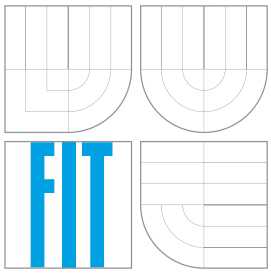
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

EXTRAKCE INFORMACÍ Z BIOMEDICÍNSKÝCH TEXTŮ

INFORMATION EXTRACTION FROM BIOMEDICAL TEXTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — PETR KNOTH
AUTHOR

VEDOUCÍ PRÁCE — Doc. RNDr. PAVEL SMRŽ, Ph.D.
SUPERVISOR

BRNO 2008

Abstrakt

V poslední době bylo vynaloženo velké úsilí k tomu, aby byly biomedicínské znalosti, typicky uložené v podobě vědeckých článků, snadněji přístupné a bylo možné je efektivně sdílet. Ve skutečnosti ale nestrukturovaná podstata těchto textů způsobuje velké obtíže při použití technik pro získávání a vyvozování znalostí. Anotování entit nesoucích jistou sémantickou informaci v textu je prvním krokem k vytvoření znalosti analyzovatelné počítačem. V této práci nejdříve studujeme metody pro automatickou extrakci informací z textů přirozeného jazyka. Dále zhodnotíme hlavní výhody a nevýhody současných systémů pro extrakci informací a na základě těchto znalostí se rozhodneme přijmout přístup strojového učení pro automatické získávání exktrakčních vzorů při našich experimentech. Bohužel, techniky strojového učení často vyžadují obrovské množství trénovacích dat, která může být velmi pracné získat. Abychom dokázali čelit tomuto nepříjemnému problému, prozkoumáme koncept tzv. bootstrapping techniky. Nakonec ukážeme, že během našich experimentů metody strojového učení pracovaly dostatečně dobře a dokonce podstatně lépe než základní metody. Navíc v úloze využívající techniky bootstrapping se podařilo významně snížit množství dat potřebných pro trénování extrakčního systému.

Klíčová slova

extrakce informací, strojové učení, zpracování přirozeného jazyka

Abstract

Recently, there has been much effort in making biomedical knowledge, typically stored in scientific articles, more accessible and interoperable. As a matter of fact, the unstructured nature of such texts makes it difficult to apply knowledge discovery and inference techniques. Annotating information units with semantic information in these texts is the first step to make the knowledge machine-analyzable. In this work, we first study methods for automatic information extraction from natural language text. Then we discuss the main benefits and disadvantages of the state-of-art information extraction systems and, as a result of this, we adopt a machine learning approach to automatically learn extraction patterns in our experiments. Unfortunately, machine learning techniques often require a huge amount of training data, which can be sometimes laborious to gather. In order to face up to this tedious problem, we investigate the concept of weakly supervised or bootstrapping techniques. Finally, we show in our experiments that our machine learning methods performed reasonably well and significantly better than the baseline. Moreover, in the weakly supervised learning task we were able to substantially bring down the amount of labeled data needed for training of the extraction system.

Keywords

information extraction, machine learning, natural language processing

Citace

Petr Knoth: Information Extraction from Biomedical Texts, diplomová práce, Brno, FIT VUT v Brně, 2008

Information Extraction from Biomedical Texts

Declaration

I hereby declare that I am the sole author of this thesis and that, to the best of my knowledge and belief, it contains no material previously published or written by another person. Where other sources of information have been used, they have been acknowledged.

.....
Petr Knoth
May 12, 2008

Acknowledgements

I would like to express my sincere thanks to my supervisor Pavel Smrž, who guided me throughout the study and who was always prepared to offer constructive technical and organizational help. I would also like to thank Mark Craven (University of Wisconsin) for his willingness to provide me with his annotated data set. Many thanks belongs also to Marco Wiering (University of Groningen) who is responsible for making me passionate about natural language processing and machine learning. Last but not least, I would like to thank my family and all other people who supported me during my studies.

Contents

Chapter 1

Introduction

In recent years the amount of unstructured data stored on the Internet and other digital sources has increased significantly. This data, however, contains often valuable, but hardly retrievable information. The term *unstructured data* refers mainly to data that does not have a data structure. As a result of this, the unstructured data is not easily readable by machines.

Information extraction (IE) is usually defined as the process of selectively structuring and combining data that are explicitly stated or implied in one or more documents. This process involves a semantic classification of certain pieces of information and is considered as a light form of text understanding [36]. The structured information can be then in turn used as a basis for question answering, machine translation, semantic web systems etc. Currently, there is a considerable interest in using these systems for information retrieval. This is caused by an increasing need to localize precise information, rather than just retrieving a list of the most relevant documents.

