on ontologies as a tool for finding and annotating clinical entities. Instead of, for instance, using machine learning techniques on annotated text to build such a term-mapper, the idea is to use the textual descriptions of the clinical term within the ontology. We described how Patrick et al. (2007) mapped n-grams from free text into ontology concepts, by measuring the overlap in (normalized) tokens of the n-gram and the textual description of the term. We also used a system which mapped look-up windows in free text to ontology concepts by querying the dictionary with the tokens in the lookup-window.

The benefit of this approach, in contrast with machine learning techniques, is that we are relieved of the burden of having to acquire and annotate huge amounts of clinical records. So the need to de-identify a number of clinical records, as well as the need to hire domain experts to annotate these records, declines. On the other hand, it might be that using unsupervised techniques such as Patrick et al. (2007) and cTAKES (Schuler et al., 2008) degrades the performance. This should be investigated more thoroughly by evaluating the different systems on the same evaluation set. We expanded the cTAKES system with a machine learning module, due to difficulties evaluating the unsupervised method of finding clinical entities: We did not have access to any corpora annotated with clinical entities. The machine-learning module used the output from the system to classify morbidities in an corpora annotated with these morbidities.

We also sought to find out what kind of resources already exist, and gave an overview of some of the important resources in chapter 5. We covered SNOMED CT and UMLS when discussing ontologies and terminologies, and saw how they are built up with semantic constrains and definitions. Different IE systems developed for the clinical domain were also presented, and we see that they all share a basic strategy: In a pipeline fashion, the systems first uncover linguistic entities and structure, normalize the tokens and use these linguistic elements as input to the IE part of the system. Finally we covered the corpora available for research purposes. These were predominantly annotated with clinical information, such as ICD-9 codes, smoking status, morbidities, medications and clinical concepts. Some corpora were also annotated with IE-specific information, such as negations, speculations and co-references.

But, how can such a system be of any benefit? We defined some use-cases in section 2.4, but we are unfortunately not in any position to estimate the fruitfulness of the system we evaluated in any clinical or research setting: Such a study must include analysis performed by domain experts, likely medical scientists. But we have exposed what such a system is capable of, given the limited means currently available — hopefully in terms apprehensible for such domain experts with little experience or knowledge of computational linguistics. We hope this encourages further research within this highly interdisciplinary field.

8.2 The Road Ahead

Future work

Our experiments could have been improved by engineering other, and perhaps better, features. For instance we could, instead of extracting *all* uncovered entities in the text as features, focused on entities related to the different morbidities. Reducing the amount of features to only include relevant ones might help because the decision tree algorithm is confused as to which feature indicates sparse classes. We could also benefited from term lists and the detection of specific phrases which is likely to signify the presence of one of the morbidities. Such terms and phrases could then be used to extract supplementary features. Nevertheless, locating the relevant entities, terms or phrases necessitates help from domain experts: We were not in a situation to determine which of the uncovered entities in the text is important for each morbidity.

We could also have utilized the SNOMED CT ontology to a larger extent, for instance by expanding the extracted information as described in section 4.3 thus harvesting more features. SNOMED CT could also help us uncover which entities are related to the morbidities we were trying to detect, by exploring the relations involving the concepts defining each of the morbidities. But even this task would have benefited from help of a domain expert, to find the relevant initial concepts describing the morbidities.

In our experiments we have primarily used machine learning based on decision trees to identify each morbidity. We would expect even better performance if we used other algorithms, such as support vector machines. We did not find the time to include additional experiments in our work, so these experiments are left to future research.

In a more general perspective, one could check if an IE system which utilizes ontologies does it better than an IE system which does not utilize any ontology, in other words: Does ontologies contribute in finding named entities? We have not seen any good way to investigate this directly, because of the lack of corpora to evaluate this and because it is difficult to compare a system which utilizes ontologies and a system that does not — they would necessarily be different on other accounts as well.

When mapping terms from free text to an ontology, our experiments and research indicates that using the textual definitions of the ontologies directly — such as the query method used in cTAKES — enable us to capture a wider range of entities to extract. Using rule-based approaches limits the variety of entities by the fact that someone needs to write these rules to capture the entities. Using algorithms based on machine-learning necessitates a large corpus annotated with named entities.

Developing State of the Art IE Systems

With regards to developing new IE systems, the conclusion should now be apparent: We need access to clinical records. We have included a discussion of this in appendix B. The next step is annotating these records with linguistic and clinical information, so that we can train and evaluate the different modules making up an IE system. Moreover, the importance of ontologies should not be underestimated. As we have seen, access to ontologies with textual descriptions written in the same language as the records proves to be a valuable tool, and can even be used without developing sophisticated NLP machinery in advance — as we saw in the n-gram matcher of Patrick et al. (2007).

There are good benefits of basing the IE system on a standardized pipeline-system for handling unstructured data such as UIMA. There is less effort required when expanding existing systems, altering modules for own needs and sharing components with other teams, as we witnessed first-hand when working on cTAKES. We would therefore encourage new projects to use existing standards and frameworks when developing NLP and IE modules.

We have also witnessed the need of keeping a consistent annotation scheme. As we pointed out, a POS-tagger is best served data tokenized in the same manner as the training data. A parser or a chunker will not work if used on data annotated with different POS tags than what it is trained on. But even seemingly small choices, as when the sentence splitter in cTAKES decides that each End of Line (EOL) also end a sentence, have severe consequences. When we worked on data where EOL frequented within sentences, we ran into several problems: We could remove EOL in the documents, but not without a loss of information — there was always an EOL before new sections, which could be used when segmenting sections in the patient summaries. We tried to alter the behaviour of the sentence segmentation module in cTAKES, but this gave unrecognizable input to the POS-tagger which ended in obscure POS-tagging (EOL could be tagged as a noun, for instance). So each decision should be taken with great care, and be followed consistently throughout the project.

Automatic structuring of information in clinical reports in the future thus rests on important actions we need to take today: Translating or developing a biomedical ontology for the target language is an important step. Releasing clinical data for scientific purposes seems necessary. Therefore, we see no need to delay any project aiming at IE in the clinical domain.

Bibliography

Aramaki, E., Y. Miura, M. Tonoike, T. Ohkuma, H. Mashuichi, and K. Ohe (2009). Text2table: medical text summarization system based on named entity recognition and modality identification. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, BioNLP '09, Stroudsburg, PA, USA, pp. 185–192. Association for Computational Linguistics.

Bashyam, V. and R. Taira (2007, 1 2007-april 5). Identifying anatomical phrases in clinical reports by shallow semantic parsing methods. In *Computational Intelligence and Data Mining, 2007.*, pp. 210 –214. IEEE.

Berges, I., J. Bermudez, A. Goñi, and A. Illarramendi (2010). Semantic interoperability of clinical data. In *Proceedings of the First International Workshop on Model-Driven Interoperability*, MDI '10, New York, NY, USA, pp. 10–14. ACM.

Bodenreider, O., B. Smith, A. Kumar, and A. Burgun (2007). Investigating subsumption in SNOMED CT: An exploration into large description logic-based biomedical terminologies. *Artificial Intelligence in Medicine 39*(3), 183–195.

Brants, T. (2000). Tnt - a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, Seattle, WA.

Browne, A. C., A. T. McCray, and S. Srinivasan (2000). *The SPECIALIST Lexicon*. Lister Hill National Center for Biomedical Communications.

Bush, J. (2003, June). Open-source software: Just what the doctor ordered? *American Academy of Family Physicians*. Downloaded from http://www.aafp.org/fpm/2003/0600/p65.html April 2012.

Chapman, W. W., W. Bridewell, P. Hanbury, G. F. Cooper, and B. G. Buchanan (2001). A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform 2001*, 34–301.

Chapman, W. W., D. Chu, and J. N. Dowling (2007, June 29). ConText: An Algorithm for Identifying Contextual Features from Clinical Text. In *Proceedings of the Workshop on BioNLP 2007, Biological, Translational,* *and Clinical Language Processing*, Prague, Czech Republic. The Association for Computational Linguistics.

Chapman, W. W., P. M. Nadkarni, L. Hirschman, L. W. D'Avolio, G. K. Savova, and O. Uzuner (2011). Overcoming barriers to NLP for clinical text: The role of shared tasks and the need for additional creative solutions. *J Am Med Inform Assoc 18*, 540–3.

Childs, L. C., R. Enelow, L. Simonsen, N. H. Heintzelman, K. M. Kowalski, and R. J. Taylor (2009, July). Description of a Rule-based System for the i2b2 Challenge in Natural Language Processing for Clinical Data. *Journal of the American Medical Informatics Association 16*(4), 571–575.

Cho, P. S., R. K. Taira, and H. Kangarloo (2003). Automatic section segmentation of medical reports. *AMIA Annual Symposium proceedings*, 155–159.

Coden, A. R., S. V. Pakhomov, R. K. Ando, P. H. Duffy, and C. G. Chute (2005). Domain-specific language models and lexicons for tagging. *Journal of Biomedical Informatics 38*(6), 422 – 430.

Dalianis, H., M. Hassel, and S. Velupillai (2009, October 14-16). The Stockholm EPR Corpus - Characteristics and Some Initial Findings. In *Proceedings of ISHIMR 2009, Evaluation and implementation of e-health and health information initiatives: international perspectives. 14th International Symposium for Health Information Management Research*, Kalmar, Sweden.

Dwivedi, S. K. and P. P. Sukhadeve (2011). Rule based Part of speech Tagger for Homoeopathy Clinical realm. *International Journal of Computer Science Issues*, 350 – 354.

Estelrich, A. (2010). epSOS Semantic Interoperability. In *Semantic Days 2010.*

Farkas, R., V. Vincze, G. Móra, J. Csirik, and G. Szarvas (2010, July). The CoNLL-2010 Shared Task: Learning to Detect Hedges and their Scope in Natural Language Text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, Uppsala, Sweden, pp. 1–12. Association for Computational Linguistics.

Ferrucci, D. and A. Lally (2004, September). UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering 10*(3-4), 327–348.

Friedman, C., G. Hripcsak, W. DuMouchel, S. B. Johnson, and P. D. Clayton (1995). Natural language processing in an operational clinical information system. *Natural Language Engineering 1*, 83–108.

Götz, T. and O. Suhre (2004). Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal 43*(3), 476–489.

Guergana, K. S., K. Kipper-Schuler, J. D. Buntrock, and C. G. Chute (2008). UIMA-based Clinical Information Extraction System. *LREC 2008: Towards enhanced interoperability for large HLT systems: UIMA for NLP*.

Hahn, U., E. Buyko, K. Tomanek, S. Piao, J. McNaught, Y. Tsuruoka, and S. Ananiadou (2007). An annotation type system for a data-driven NLP pipeline. In *Proceedings of the Linguistic Annotation Workshop*, Stroudsburg, PA, USA, pp. 33–40. Association for Computational Linguistics.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten (2009, November). The weka data mining software: an update. *SIGKDD Explor. Newsl. 11*, 10–18.

Hayes, P. (Ed.) (2004, February). *RDF Semantics.* W3C Recommendation. World Wide Web Consortium.

Hitzler, P., M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph (Eds.) (27 October 2009). *OWL 2 Web Ontology Language: Primer.* W3C Recommendation. Available at http://www.w3.org/TR/owl2-primer/.

Hitzler, P., R. Sebastian, and M. Krötzsch (2009). *Foundations of Semantic Web Technologies.* London: Chapman & Hall/CRC.

Hobbs, J. R. (2002). Information extraction from biomedical text. *Journal of Biomedical Informatics 35*(4), 260–264.

IHTSDO, I. H. T. S. D. O. (2012). Snomed clinical terms user guide 2012. *Development* (July).

Jurafsky, D. and J. H. Martin (2008, May). *Speech and Language Processing (2nd Edition)* (2 ed.). Prentice Hall.

Klyne, G. and J. J. Carroll (2004, February). Resource description framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210.

Levin, M. A., M. Krol, A. M. Doshi, and D. L. Reich (2007). Extraction and mapping of drug names from free text to a standardized nomenclature. *Proceedings of the AMIA Annual Symposium*, 438–442.

Liu, H., Y. A. Lussier, and C. Friedman (2001, August). Disambiguating Ambiguous Biomedical Terms in Biomedical Narrative Text: An Unsupervised Method. *Journal of Biomedical Informatics 34*(4), 249–261.

Manning, C. D., P. Raghavan, and H. Schütze (2008). *Introduction to Information Retrieval.* New York, NY, USA: Cambridge University Press.

Meystre, S. M. (2009). Detecting intuitive mentions of diseases in narrative clinical text. In *Proceedings of the 12th Conference on Artificial Intelligence in Medicine: Artificial Intelligence in Medicine*, AIME '09, Berlin, Heidelberg, pp. 216–224. Springer-Verlag.

Meystre, S. M., G. K. Savova, K. C. Kipper-Schuler, and J. F. Hurdle (2008). Extracting information from textual documents in the electronic health record: a review of recent research. *Yearbook of medical informatics 47*(Suppl 1), 128–144.

Mishra, N. K., D. M. Cummo, J. J. Arnzen, and J. Bonander (2009, July). A Rule-based Approach for Identifying Obesity and Its Comorbidities in Medical Discharge Summaries. *Journal of the American Medical Informatics Association 16*(4), 576–579.

MITRE (2006). Electronic Health Records Overview. *NIH National Center for Research Resources*.

Nadeau, D. and S. Sekine (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes 30*(1), 3–26.

Ogren, P., G. Savova, and C. Chute (2008). Constructing evaluation corpora for automated clinical named entity recognition. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Oliver, D. E. and R. B. Altman (1994). Extraction of SNOMED concepts from medical record texts. In *Proceedings of the 18th Annual SCAMC*, Washington, pp. 179–183. McGraw Hill.

OWL Working Group, W. (27 October 2009). *OWL 2 Web Ontology Language: Document Overview.* W3C Recommendation. Available at http://www.w3.org/TR/owl2-overview/.

Pantazos, K., S. Lauesen, and S. Lippert (2011). De-identifying an EHR Database - Anonymity, Correctness and Readability of the Medical Record. *Stud Health Technol Inform 169*, 862–6.

Patrick, J., Y. Wang, and P. Budd (2007). An automated system for conversion of clinical notes into snomed clinical terminology. In *Proceedings of the fifth Australasian symposium on ACSW frontiers - Volume 68*, ACSW '07, Darlinghurst, Australia, Australia, pp. 219–226. Australian Computer Society, Inc.

Pestian, J. P., C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch (2007). A shared task involving multi-label classification of clinical free text. In *BioNLP '07: Proceedings of the Workshop on BioNLP 2007*, Morristown, NJ, USA, pp. 97–104. Association for Computational Linguistics.

Phillips, L. (2000). The Double Metaphone Search Algorithm. *CC Plus Plus Users Journal 18*(6), 38–43.