Consider for example a biomedical scientist, who is developing a new treatment for a certain disease. He wants to know what can be the side effects of the substances that his treatment is composed of. Since there is an overwhelming number of scientific studies and research articles that should all be taken into account, it would be great to make a conclusion from all the knowledge provided there. This example shows that although there is a strong need for structured information, it is hardly possible for a human to process all data manually, because there is simply too much of it. Unfortunately, machines are also not able to directly query for the target information, because it is not stored in a structured format. *Information extraction (IE)* is the subdiscipline of artificial intelligence that tries to solve these kinds of problems [36].

The problem of transforming unstructured information to structured information can be solved by assigning special tags, often called *annotations*, to certain pieces of the unstructured data. In information extraction, annotations generally provide *metadata* that describe the content of individual entities in a text. Most approaches in information extraction in the past relied on *extraction rules* that served as a key to identify which entities to annotate and which type of annotation to assign.

It should be noted that *information extraction* systems should be preferably *domain independent*, or at least easily portable across different domains. In practise, many information extraction systems lack this feature mainly because of technological limitations arising, for example, with hand-coded extraction rules. In this work we are therefore mostly concerned with techniques that are capable of working *semi- or fully automatically.*

It is evident that current *IE* technology would allow rapid creation of extraction systems

for new tasks whose performance would approach a human level. Nevertheless, even systems without near perfect precision can be of real value [15]. In these cases it can be often important to provide an interface where the user can semi-automatically validate the results given by the system.

Although we are in this work concentrating on *information extraction* from plain natural language text, *information extraction* as a discipline is also concerned with other multimedia sources such as image or video. The ability to extract names of organizations, people, locations, dates and times is essential for almost all information extraction systems. However, current approaches applied to video streams have significant shortcomings. Most methods are either rule-based, or require huge amounts of manually labeled training data to achieve a reasonable level of performance. The methods may identify a name, company, or location, but this is only a small part of the information that should be extracted; we would like to know that a particular person is a politician and that a location is a vacation resort [58]. Besides that much work currently concentrates on information extraction from Web pages. This source of information is regarded as *semi-structured* text. Information extraction techniques for plain text are not best suited for online documents where also visual aspects and a logical structure of the documents can be taken into account in contrast to plain text. A comprehensive overview on how to handle such documents can be found in [7].

In this work we study information extraction techniques with a special interest in state-of-the-art machine learning approaches, which allow us to build fully automatic and relatively domain independent systems. An important part of this work lies in the area of related work. We try to always present how the proposed techniques and mathematical models can be put into practice and we discuss their benefits and disadvantages. Acquired with this knowledge, we decide to focus in our experiments mainly on the task of semantic relation extraction. For a start, we implement and evaluate an extraction system in a general domain. Similar, but more advanced techniques are then used on two tasks in the biomedical field, namely detection and classification of semantic relations. To motivate these tasks, we provide here a brief introduction.

Introduction to Case Studies

One of the huge sources of human knowledge is Wikipedia. Wikipedia is a multilingual, open-access, free[1] content encyclopedia project operated by the non-profit Wikimedia Foundation [59]. Although Wikipedia can often serve as a great source of information, the stored knowledge may be difficult to locate or it is just a time consuming process. The problem is that Wikipedia's search capabilities are limited to full-text search, which only allows very limited access to this knowledge base [5]. DBpedia is a project that face up to this precise query problem using semantic web techniques.

In chapter 6 we present and evaluate a simple method for automatic extraction of capital city entities from Wikipedia articles based on pattern matching. The motivation for development of such system could be, for example, an automatic or semi-automatic extraction of infobox tables of figure 1.1 from Wikipedia articles. The knowledge stored in these structured tables can be in turn used to improve the search capabilities of Wikipedia or only to provide the user with well-arranged information.

Among the application domains of information extraction, the biomedical domain is currently the most important [36]. There have been many attempts to extract information

[1]Some language versions such as English one contains non-free images.

Figure 1.1: Wikipedia infobox template for Czech Republic

from *patient reports* and use it in order to obtain decision support systems, patient management systems or provide a support for clinical research. Another sources of valuable medical information are biomedical databases. The serious necessity to access this information efficiently is caused mainly by the large amount of biological and medical literature that is growing exponentially. Figure 1.2 shows the growth of *MIM* and *OMIM* database in terms of numbers of entries in each edition. New results and publications are appearing every day in research journals. Many of these publications are available online, for example, in the *MEDLINE* database.

The MEDLINE database can be accessed online using *PubMed* service, which is available via the *Entrez retrieval system* developed by the *National Center for Biotechnology (NCBI)*. Entrez is a text-based search and retrieval system used at *NCBI* for services, such as PubMed, OMIM, *Protein Sequences* and many others. Although MEDLINE contains nearly 11 million records from more than 7,300 different publications dating from 1965 until present, its search capabilities are unfortunately still very limited.