Ruch, P., R. Baud, and A. Geissbühler (2003, 2003 Sep-Oct). Using lexical disambiguation and named-entity recognition to improve spelling correction in the electronic patient record. *Artificial intelligence in medicine 29*, 169–84.

Sager, N., M. Lyman, C. Bucknall, N. Nhan, and L. J. Tick (1994, March). Natural Language Processing and the Representation of Clinical Data. *Journal of the American Medical Informatics Association 1*(2), 142.

Sager, N., M. Lyman, N. T. Nhàn, and L. J. Tick (1994). Automatic encoding into SNOMED III: a preliminary investigation. *Proc Annu Symp Comput Appl Med Care*, 230–4.

Savova, G., K. Kipper-Schuler, J. Buntrock, and C. Chute (2008). Uima-based clinical information extraction system. In *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP*, pp. 39–42. UIMA Workshop, LREC 2008.

Savova, G. K., J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler, and C. G. Chute (2010). Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association 17*(5), 507–513.

Schuler, K., V. Kaggal, J. J. Masanz, P. V. Ogren, and G. K. Savova (2008). System evaluation on a named entity corpus from clinical notes. In *Language Resources and Evaluation Conference.*

Sevenster, M., R. van Ommering, and Y. Qian (2012). Automatically Correlating Clinical Findings and Body Locations in Radiology Reports Using MedLEE. *Journal of Digital Imaging 25*, 240–249. 10.1007/s10278-011-9411-0.

Spyns, P., N. Nhn, E. Baert, N. Sager, and G. De Moor (1998). Medical Language Processing applied to extract clinical information from Dutch medical documents. In C. Cesnik, C. Safran, and P. Degoulet (Eds.), *Proceedings of the 9th World Congress on Medical Informatics (MEDINFO98)*, Amsterdam, pp. 685–689.

Tao, C., W.-Q. Q. Wei, H. R. Solbrig, G. Savova, and C. G. Chute (2010). CNTRO: A Semantic Web Ontology for Temporal Relation Inferencing in Clinical Narratives. *Proceedings of the AMIA Annual Symposium 2010*, 787–791.

Tolentino, H. D., M. M. Matters, W. Walop, B. Law, W. Tong, F. Liu, P. Fontelo, K. Kohl, and D. C. Payne (2007, February). A UMLS-based spell checker for natural language processing in vaccine safety. *BMC Medical Informatics and Decision Making 7*(3), 3+.

Tveit, A., O. Edsberg, T. B. Røst, A. Faxvaag, O. Nytrø, T. Nordgård, M. T. Ranang, and A. Grimsmo (2004). Anonymization of general practioner medical records. *Proceedings of the HelsIT04 Conference* (7489).

UMLS (Ed.) (2009). *UMLS Reference Manual*. Bethesda (MD): National Library of Medicine (US).

Uzuner, O. (2008). Second i2b2 workshop on natural language processing challenges for clinical records. *Proceedings of the AMIA Annual Symposium*, 1252–1253.

Uzuner, O. (2009). Recognizing obesity and comorbidities in sparse data. *Journal of the American Medical Informatics Association 16*(4), 561–570.

Uzuner, O., A. Bodnari, S. Shen, T. Forbush, J. Pestian, and B. R. South (2012). Evaluating the state of the art in coreference resolution for electronic medical records. *Journal of the American Medical Informatics Association*.

Uzuner, O., I. Goldstein, Y. Luo, and I. Kohane (2008, January). Identifying patient smoking status from medical discharge records. *Journal of the American Medical Informatics Association : JAMIA 15*(1), 14–24.

Uzuner, O., Y. Luo, and P. Szolovits (2007). Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association : JAMIA 14*(5), 550–563.

Uzuner, z., I. Solti, F. Xia, and E. Cadag (2010). Community annotation experiment for ground truth generation for the i2b2 medication challenge. *JAMIA 17*(5), 519–523.

Uzuner, z., B. R. South, S. Shen, and S. L. DuVall (2011). 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *JAMIA 18*(5), 552–556.

Velupillai, S. (2012, April). *Shades of Certainty – Annotation and Classification of Swedish Medical Records*. Doctoral thesis, Department of Computer and Systems Sciences, Stockholm University, Stockholm, Sweden.

Vincze, V., G. Szarvas, R. Farkas, G. Mora, and J. Csirik (2008). The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics 9*(Suppl 11), S9+.

Wu, Y., S. Rosenbloom, J. Denny, R. Miller, S. Mani, D. Giuse, and H. Xu (2011). Detecting abbreviations in discharge summaries using machine learning methods. Volume 2011.

Xu, H., P. Stetson, and C. Friedman (2007). A study of abbreviations in clinical notes. In J. Teich, J. Suermondt, and G. Hripcsak (Eds.), *Proceedings of the AMIA Annual Symposium*, Washington DC, USA, pp. 821–825. Bethesda, MD: AMIA Press.

Yang, Y. and C. G. Chute (1993). An application of least squares fit mapping to text information retrieval. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, New York, NY, USA, pp. 281–290. ACM.

Zhou, L. and G. Hripcsak (2007). Temporal reasoning with medical data — a review with emphasis on medical natural language processing. *Journal of Biomedical Informatics 40*(2), 183 – 202.

Appendix A

Evaluations

In the following pages we find the system evaluations of our classifiers. Tables A.1 to A.7 report on the performance during our temporary evaluations. Tables A.9 and A.10 report on the performance on the final test set, with two different systems. Table A.11 reports on the experimental binary classification.

For convenience, table A.8 comparing the different classifiers for each morbidity is included. These are the classifiers trained on the initial training set and evaluated on the held-out set (i.e. it summarizes all F-macro values from table A.1 to table A.7).

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8198	0.8156	0.8198	0.8112	0.8198	0.8131
Depression	0.9391	0.9367	0.9391	0.8447	0.9391	0.8821
Hypertriglyceridemia	0.9565	0.6909	0.9565	0.7365	0.9565	0.7109
Gallstones	0.9829	0.9594	0.9829	0.9594	0.9829	0.9594
OSA	0.9310	0.8919	0.9310	0.5867	0.9310	0.5716
Asthma	0.9636	0.9487	0.9636	0.9009	0.9636	0.9228
CAD	0.8991	0.9305	0.8991	0.6017	0.8991	0.5994
PVD	0.9636	0.9780	0.9636	0.9130	0.9636	0.9411
Gout	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Diabetes	0.9550	0.9589	0.9550	0.9347	0.9550	0.9456
CHF	0.7476	0.7313	0.7476	0.7399	0.7476	0.7343
Venous Insufficiency	0.9143	0.7551	0.9143	0.6842	0.9143	0.7120
GERD	0.9231	0.9518	0.9231	0.8621	0.9231	0.8947
OA	0.9057	0.8397	0.9057	0.8397	0.9057	0.8397
Hypercholesterolemia	0.8300	0.8312	0.8300	0.8276	0.8300	0.8286
Hypertension	0.7383	0.5614	0.7383	0.5351	0.7383	0.5325
Intuitive	0.9064	0.9286	0.9064	0.5942	0.9064	0.5947

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8908	0.8945	0.8908	0.8796	0.8908	0.8853
Depression	0.9328	0.8813	0.9328	0.7904	0.9328	0.8273
Hypertriglyceridemia	0.9916	0.9958	0.9916	0.7500	0.9916	0.8312
Gallstones	0.9580	0.9726	0.9580	0.4642	0.9580	0.4681
OSA	0.9322	0.9277	0.9322	0.4325	0.9322	0.4301
Asthma	0.9746	0.9713	0.9746	0.6250	0.9746	0.6311
CAD	0.8205	0.4268	0.8205	0.4346	0.8205	0.4306
PVD	0.9832	0.9673	0.9832	0.9673	0.9832	0.9673
Gout	0.9832	0.9782	0.9832	0.6667	0.9832	0.6555
Diabetes	0.8739	0.9236	0.8739	0.4427	0.8739	0.4325
CHF	0.7542	0.5083	0.7542	0.5216	0.7542	0.5137
Venous Insufficiency	0.9748	0.7857	0.9748	0.9870	0.9748	0.8570
GERD	0.9748	0.9766	0.9748	0.6466	0.9748	0.6449
OA	0.9328	0.8961	0.9328	0.8656	0.9328	0.8798
Hypercholesterolemia	0.8898	0.6164	0.8898	0.5998	0.8898	0.6059
Hypertension	0.6864	0.7112	0.6864	0.3718	0.6864	0.3705
Textual	0.9099	0.4497	0.9099	0.4503	0.9099	0.4500

Table A.1: Scores on Named Entities only. Decision tree.

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.7027	0.7010	0.7027	0.6731	0.7027	0.6758
Depression	0.8174	0.6468	0.8174	0.5737	0.8174	0.5857
Hypertriglyceridemia	0.9652	0.9826	0.9652	0.5000	0.9652	0.4912
Gallstones	0.8462	0.4381	0.8462	0.4806	0.8462	0.4583
OSA	0.8966	0.9643	0.8966	0.4222	0.8966	0.4548
Asthma	0.8636	0.9312	0.8636	0.5312	0.8636	0.5219
CAD	0.7798	0.8693	0.7798	0.4957	0.7798	0.5015
PVD	0.8364	0.8460	0.8364	0.6247	0.8364	0.6526
Gout	0.8596	0.9298	0.8596	0.5000	0.8596	0.4623
Diabetes	0.7838	0.8422	0.7838	0.6553	0.7838	0.6709
CHF	0.7573	0.7446	0.7573	0.7585	0.7573	0.7478
Venous Insufficiency	0.9143	0.9567	0.9143	0.5500	0.9143	0.5683
GERD	0.7500	0.8713	0.7500	0.5517	0.7500	0.5199
OA	0.8208	0.6635	0.8208	0.5206	0.8208	0.4979
Hypercholesterolemia	0.6800	0.6822	0.6800	0.6740	0.6800	0.6736
Hypertension	0.7757	0.6405	0.7757	0.5148	0.7757	0.4746
Intuitive	0.8175	0.8646	0.8175	0.5129	0.8175	0.5200

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.7227	0.7265	0.7227	0.6908	0.7227	0.6948
Depression	0.8824	0.9407	0.8824	0.5333	0.8824	0.5310
Hypertriglyceridemia	0.9832	0.9916	0.9832	0.5000	0.9832	0.4958
Gallstones	0.8319	0.7649	0.8319	0.2569	0.8319	0.2519
OSA	0.8729	0.9190	0.8729	0.3100	0.8729	0.3276
Asthma	0.8644	0.9544	0.8644	0.3542	0.8644	0.3481
CAD	0.7179	0.8833	0.7179	0.3704	0.7179	0.3605
PVD	0.8655	0.8098	0.8655	0.5784	0.8655	0.5993
Gout	0.8487	0.9492	0.8487	0.3529	0.8487	0.3428
Diabetes	0.7563	0.9125	0.7563	0.3183	0.7563	0.3203
CHF	0.7119	0.8160	0.7119	0.4944	0.7119	0.4824
Venous Insufficiency	0.9664	0.9832	0.9664	0.5000	0.9664	0.4915
GERD	0.8319	0.9435	0.8319	0.3500	0.8319	0.3342
OA	0.8319	0.9153	0.8319	0.5238	0.8319	0.4992
Hypercholesterolemia	0.7034	0.8009	0.7034	0.4744	0.7034	0.4708
Hypertension	0.7203	0.7294	0.7203	0.3729	0.7203	0.3681
Textual	0.8197	0.8941	0.8197	0.3834	0.8197	0.3873

Table A.2: Scores on Named Entities only. Naïve Bayes.

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.7928	0.8089	0.7928	0.7659	0.7928	0.7740
Depression	0.9130	0.8709	0.9130	0.8092	0.9130	0.8353
Hypertriglyceridemia	0.9391	0.6336	0.9391	0.7275	0.9391	0.6658
Gallstones	0.9231	0.8137	0.9231	0.8329	0.9231	0.8229
OSA	0.8879	0.5332	0.8879	0.5700	0.8879	0.5480
Asthma	0.9000	0.8800	0.9000	0.6822	0.9000	0.7330
CAD	0.8991	0.9299	0.8991	0.5988	0.8991	0.5976
PVD	0.9273	0.9330	0.9273	0.8421	0.9273	0.8778
Gout	0.9912	0.9949	0.9912	0.9688	0.9912	0.9813
Diabetes	0.5676	0.4593	0.5676	0.4584	0.5676	0.4557
CHF	0.6699	0.6712	0.6699	0.6838	0.6699	0.6646
Venous Insufficiency	0.9143	0.7517	0.9143	0.7289	0.9143	0.7396
GERD	0.8558	0.8689	0.8558	0.7625	0.8558	0.7938
OA	0.7830	0.6263	0.7830	0.6210	0.7830	0.6235
Hypercholesterolemia	0.7300	0.7370	0.7300	0.7344	0.7300	0.7298
Hypertension	0.6262	0.4789	0.6262	0.4777	0.6262	0.4780
Intuitive	0.8351	0.5451	0.8351	0.5324	0.8351	0.5380

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8151	0.8231	0.8151	0.7939	0.8151	0.8017
Depression	0.9412	0.8738	0.9412	0.8522	0.9412	0.8626
Hypertriglyceridemia	0.9748	0.4915	0.9748	0.4957	0.9748	0.4936
Gallstones	0.9328	0.9474	0.9328	0.4284	0.9328	0.4368
OSA	0.8814	0.6263	0.8814	0.3912	0.8814	0.3832
Asthma	0.9153	0.9145	0.9153	0.5142	0.9153	0.5395
CAD	0.8376	0.6859	0.8376	0.4448	0.8376	0.4401
PVD	0.9664	0.9345	0.9664	0.9345	0.9664	0.9345
Gout	0.9832	0.9782	0.9832	0.6667	0.9832	0.6555
Diabetes	0.5126	0.4471	0.5126	0.2045	0.5126	0.1999
CHF	0.7712	0.8468	0.7712	0.6105	0.7712	0.6528
Venous Insufficiency	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
GERD	0.9664	0.9725	0.9664	0.6299	0.9664	0.6343
OA	0.8235	0.6906	0.8235	0.6684	0.8235	0.6780
Hypercholesterolemia	0.8814	0.6135	0.8814	0.5970	0.8814	0.6051
Hypertension	0.7034	0.7159	0.7034	0.3651	0.7034	0.3605
Textual	0.8693	0.4495	0.8693	0.4401	0.8693	0.4432

Table A.3: Scores on Named Entities marked with segment.