Therefore, in chapter 7, we investigate the current drawbacks of these systems and present our experiments to automatically detect semantic relations between *proteins and subcellular locations*. Our experiments will use state-of-the-art machine learning techniques in order to become entirely *domain independent*. In the second part of that chapter, we argue that only detection of semantic relations is often not sufficient. As a result, we experiment with assigning 10 different semantic labels to *protein-protein* interactions. Finally, we tackle the problem of minimizing the amount of training data needed to train our models.

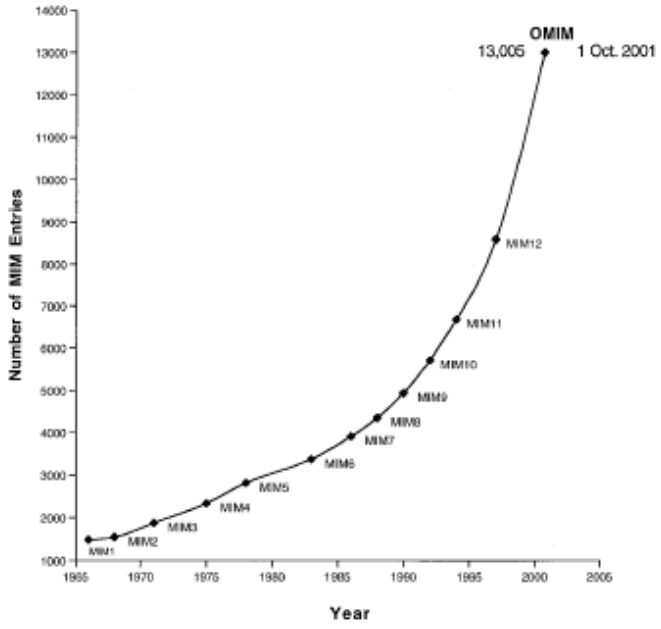


Figure 1.2: Growth of the *OMIM* and *MIM* database in terms of entries in each edition [17]

Chapter 2

The Information Extraction Process

In this part, the main components of a typical information extraction system are described and it is shown how these components cooperate. Later, possible output formats of information extraction systems are discussed.

The information extraction system architecture in figure 2.1 has two distinct phases: The *training phase* and the *deployment phase*. In the training phase the system acquires extraction patterns that can be created manually or learned using machine learning techniques. A set of texts *(corpus)* selected in this stage, should be preferably similar to the domain the system is intended for. In case of using machine learning for automatic or semi-automatic learning of extraction patterns, the corpus is often required to contain annotated examples of entities and/or relations involved in the extraction task.

Before the texts can be used for extrapolating extraction rules, they usually go through a preprocessing phase (T2) in which their formal characteristics are normalized. Normalization comprises harmonizing spelling and capitalization and cleaning up unnecessary metadata. For some applications it can be useful to perform a couple of *natural language processing (NLP)* tasks. This involves simple tasks, such as creating stop list (excluding determiners), stemming (restoring words to a root form) or lemmatization (restoring words to a dictionary form). More complicated NLP tasks, which belong to the preprocessing phase, refer to the enrichment of textual data with linguistic metadata that will be used as parameters in the acquisition process [36]. This step involves processes such as part-of-speech (POS) tagging, chunking and phrase chunking (detecting sentence boundaries and detecting noun and verb phrases respectively), shallow and full parsing, semantic tagging etc. Fortunately, there exists a wide range of freely available tools for these kinds of tasks.

When the manual approach is applied to the generation of extraction patterns, a domain specialist uses the preprocessed corpus to induce the extraction grammar.[1] In this case, the extraction grammar can be seen as a set of hand-made rules. By contrast, when machine learning is involved in the acquisition learning part, the extraction grammar can be also represented as a mathematical function or a model that is able to predict the class of a given example. We will discuss these models in chapter 4.

In the 1990s, there was a growing concern in application of machine learning in information extraction. Most of these methods were based on supervised techniques to learn

[1] This process is in information extraction from semi-structured data sometimes called *wrapper construction*. *Wrapper* is a procedure that provides the extraction of particular data in a document [7].