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.7568	0.7502	0.7568	0.7446	0.7568	0.7468
Depression	0.9304	0.9296	0.9304	0.8197	0.9304	0.8619
Hypertriglyceridemia	0.9565	0.6533	0.9565	0.6160	0.9565	0.6316
Gallstones	0.9402	0.8855	0.9402	0.8117	0.9402	0.8433
OSA	0.8966	0.8534	0.8966	0.4789	0.8966	0.4949
Asthma	0.9273	0.9150	0.9273	0.7759	0.9273	0.8255
CAD	0.8899	0.9213	0.8899	0.5967	0.8899	0.5922
PVD	0.9091	0.8711	0.9091	0.8466	0.9091	0.8580
Gout	0.9825	0.9949	0.9825	0.9636	0.9825	0.9788
Diabetes	0.9369	0.9289	0.9369	0.9217	0.9369	0.9252
CHF	0.7961	0.7822	0.7961	0.7947	0.7961	0.7864
Venous Insufficiency	0.9048	0.7146	0.9048	0.6342	0.9048	0.6617
GERD	0.7788	0.7246	0.7788	0.7198	0.7788	0.7221
OA	0.8585	0.7625	0.8585	0.7287	0.8585	0.7433
Hypercholesterolemia	0.7600	0.7627	0.7600	0.7627	0.7600	0.7600
Hypertension	0.7664	0.6202	0.7664	0.5532	0.7664	0.5522
Intuitive	0.8768	0.5749	0.8768	0.5679	0.8768	0.5711

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8403	0.8619	0.8403	0.8153	0.8403	0.8261
Depression	0.8235	0.4994	0.8235	0.4997	0.8235	0.4946
Hypertriglyceridemia	0.9832	0.9916	0.9832	0.5000	0.9832	0.4958
Gallstones	0.8992	0.9042	0.8992	0.3902	0.8992	0.3963
OSA	0.8983	0.8939	0.8983	0.3831	0.8983	0.3877
Asthma	0.9322	0.9479	0.9322	0.5384	0.9322	0.5674
CAD	0.8547	0.5405	0.8547	0.5418	0.8547	0.5407
PVD	0.9076	0.8171	0.9076	0.8314	0.9076	0.8240
Gout	0.9328	0.9213	0.9328	0.5816	0.9328	0.5845
Diabetes	0.8824	0.9324	0.8824	0.4377	0.8824	0.4348
CHF	0.7288	0.8277	0.7288	0.5062	0.7288	0.4940
Venous Insufficiency	0.9832	0.8333	0.9832	0.9913	0.9832	0.8956
GERD	0.9244	0.6218	0.9244	0.5599	0.9244	0.5851
OA	0.8319	0.7032	0.8319	0.6361	0.8319	0.6569
Hypercholesterolemia	0.7203	0.5176	0.7203	0.4740	0.7203	0.4736
Hypertension	0.6695	0.6919	0.6695	0.3568	0.6695	0.3538
Textual	0.8635	0.4952	0.8635	0.4424	0.8635	0.4584

Table A.4: Scores on Named Entity only, minus family history, allergies and past history segments.

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8198	0.8156	0.8198	0.8112	0.8198	0.8131
Depression	0.9391	0.9367	0.9391	0.8447	0.9391	0.8821
Hypertriglyceridemia	0.9652	0.7410	0.9652	0.7410	0.9652	0.7410
Gallstones	0.9829	0.9594	0.9829	0.9594	0.9829	0.9594
OSA	0.9310	0.8919	0.9310	0.5867	0.9310	0.5716
Asthma	0.9727	0.9845	0.9727	0.9062	0.9727	0.9404
CAD	0.9083	0.9360	0.9083	0.6097	0.9083	0.6061
PVD	0.9636	0.9780	0.9636	0.9130	0.9636	0.9411
Gout	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Diabetes	0.9550	0.9695	0.9550	0.9265	0.9550	0.9446
CHF	0.7379	0.7223	0.7379	0.7322	0.7379	0.7253
Venous Insufficiency	0.9143	0.7551	0.9143	0.6842	0.9143	0.7120
GERD	0.9038	0.9220	0.9038	0.8382	0.9038	0.8684
OA	0.9057	0.8397	0.9057	0.8397	0.9057	0.8397
Hypercholesterolemia	0.8300	0.8312	0.8300	0.8276	0.8300	0.8286
Hypertension	0.7383	0.5614	0.7383	0.5351	0.7383	0.5325
Intuitive	0.9064	0.9288	0.9064	0.5939	0.9064	0.5946

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8824	0.8836	0.8824	0.8724	0.8824	0.8769
Depression	0.9412	0.8933	0.9412	0.8237	0.9412	0.8538
Hypertriglyceridemia	0.9916	0.9958	0.9916	0.7500	0.9916	0.8312
Gallstones	0.9580	0.9726	0.9580	0.4642	0.9580	0.4681
OSA	0.9322	0.9277	0.9322	0.4325	0.9322	0.4301
Asthma	0.9746	0.9904	0.9746	0.6250	0.9746	0.6396
CAD	0.8462	0.6941	0.8462	0.4479	0.8462	0.4460
PVD	0.9832	0.9673	0.9832	0.9673	0.9832	0.9673
Gout	0.9832	0.9782	0.9832	0.6667	0.9832	0.6555
Diabetes	0.8487	0.9091	0.8487	0.4334	0.8487	0.4195
CHF	0.7712	0.5181	0.7712	0.5321	0.7712	0.5249
Venous Insufficiency	0.9748	0.7857	0.9748	0.9870	0.9748	0.8570
GERD	0.9748	0.9766	0.9748	0.6466	0.9748	0.6449
OA	0.8908	0.8100	0.8908	0.8214	0.8908	0.8155
Hypercholesterolemia	0.8898	0.6164	0.8898	0.5998	0.8898	0.6059
Hypertension	0.6949	0.7184	0.6949	0.3757	0.6949	0.3748
Textual	0.9088	0.6989	0.9088	0.4508	0.9088	0.4499

Table A.5: Scores on Named Entities only, minus family history sections.

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8739	0.8753	0.8739	0.8637	0.8739	0.8682
Depression	0.9565	0.9502	0.9565	0.8947	0.9565	0.9195
Hypertriglyceridemia	0.9652	0.7410	0.9652	0.7410	0.9652	0.7410
Gallstones	0.9915	0.9952	0.9915	0.9643	0.9915	0.9791
OSA	0.9138	0.8669	0.9138	0.5800	0.9138	0.5534
Asthma	0.9727	0.9845	0.9727	0.9062	0.9727	0.9404
CAD	0.9174	0.9412	0.9174	0.6263	0.9174	0.6129
PVD	0.9545	0.9728	0.9545	0.8913	0.9545	0.9251
Gout	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Diabetes	0.9730	0.9720	0.9730	0.9641	0.9730	0.9679
CHF	0.7379	0.7320	0.7379	0.7486	0.7379	0.7314
Venous Insufficiency	0.9238	0.8030	0.9238	0.6895	0.9238	0.7294
GERD	0.8846	0.8792	0.8846	0.8248	0.8846	0.8462
OA	0.8962	0.8361	0.8962	0.7928	0.8962	0.8118
Hypercholesterolemia	0.8400	0.8407	0.8400	0.8418	0.8400	0.8399
Hypertension	0.7570	0.5814	0.7570	0.5324	0.7570	0.5223
Intuitive	0.9122	0.9360	0.9122	0.5948	0.9122	0.5984