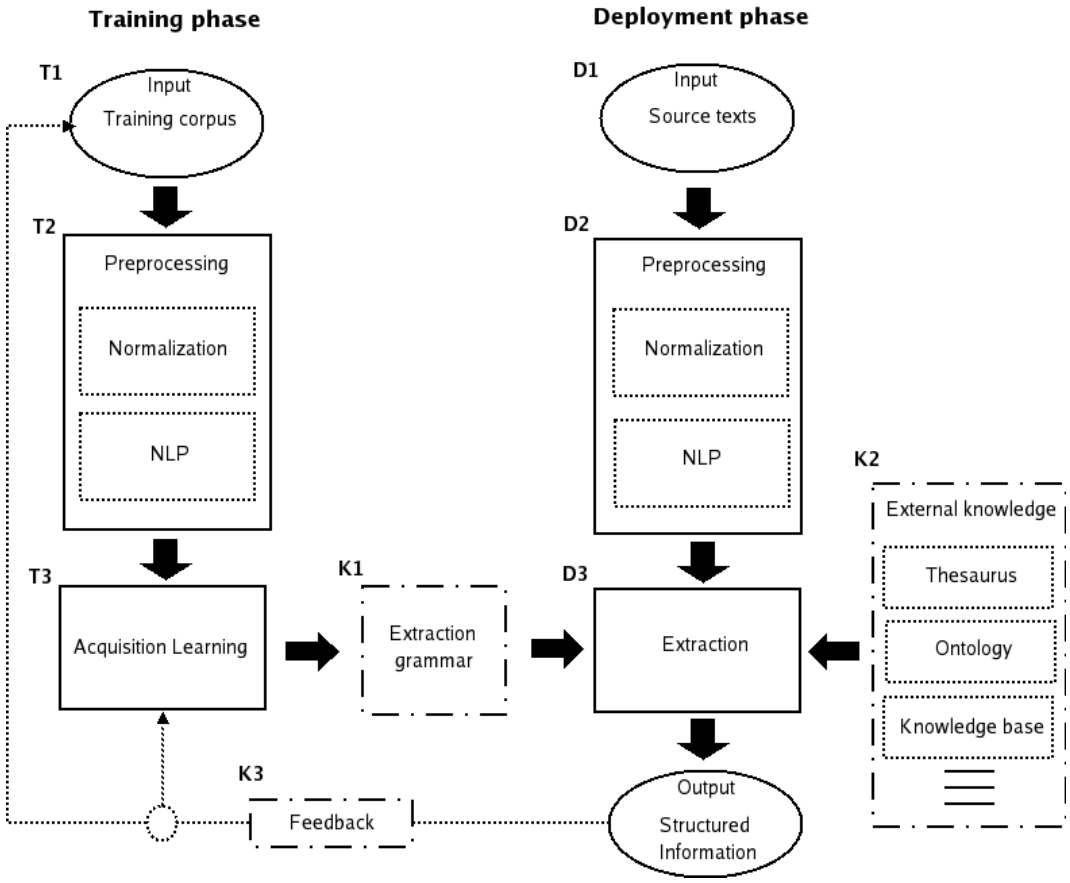


Figure 2.1: The architecture of a typical information extraction system according to [36]. **T** - Training phase component, **D** - Deployment phase component, **K** - Knowledge component

.

extraction patterns from plain, but also semi-structured text. In that time, several systems were developed for automatic generation of extraction patterns. One of such systems is AutoSlog [43], which uses a predefined set of 13 linguistic patterns. Other systems involve PALKA [18] or CRYSTAL that are based on manually constructed concept hierarchies.

In the last decade, it is becoming more and more popular to let the system start the acquisition process from a relatively small set of annotated examples *(seed)*. After that, the extraction grammar is applied on real texts in the extraction phase. The output is then used to provide feedback in the acquisition learning phase. The model is retrained and the whole process is iterated until the model achieves a reasonable performance. This technique is often denoted as *bootstrapping*. Well-known extraction systems that rely on bootstrapping are, for instance, Snowball [2] or AutoSlog-TS [44], which is based on a previous version of the AutoSlog system. Espresso [39] even combines bootstrapping approach with a Web-based knowledge expansion in order to exploit the seeds as much as possible.

As it was said, in the deployment phase the system identifies and classifies relevant semantic information in new texts, i.e., texts that were not included in the training corpus. The preprocessing component (D2) is as similar as possible to that in the training phase. After preprocessing, the input texts are passed on to the extraction component (D3).

Existing literature usually does not focus on the real world implementation of information extraction, but rather on development and testing. As a consequence, the deployment phase is often called the evaluation or testing phase [36]. In this phase, the system uses the extraction grammar (K1) produced by the acquisition learning component (T3) and possibly some other knowledge sources (K2) that can improve the quality of the extraction process.

Among the knowledge components of (K2) let us highlight the ontology, which is currently argued to be able to significantly improve the quality of information extraction [24]. In addition, the extracted information can be used to extend the ontology with new facts and to keep it up-to-date. Within the last 30 years, the field of information extraction gained a lot of importance mainly fostered by the Message Understanding Conferences (MUCs), which provided a methodology for researchers to present and evaluate their work. They started with the fundamental tasks, such as *named entity recognition* or *coreference resolution* going to more complicated tasks referring to *scenario extraction* or *relation extraction*. The structure of the output was predefined by a template, which specified the attributes that could be extracted for single events or more complex scenarios. The goal of the information extraction system was to fill the template *slots* from the source text as precisely as possible. In [30] they argue that nowadays are available efficient inference systems, which can work on the ontological structure, are available and therefore the *ontology-based* description of the domain should be favoured over the comparatively rigid *template-based* specifications of the output.

The template-based systems are called *document-driven* systems, because their workflow follows the documents. On the contrary, in [33] it is proposed an alternative *ontology-driven* information extraction system that instead of searching the documents sequentially processes the ontology in some order. They seek to use ontological data and structure to enhance the assessment of the content that is found. They claim that search based on this approach enables them to consider much larger set of documents than could be handled via individual *document-driven* processing. Another example of using ontology to improve the quality of information extraction is described in [3], where an approach to learn information extraction patterns from natural language text using Inductive Logic Programming (ILP) is presented. In this approach, ontology is viewed only as a knowledge artifact that represents the conceptualizations and on which a human annotator can lean on while making annotations on a corpus. For more details on related work in this field see [65].

Information extraction can be also used to populate the ontology with entities that are extracted from texts. Note that ontologies in general encode conceptualizations that are not bound to specific text and apply in general. To the contrary, information extraction systems are very well-suited to find instances of concepts rather than concepts themselves and therefore they can be better used for populating ontologies than for constructing them [6]. There are two main classical approaches to ontology population. The first one relies on a use of patterns (e.g. Hearst-patterns) [20]. In the second approach, the task is addressed using contextual features [10]. Pattern-based approaches search for phrases that explicitly show that there is an *"is-a"* or *"part-of"* relation between two words. Unfortunately, these phrases do not appear too often in a corpus. On the other hand, context feature-based approaches use a corpus to explore the context in which a semantic class tends to appear. Fortunately, but not surprisingly, both approaches can be combined.

Generally, the state-of-the-art ontology population methods can be divided according to a different use of training data to unsupervised methods, which usually have low perfor-

mance, and supervised machine learning methods that reach higher accuracy, but require manual annotations of the training set [55]. For example, in [16] the latter method is used to create a named entity tagger that is able to classify person instances to different subcategories. These persons can be then used to populate the ontology nodes corresponding to politicians, entertainers etc.

Chapter 3

Patterns

3.1 Pattern Recognition

In this thesis we are concerned with plain natural language texts. Our assumption is that natural language texts are not completely irregular and that is why we are able to identify common patterns, which can serve as a first step for retrieving the semantics of a language.

This is a consequence of the principle of compositionality and states that a meaning of any complex linguistic expression is a function of the meanings of its constituent parts[1] [36]. If a natural language would be entirely random sequence of words, it would be impossible for humans to make any sense of it. We suppose that the meaning of a sentence is determined by lexical elements (words and word groups), grammatical constructions (phrases, sentences etc.) and the pragmatic ordering structure (paragraphs, headlines etc.).

Information extraction is often used for semantic extraction of entities, which are known in advance. For instance, it might be important to extract all places of conferences referring to NLP tasks from a certain text. Consider for a moment that we have all possible contexts of demanded entities stored in a database. We refer to these contexts as *patterns*. Then, given an unknown text we can compare each sentence of the unknown text with all patterns in our database to find out whether the information we are looking for occurs in the unknown text. Given the assumption that we have all possible patterns we are likely to identify most of the demanded entities.

However, in reality this task gets indeed much more complicated. As a matter of fact, natural language is highly ambiguous. The ambiguity is not only present at the word sense level, we refer the reader to [31] for more details, but also the meaning of a whole sentence can be ambiguous [27]. Although quite powerful unsupervised algorithms have been developed for word-sense disambiguation such as [63], based on one sense per discourse and one sense per collocation, we still have to deal with other sources of uncertainty. Firstly, it is possible to write a sentence in large numbers of different ways. Secondly, the probability of seeing given entity in the context[2] of the whole document is extremely low. That is why we usually cannot have all contexts (patterns), while searching for some entity. As a result of this phenomenon, we are interested in creating such patterns that are general enough to cover various cases of appearance of a given entity, while sufficiently selective to minimize the error of the entity misclassification. This is often denoted as a trade-off between *recall*

[1] In fact, this is a hard problem because natural language often does not obey this principle. For example white hair is grey, white skin has rosy color etc. The meaning of the whole is the sum of the parts plus some additional meaning that cannot be predicted from the parts [31].

[2] The context is here represented by all words in the whole article appearing around the desired entity.

and *precision* and we will discuss it later on.