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.9328	0.9372	0.9328	0.9245	0.9328	0.9297
Depression	0.9160	0.8179	0.9160	0.7808	0.9160	0.7976
Hypertriglyceridemia	0.9832	0.7457	0.9832	0.7457	0.9832	0.7457
Gallstones	0.9580	0.9726	0.9580	0.4642	0.9580	0.4681
OSA	0.9153	0.9067	0.9153	0.4275	0.9153	0.4165
Asthma	0.9746	0.9904	0.9746	0.6250	0.9746	0.6396
CAD	0.8632	0.7732	0.8632	0.5030	0.8632	0.5086
PVD	0.9832	0.9673	0.9832	0.9673	0.9832	0.9673
Gout	0.9832	0.9782	0.9832	0.6667	0.9832	0.6555
Diabetes	0.8571	0.7930	0.8571	0.5584	0.8571	0.5497
CHF	0.7542	0.8381	0.7542	0.5222	0.7542	0.5115
Venous Insufficiency	0.9916	0.9000	0.9916	0.9957	0.9916	0.9423
GERD	0.9748	0.9766	0.9748	0.6466	0.9748	0.6449
OA	0.8992	0.8265	0.8992	0.8265	0.8992	0.8265
Hypercholesterolemia	0.8983	0.9359	0.8983	0.7664	0.8983	0.8220
Hypertension	0.6441	0.3602	0.6441	0.3525	0.6441	0.3549
Textual	0.9083	0.7802	0.9083	0.4895	0.9083	0.5029

Table A.6: Scores on Named Entities only, minus family history sections. NEs are marked with negation.

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8559	0.8533	0.8559	0.8483	0.8559	0.8505
Depression	0.9217	0.8983	0.9217	0.8145	0.9217	0.8484
Hypertriglyceridemia	0.9739	0.9868	0.9739	0.6250	0.9739	0.6933
Gallstones	0.9915	0.9952	0.9915	0.9643	0.9915	0.9791
OSA	0.9224	0.5677	0.9224	0.5644	0.9224	0.5661
Asthma	0.9818	0.9896	0.9818	0.9375	0.9818	0.9614
CAD	0.9450	0.9592	0.9450	0.6414	0.9450	0.6323
PVD	0.9545	0.9728	0.9545	0.8913	0.9545	0.9251
Gout	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Diabetes	0.9730	0.9784	0.9730	0.9641	0.9730	0.9711
CHF	0.8447	0.8405	0.8447	0.8496	0.8447	0.8415
Venous Insufficiency	0.9524	0.9750	0.9524	0.7500	0.9524	0.8205
GERD	0.9327	0.9573	0.9327	0.8793	0.9327	0.9091
OA	0.9528	0.9728	0.9528	0.8684	0.9528	0.9103
Hypercholesterolemia	0.8900	0.8901	0.8900	0.8890	0.8900	0.8895
Hypertension	0.9065	0.8848	0.9065	0.8361	0.9065	0.8570
Intuitive	0.9384	0.6265	0.9384	0.6137	0.9384	0.6196

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8992	0.9015	0.8992	0.8898	0.8992	0.8945
Depression	0.9412	0.8512	0.9412	0.9093	0.9412	0.8769
Hypertriglyceridemia	0.9832	0.9916	0.9832	0.5000	0.9832	0.4958
Gallstones	0.9748	0.9929	0.9748	0.4833	0.9748	0.4878
OSA	0.9068	0.6619	0.9068	0.4119	0.9068	0.4119
Asthma	0.9831	0.9935	0.9831	0.6458	0.9831	0.6526
CAD	0.9145	0.9189	0.9145	0.7122	0.9145	0.6884
PVD	0.9664	0.9542	0.9664	0.9117	0.9664	0.9314
Gout	0.9832	0.9782	0.9832	0.6667	0.9832	0.6555
Diabetes	0.8908	0.9419	0.8908	0.5586	0.8908	0.6034
CHF	0.8644	0.9126	0.8644	0.5951	0.8644	0.5860
Venous Insufficiency	0.9832	0.9915	0.9832	0.7500	0.9832	0.8290
GERD	0.9664	0.6633	0.9664	0.6167	0.9664	0.6379
OA	0.9412	0.9197	0.9412	0.8707	0.9412	0.8927
Hypercholesterolemia	0.9407	0.7997	0.9407	0.7985	0.9407	0.7991
Hypertension	0.9407	0.8023	0.9407	0.7811	0.9407	0.7908
Textual	0.9425	0.6488	0.9425	0.5808	0.9425	0.6063

Table A.7: Scores on NEs marked as present and negated, feature-values are frequency count. Family history sections are ignored.

Intuitive judgements

Disease	NE-only	+SEG	-ALL	-FAM	NEG-FAM	FREQ
Obesity	0.8131	0.7740	0.7468	0.8131	**0.8682**	0.8505
Depression	0.8821	0.8353	0.8619	0.8821	**0.9195**	0.8484
Hypertriglyceridemia	0.7109	0.6658	0.6316	**0.7410**	**0.7410**	0.6933
Gallstones	0.9594	0.8229	0.8433	0.9594	**0.9791**	**0.9791**
OSA	**0.5716**	0.5480	0.4949	**0.5716**	0.5534	0.5661
Asthma	0.9228	0.7330	0.8255	0.9404	0.9404	**0.9614**
CAD	0.5994	0.5976	0.5922	0.6061	0.6129	**0.6323**
PVD	**0.9411**	0.8778	0.8580	**0.9411**	0.9251	0.9251
Gout	**1.0000**	0.9813	0.9788	**1.0000**	**1.0000**	**1.0000**
Diabetes	0.9456	0.4557	0.9252	0.9446	0.9679	**0.9711**
CHF	0.7343	0.6646	0.7864	0.7253	0.7314	**0.8415**
Venous Insufficiency	0.7120	0.7396	0.6617	0.7120	0.7294	**0.8205**
GERD	0.8947	0.7938	0.7221	0.8684	0.8462	**0.9091**
OA	0.8397	0.6235	0.7433	0.8397	0.8118	**0.9103**
Hypercholesterolemia	0.8286	0.7298	0.7600	0.8286	0.8399	**0.8895**
Hypertension	0.5325	0.4780	0.5522	0.5325	0.5223	**0.8570**
Intuitive	0.5947	0.5380	0.5711	0.5946	0.5984	**0.6196**

Textual judgements

Disease	NE-only	+SEG	-ALL	-FAM	NEG-FAM	FREQ
Obesity	0.8853	0.8017	0.8261	0.8769	**0.9297**	0.8945
Depression	0.8273	0.8626	0.4946	0.8538	0.7976	**0.8769**
Hypertriglyceridemia	**0.8312**	0.4936	0.4958	**0.8312**	0.7457	0.4958
Gallstones	0.4681	0.4368	0.3963	0.4681	0.4681	**0.4878**
OSA	**0.4301**	0.3832	0.3877	**0.4301**	0.4165	0.4119
Asthma	0.6311	0.5395	0.5674	0.6396	0.6396	**0.6526**
CAD	0.4306	0.4401	0.5407	0.4460	0.5086	**0.6884**
PVD	**0.9673**	0.9345	0.8240	**0.9673**	**0.9673**	0.9314
Gout	**0.6555**	**0.6555**	0.5845	**0.6555**	**0.6555**	**0.6555**
Diabetes	0.4325	0.1999	0.4348	0.4195	0.5497	**0.6034**
CHF	0.5137	0.6528	0.4940	0.5249	0.5115	**0.5860**
Venous Insufficiency	0.8570	1.0000	0.8956	0.8570	**0.9423**	0.8290
GERD	**0.6449**	0.6343	0.5851	**0.6449**	**0.6449**	0.6379
OA	0.8798	0.6780	0.6569	0.8155	0.8265	**0.8927**
Hypercholesterolemia	0.6059	0.6051	0.4736	0.6059	**0.8220**	0.7991
Hypertension	0.3705	0.3605	0.3538	0.3748	0.3549	**0.7908**
Textual	0.4500	0.4432	0.4584	0.4499	0.5029	0.6063

Table A.8: F-macro evaluations on the held-out data set. A comparison of the different classifiers across all 16 morbidities.