3.2 Pattern Representation

As it was learnt in the foregoing parts, information extraction relies on pattern recognition methods. Pattern recognition (also known as classification or pattern classification) aims at classifying data (patterns) based on either a priori knowledge that is acquired by human experts or on knowledge automatically learned from data. A system, that automatically sorts patterns into classes or categories is called a *pattern classifier* [36]. The classification patterns consists of features and their values. In our case, the features are textual characteristics that can be identified or measured, and that are assumed to have a discriminative value.

In this section we discuss what are the information units and their relations we want to extract and how these units can be described with feature vector or other object that captures the necessary feature values for correct classification.

Features

Single words, base phrases, clauses, sentences, passages etc. might all be considered as information units to extract. However, we are mostly concerned with information extraction of *base phrases* (base noun and verb phrases).

Machine learning and statistical approaches rely mainly on feature vectors extracted directly from a document. It would not be feasible to use all features present in a text, so we select the important ones at hand. This selection should be carefully considered, since inappropriate feature vectors may cause inferior extraction results. All the information units we have identified in a document can be represented by their feature vector. Features can have numeric values e.g. discrete or real. A special type of discrete features are binary features. Features can also have nominal values (for example certain words). However, these features are often translated into a set of numeric values. A common demand for features is that they have to be sufficiently discriminative. Therefore, the features are frequently scaled, reduced or weighted. In weighting, we try to reflect the importance of a certain feature in a given context.

A typical choice of features in document classification is a *bag-of-words* approach assuming that words are independent of their position. In a number of experiments it has been found that more sophisticated representations do not yield significantly better results [48]. However, in information extraction we are usually taking into account only a close neighborhood or *context window* of an information unit and the position in the text can sometimes play an important role. The best size of the context window often varies according to the extraction task.

Generally, features can be according to their position in a text divided into:

- features occurring in the information unit itself
- features occurring in the context window
- features of units which are linked in a certain relation
- features occurring in the complete document

As it was stated above, the selection of relevant features is a difficult process and often depends on the extraction task. Now, we should try to answer the question what kinds of relevant features are stored in a text. In summary, from every natural language text lexical, syntactic, semantic and discourse features can be extracted.

Lexical Features

Lexical features refer to the attributes of lexical items (often words) of a text [36]. We assume that lexical context of a target word may restrict its semantics.

A common choice for lexical features in NLP are words. Each word can be represented as 0 or 1 in a feature vector (binary features), depending on an occurrence or nonoccurence of a word in a certain context. Words in a context window might receive a weight according to their importance. One of the classical weighting function is $tfidf$ *(term frequency inverse document frequency)*. The main idea of $tfidf$ is that the most semantically important words for a given document tend to occur frequently in the document, despite their possibly rare occurrence in the whole collection.

So far, we have assumed that members of our feature vector are individual words. This model is often called *unigram*. *Bigrams* are pairs of words[3] occurring in close proximity to each other, and in a particular order[37]. Commonly, some unigrams and bigrams are ignored by specifying a stop list composed mainly of prepositions, articles and conjunctions.

Note that feature vectors can have too many dimensions using the approach described above. As a result of this, *reduction* techniques are used to reduce the number of these dimensions. For example, entities can be referred to by their *synonym, hypernym, hyponym* or *meronym*. Thesauri or lexical databases such as WordNet [34] contain these term relationships. Another popular, but purely statistical method for dimensionality reduction is *Latent Semantic Analysis (LSA)*.

Syntactic Features

The most common syntactic feature used in information extraction is the *part-of-speech (POS)* of a word. There exist a number of part-of-speech taggers operating with a very high accuracy. For instance Natural Language Toolkit (NLTK) [29] provides several different implementations of POS taggers.

A number of syntactic features rely on parsing of the sentence structure. The grammatical role, such as subject, direct object and indirect object might play a role in the extraction process. Grammatical roles, which are sometimes also called syntactic roles, are detected with the help of rules applied on the parse tree of a sentence. Parse information is also important in detecting relations between entities [36].

Semantic Features

Semantic features refer to single or multi-word units that are classified into semantic classes. One of the most common semantic recognition task is *Named Entity Recognition* where we are interested in identifying entities such as person, organization, money etc. Semantic features can be either extracted using trained classifiers or using an external knowledge sources. Especially useful are semantic lexicons, which contain individual words with their possible semantic classes. There also exist *gazetteers* for geographical or other names.

[3]Bigrams can also be used for different types of features such as syntactic ones.

Unfortunately, semantic *lexica* often have to be completed with domain specific knowledge in order to be used in practical applications.

Discourse Features

Discourse features refer to features the values of which are computed by using text fragments, i.e., a discourse or a connected speech or writing larger than the sentence. Discourse features are used to a lesser extent, but will certainly become more important in future semantic classifications [36].