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.8881	0.8863	0.8881	0.8852	0.8881	0.8857
Depression	0.9266	0.9154	0.9266	0.8641	0.9266	0.8863
Hypertriglyceridemia	0.9753	0.8962	0.9753	0.8357	0.9753	0.8631
Gallstones	0.9715	0.9660	0.9715	0.9276	0.9715	0.9455
OSA	0.9596	0.6158	0.9596	0.6056	0.9596	0.6106
Asthma	0.9788	0.9570	0.9788	0.9570	0.9788	0.9570
CAD	0.9301	0.9503	0.9301	0.6229	0.9301	0.6196
PVD	0.9591	0.9518	0.9591	0.6044	0.9591	0.6112
Gout	0.9920	0.9692	0.9920	0.9954	0.9920	0.9818
Diabetes	0.9457	0.9360	0.9457	0.9360	0.9457	0.9360
CHF	0.8242	0.8865	0.8242	0.5522	0.8242	0.5509
Venous Insufficiency	0.9532	0.8628	0.9532	0.7191	0.9532	0.7702
GERD	0.9085	0.9234	0.9085	0.5624	0.9085	0.5743
OA	0.8853	0.8832	0.8853	0.5478	0.8853	0.5488
Hypercholesterolemia	0.8445	0.8418	0.8445	0.8448	0.8445	0.8429
Hypertension	0.7578	0.4836	0.7578	0.4935	0.7578	0.4726
Intuitive	0.9208	0.6065	0.9208	0.6042	0.9208	0.6053

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.9047	0.7058	0.9047	0.4546	0.9047	0.4549
Depression	0.8696	0.7318	0.8696	0.6691	0.8696	0.6924
Hypertriglyceridemia	0.9822	0.7797	0.9822	0.6970	0.9822	0.7308
Gallstones	0.9744	0.9899	0.9744	0.6245	0.9744	0.6391
OSA	0.9682	0.6330	0.9682	0.6153	0.9682	0.6237
Asthma	0.9782	0.7194	0.9782	0.4898	0.9782	0.4790
CAD	0.8893	0.5596	0.8893	0.5101	0.8893	0.5205
PVD	0.9684	0.9579	0.9684	0.8951	0.9684	0.9233
Gout	0.9842	0.9600	0.9842	0.9656	0.9842	0.9628
Diabetes	0.8807	0.6423	0.8807	0.6683	0.8807	0.6546
CHF	0.8145	0.6291	0.8145	0.5780	0.8145	0.5905
Venous Insufficiency	0.9921	0.8571	0.9921	0.9960	0.9921	0.9146
GERD	0.9821	0.9801	0.9821	0.4868	0.9821	0.4834
OA	0.8964	0.8347	0.8964	0.7761	0.8964	0.8006
Hypercholesterolemia	0.8785	0.8586	0.8785	0.5256	0.8785	0.5524
Hypertension	0.6846	0.6816	0.6846	0.3467	0.6846	0.3423
Textual	0.9158	0.5958	0.9158	0.5178	0.9158	0.5433

Table A.9: Scores on final configuration on test-set using nominal feature values.

Intuitive judgement

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.9217	0.9320	0.9217	0.9146	0.9217	0.9216
Depression	0.9476	0.9393	0.9476	0.9049	0.9476	0.9206
Hypertriglyceridemia	0.9424	0.5751	0.9424	0.5157	0.9424	0.5185
Gallstones	0.9817	0.9893	0.9817	0.9438	0.9817	0.9648
OSA	0.9414	0.6019	0.9414	0.5815	0.9414	0.5913
Asthma	0.9703	0.9616	0.9703	0.9154	0.9703	0.9367
CAD	0.9367	0.9551	0.9367	0.6266	0.9367	0.6240
PVD	0.9484	0.9484	0.9484	0.5745	0.9484	0.5924
Gout	0.9660	0.9470	0.9660	0.8889	0.9660	0.9152
Diabetes	0.9415	0.9273	0.9415	0.9368	0.9415	0.9318
CHF	0.8219	0.8814	0.8219	0.5538	0.8219	0.5504
Venous Insufficiency	0.9555	0.9366	0.9555	0.6884	0.9555	0.7566
GERD	0.9225	0.9471	0.9225	0.5684	0.9225	0.5868
OA	0.8961	0.9021	0.8961	0.5496	0.8961	0.5580
Hypercholesterolemia	0.8817	0.8793	0.8817	0.8836	0.8817	0.8806
Hypertension	0.9260	0.8988	0.9260	0.8596	0.9260	0.8773
Intuitive	0.9324	0.6195	0.9324	0.6097	0.9324	0.6142

Textual judgement

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.9087	0.7087	0.9087	0.4571	0.9087	0.4576
Depression	0.9585	0.9255	0.9585	0.9005	0.9585	0.9124
Hypertriglyceridemia	0.9783	0.6577	0.9783	0.5480	0.9783	0.5714
Gallstones	0.9744	0.9864	0.9744	0.6276	0.9744	0.6393
OSA	0.9443	0.6273	0.9443	0.5532	0.9443	0.5834
Asthma	0.9722	0.9684	0.9722	0.4787	0.9722	0.4734
CAD	0.9175	0.6417	0.9175	0.6154	0.9175	0.6268
PVD	0.9724	0.9847	0.9724	0.8906	0.9724	0.9308
Gout	0.9960	0.9815	0.9960	0.9978	0.9960	0.9895
Diabetes	0.9066	0.9186	0.9066	0.6442	0.9066	0.6542
CHF	0.8649	0.7425	0.8649	0.6169	0.8649	0.6325
Venous Insufficiency	0.9980	0.9990	0.9980	0.9500	0.9980	0.9732
GERD	0.9742	0.7317	0.9742	0.4693	0.9742	0.4753
OA	0.9343	0.9009	0.9343	0.8589	0.9343	0.8779
Hypercholesterolemia	0.9044	0.8691	0.9044	0.5419	0.9044	0.5666
Hypertension	0.9760	0.9836	0.9760	0.8123	0.9760	0.8700
Textual	0.9490	0.6519	0.9490	0.5766	0.9490	0.6052

Table A.10: Scores on final configuration on test-set using frequency feature values.