Chapter 4

Machine Learning for Information Extraction

This chapter gives an overview of the state-of-the-art methods and algorithms for automatic extraction and acquisition of valuable patterns from a text. The problematics of expensive data labeling is also dealed with. As a result of this, we are mostly interested in algorithms that are able to produce good results while starting only from a small set of hand provided examples.

Early information extraction systems relied on rules or patterns that were usually handcrafted. Currently, machine learning plays a central role in the information extraction paradigm [36]. In most cases, *supervised learning* is applied to automatically generate extraction grammar or extraction patterns that can be seen as a set of rules, mathematical function or only as a mathematical model. A predominant approach to addressing this challenge has been to annotate a large corpus with the information to be extracted, and then use a learning procedure to learn some characteristics from the annotated corpus [62]. Unfortunately, this approach usually requires a large tagged data set. Therefore, we discuss currently promising techniques, which take supervised methods from section 4.1 as a basis, for development of methods that are able to learn from an untagged text. We refer to them as *weakly supervised* techniques and we give a brief overview of them in section 4.2.

4.1 Supervised Techniques

Supervised techniques have been very successful in information extraction. One of their main benefits is that it is often much easier to provide a set of examples, which are in turn used by machine learning algorithms to generalize than writing extraction rules from scratch.

In chapter 3 we described that information units or entities to be classified can be represented by a feature vector. The feature vector usually represents the context in which the classified entity appears in a text, while a bigger weight is often assigned to a close context, rather than global context.

Let x be a feature vector. The goal of machine learning is to assign a label y to a new example. Among the statistical learning techniques a distinction is often made between generative and discriminative classifiers. Given inputs x and their labels y, a *generative classifier* learns a model of the joint probability, $p(x, y)$ and makes its predictions by using Bayes' rule to calculate $p(y|x)$ and then selects the most likely label y. Examples

of generative models are *naïve Bayes* and *hidden Markov model*. A *discriminative classifier* models the posterior probability $p(y|x)$ directly and selects the most likely label y or learns a direct map from inputs x to class labels [36]. Examples of discriminative models are *maximum entropy model* and *Support Vector Machine*.

Sometimes it is valuable not to classify a feature vector separately from other feature vectors, because a relation can exist between the various classes. Thus the class to which a feature vector is assigned does not depend only on the feature vector itself, but also on the values of other feature vectors and the relations that hold among the classes. *Context dependent* classifiers such as *hidden Markov model* can partly solve this problem. Finally, some techniques present the learned patterns in a human readable format. These techniques involve *rule and tree learning* or *relational learning*.

In information extraction we often face relatively complex problems, because many tasks contain more nontrivial parts. For example, *relation extraction* is a task consisting of the following subtasks: 1) detect the boundaries of the possible constituents 2) determine their class 3) detect if some relation holds between them and 4) classify the type of the relation between those constituents. We have the choice to deal with these problems separately or we can try to solve them at once. For instance, in [12] SVMs were used for points 3) and 4) of the example. Better results were experienced when one SVM was trained for the binary task of relation detection and the other one for the multiclass classification, in comparison to the approach when only one multiclass SVM was doing both tasks simultaneously. But in general, it can be difficult to decide what type of architecture is appropriate for a given task. Moreover, the problems are becoming even more complicated when the semantic classes to be assigned are taxonomically structured.

In the following parts we give a very short overview on the machine learning methods in the context of information extraction that are later applied in the practical part of this work. Note that we do not cover all machine learning methods that are applicable in the information extraction domain, but rather concentrate on those that were used in our experiments. *Hidden Markov Models* and *Conditional Random Fields* are methods that definitely belong to the state-of-the-art, but require quite a big annotated corpora to achieve top performance. As a result of this, we did not use them in our experiments. More on these techniques can be found in [36].

4.1.1 Naïve Bayes

Probabilistic classifiers view the problem of classification in terms of $P(c|d)$,that is the probability that a document[1] represented by a vector $\overrightarrow{d} =< t_1, t_2, \ldots, t_{n_d} >$ of binary or weighted terms belongs to a class c. Then, this probability is computed by an application of Bayes' theorem, given by

$$P(c|\overrightarrow{d}) = \frac{P(c)P(\overrightarrow{d}|c)}{P(\overrightarrow{d})} \tag{4.1}$$

$P(c)$ is the probability that a randomly picked document belongs to a class c and $P(\overrightarrow{d})$ is the probability that a randomly picked document has vector $\overrightarrow{d}$ as its representation. The

[1] In information extraction we are usually using Bayes' classification in a context of smaller units than documents. For example, in classification of named entities we are using a context window of size n words, or in detection of relation type a context of a sentence is often used. However, from historical reasons, we will refer to the context as a document.