Intuitive judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.9217	0.9265	0.9217	0.9146	0.9217	0.9191
Depression	0.9476	0.9393	0.9476	0.9049	0.9476	0.9206
Hypertriglyceridemia	0.9424	0.5751	0.9424	0.5157	0.9424	0.5185
Gallstones	0.9817	0.9893	0.9817	0.9438	0.9817	0.9648
OSA	0.9495	0.8953	0.9495	0.8904	0.9495	0.8928
Asthma	0.9703	0.9616	0.9703	0.9154	0.9703	0.9367
CAD	0.9389	0.9345	0.9389	0.9400	0.9389	0.9369
PVD	0.9484	0.9225	0.9484	0.8561	0.9484	0.8852
Gout	0.9660	0.9470	0.9660	0.8889	0.9660	0.9152
Diabetes	0.9415	0.9273	0.9415	0.9368	0.9415	0.9318
CHF	0.8242	0.8243	0.8242	0.8250	0.8242	0.8241
Venous Insufficiency	0.9555	0.9366	0.9555	0.6884	0.9555	0.7566
GERD	0.9249	0.9275	0.9249	0.8503	0.9249	0.8814
OA	0.8983	0.8594	0.8983	0.8150	0.8983	0.8343
Hypercholesterolemia	0.8817	0.8793	0.8817	0.8836	0.8817	0.8806
Hypertension	0.9260	0.8988	0.9260	0.8596	0.9260	0.8773
Intuitive	0.9335	0.9294	0.9335	0.9137	0.9335	0.9210

Textual judgment

Disease	P-Micro	P-Macro	R-Micro	R-Macro	F-Micro	F-Macro
Obesity	0.9209	0.9228	0.9209	0.9131	0.9209	0.9172
Depression	0.9585	0.9255	0.9585	0.9005	0.9585	0.9124
Hypertriglyceridemia	0.9783	0.6577	0.9783	0.5480	0.9783	0.5714
Gallstones	0.9783	0.9819	0.9783	0.9413	0.9783	0.9602
OSA	0.9602	0.9546	0.9602	0.8768	0.9602	0.9105
Asthma	0.9782	0.9520	0.9782	0.9574	0.9782	0.9546
CAD	0.9437	0.9421	0.9437	0.9438	0.9437	0.9429
PVD	0.9724	0.9847	0.9724	0.8906	0.9724	0.9308
Gout	0.9960	0.9815	0.9960	0.9978	0.9960	0.9895
Diabetes	0.9165	0.8996	0.9165	0.9141	0.9165	0.9062
CHF	0.8790	0.8747	0.8790	0.8767	0.8790	0.8756
Venous Insufficiency	0.9980	0.9990	0.9980	0.9500	0.9980	0.9732
GERD	0.9861	0.9737	0.9861	0.9680	0.9861	0.9708
OA	0.9343	0.9009	0.9343	0.8589	0.9343	0.8779
Hypercholesterolemia	0.9203	0.9192	0.9203	0.9183	0.9203	0.9187
Hypertension	0.9780	0.9799	0.9780	0.9619	0.9780	0.9704
Textual	0.9564	0.9499	0.9564	0.9397	0.9564	0.9446

Table A.11: Final scores using frequency feature values, ignoring low-frequent classes.

Appendix B

Access to unstructured information

Since the importance of using clinical data when developing IE systems has become so obvious during our work, we include a discussion of this here.

B.1 De-identification of documents

A necessary resource when developing information extraction systems for the clinical domain is training and evaluation material, i.e. clinical records. But if one wants to redistribute sensitive clinical information de-identification is a prerequisite. This is an expensive task, since the full de-identification of a document requires a number of stated facts (private health information, PHI) to be altered or deleted, according to official privacy policy. We have already talked about some of these issues covering the i2b2 2007 de-identification task (page 56). Uzuner et al. (2007) summarise the following requirements from the Administrative Simplification Regulations (paragraph 164.514) that is to be met if the data is to be considered as de-identified (the quote within the quote is from the paragraph):

> 1 An expert must determine and document "that the risk is very small that the information could be used, alone or in combination with other reasonably available information, by an anticipated recipient to identify an individual who is a subject of the information."
>
> 2 Or, the data must be purged of a specified list of seventeen categories of possible identifiers relating to the patient or relatives, household members and employers, and any other information that may make it possible to identify the individual. Many institutions consider the clinicians caring for a patient and the names of hospitals, clinics, and wards to fall into this final category because of

the heightened risk of identifying patients from such information. (Uzuner et al., 2007, p. 550)

There has been attempts at building systems for automatic de-identification, as explored in the i2b2 de-identification challenge (Uzuner et al., 2007). But in order to make reliable de-identification systems one would need training and evaluation material, i.e. clinical records, but such records cannot be released without prior de-identification. In Scandinavia there has been some attempts at de-identifying clinical records, for instance: Tveit et al. (2004) worked on Norwegian general practitioner records. In Sweden they annotated clinical records taken from the Stockholm EPR Corpus for anonymization, as described by Velupillai (2012). In Denmark Pantazos et al. (2011) created a database of 323,122 de-identified patient records.

If one is to release de-identified clinical records one should make sure of two things: (1) One should be able to discover the cases where two different records are referring to the same person, as this could provide useful insight into, for instance, historical development of a patient. So if a particular name, say "Mary Doe" is replaced with a surrogate name "Jane Smith", all occurrences of "Mary Doe" in other records should be replaced with the same name. Other names should *not* be replaced with "Jane Smith". The same principle goes for social security numbers, etc.

(2) All replacements should be annotated in a concise manner so that the material can be used when training and/or evaluating systems for automatic de-identification. Since the task of de-identification is so expensive in itself, making an effort to add such valuable meta-data to the documents should be considered. PHI fields in the document could for instance be marked with text-span, type (name, social security number, etc.) and role (patient, doctor, family-member, facility, etc.). We imagine that a project like this would benefit from a graphical interface, where annotators could just select the relevant text and pick the type and role from drop-down lists. A computer program could then automatically insert surrogate names and numbers in accordance with (1), and output the result with the de-identified annotated text stored in a computable friendly fashion.

B.2 In-house Material

Instead of releasing clinical records to a wider research audience, one could instead allow access to the EMR/EHR belonging to an institution to in-house researchers/developers. Two notable examples of such collections used in NLP/IE research are the Mayo Clinic EMR (Savova et al., 2010) and the Stockholm EPR Corpus (Dalianis et al., 2009). The use of records unavailable to a wider research group has it downside, however: It is difficult, if not impossible, for other scientists to replicate experiments if they want to check if the original research was in error or investigate possible improvements.

When making a new IE system, it would for instance be interesting to see if there is any statistical significant difference between the new system and the previous attempts. But to fully investigate this requires access to the same system (in the best case), or at least a way of building an equivalent system. Building an equivalent system is impossible if the other systems are trained on unobtainable material.

It is also in some cases difficult to improve a module in a pipeline, while keeping the rest of the modules, without access to the same material that the other modules are trained on. This is because of the dependency between the modules; the output of one module in an IE/NLP pipeline is usually the input of the next. So even if one builds, say, a better POS-tager, it is hard to evaluate the over-all system performance because (for instance) the chunker is trained on the output of the old tagger which might make the performance drop even if the new module is performing better. If the material that all modules were trained on is available, one have the ability to make one of the modules better and train the rest of the modules with the output from the modified module together with the annotated material. This situation can be avoided by training and evaluating all modules on a gold standard.

Appendix C

Detecting Document Section Types

The following strings were used to detect the different section types in the discharge summaries. If the headline contained on of the sub-strings in the right column, the whole section would be classified as the type in the left column on corresponding row.

Allergies	“ALLERG”
Comments	“SERIOUS INTERACTION”, “DISPOSITION”, “DIET”, “PRESENT ILL”, “HOSPITAL COURSE”, “DISCHARGE CONDITION”, “OPERATIONS AND PROCEDURES”, “PLAN”, “COMMENTS”, “BRIEF RESUME OF HOSPITAL COURSE”, “SUMMARY”, “DISCHARGE PATIENT ON”, “EXAMINATION”, “DISCHARGE”, “COMPLA”, “PROCED”, “FOLLOW”, “IMPRESSI”
Diagnosis	“DIAGN”
Family history	“SOCIAL HISTORY”, “FAMILY HISTORY”
Laboratory data	“LABORA”, “LABS”
Medications	“MEDICAT”
Other	
Past	“PAST”

Table C.1: Sub-strings of headlines for detecting the section type.