estimation of $P(\overrightarrow{d})$ is problematic, since the number of possible vectors $\overrightarrow{d}$ is too high [48]. To relax this problem, it is common to make the assumption that terms are independent of their position and of all other words in the document. In addition, the prior probability $P(d)$ does not need to be estimated directly, because we normalize over all of the classes. The assumption is encoded by the following equation.

$$P(c|\overrightarrow{d}) = P(c)\prod_{k=1}^{n_d} P(t_k|c) \tag{4.2}$$

This equation corresponds to the *multinomial* naïve Bayes classifier. An alternative to multinomial model is the *multivariate Bernoulli* model. The difference between these two models is in the estimation of $P(t_k|c)$. The Bernoulli model estimates the probability $P(t_k|c)$ as the *fraction of documents* of a class c that contain a term t. In contrast, the multinomial model estimates $P(t_k|c)$ as the *fraction of tokens* or *fraction of positions* in documents of a class c that contain a term t. When classifying a test document, the Bernoulli model uses binary occurrence information, ignoring the number of occurrences, whereas the multinomial model keeps track of multiple occurrences [32].

In text classification, our goal is to find the best class for the document. The best class is the most likely and is often referred to as *maximum a posteriori (MAP)* class.

$$c_{map} = \arg\max_{c\in C} \hat{P}(c|d) = \arg\max_{c\in C} \hat{P}(c)\prod_{k=1}^{n_d} \hat{P}(t_k|c) \tag{4.3}$$

We write $\hat{P}$ for P since we do not know these probabilities and we only estimate their true value from the training set. The probability $\hat{P}(c)$ can be estimated as

$$\hat{P}(c) = \frac{n_c}{N} \tag{4.4}$$

where N_c is the number of documents in class c and N is the total number of documents. The probability $\hat{P}(t|c)$ can be estimated using a maximum likelihood estimates as

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t'\in W} T_{ct'}} \tag{4.5}$$

In order to make these estimates robust with respect to infrequent words, we can use Laplace smoothing

$$\hat{P}(t|c) = \frac{T_{ct}+1}{\sum_{t'\in W} T_{ct'} + B} \tag{4.6}$$

where $B = |W|$ is the total number of unique words in the training set.

The naïve Bayes classifier have been used in numerous scientific studies in document classification. In information extraction we are, however, mainly concerned with smaller text units. In [45] naïve Bayes is used in a traditional information extraction task to extract information about terrorist events. The main motivation relies on the fact that many false hits of information extraction systems were observed in sentences containing subjective language. The experiments were conducted on the MUC-4 information extraction data set with the use of AutoSlog-TS system [44]. Naïve Bayes served here as a filter of subjective sentences.

In [38] is presented a system called BioAnnotator, which is designed for identification and classification of biological terms in biomedical texts. BioAnnotator uses domain-based

dictionary lookup to recognize known terms and a rule engine to discover new terms. Learning module represented by naïve Bayes is involved in the task of semantic classification of terms discovered by the rule engine or by the dictionaries. During the experiment was found that naïve Bayes allows easier scaling and significantly faster training and classification than other learning techniques.

Craven and Kumlein [11] use naïve Bayes for relation classification in biomedical domain. They evaluate relations at the document level and try to find whether there exists a relation between two entities or not. We will discuss this work more in section 7.2.

4.1.2 K-Nearest Neighbours

K-Nearest Neighbours classifier belongs to an *example-based*[2] family of classifiers. These classifiers do not build an explicit, declarative representation of the category c, but rely on the category labels attached to the training documents similar to the test set document [48]. To decide whether a given document d belongs to a class c we look at the k documents that are most similar to d. If there are sufficiently many similar documents, between the k most similar documents, that are classified to the class c, then a positive decision is taken.

One of the most important things with k-NN is a good selection of a distance function. Most common functions are *euclidean* or *cosine* similarity function. The cosine similarity is a common vector based measurement calculating the similarity between two vectors on a $[0, 1]$ scale. Similarity of 1 means for two vectors to be either identical or different by a constant factor. Given feature vectors $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ the cosine similarity is defined as:

$$cos(X, Y) = \frac{x_1 \cdot y_1 + \cdots + x_n \cdot y_n}{\sqrt{x_1^2 + \cdots + x_n^2} \cdot \sqrt{y_1^2 + \cdots + y_n^2}} \tag{4.7}$$

The properties of the k-NN algorithm are quite different from other classification methods. First of all, the method does not need any training, because we only have to determine the k parameter. During the classification, we always compare the given instance to all other instances in the training set. This implies that the time complexity of the method is linear with respect to the size of the training set. As a result, k-NN method have been called *lazy learner* [48], since it postpones the decision on how to generalize until a new query is encountered.

In [26] k-NN is used (taking into account only one nearest neighbour) for detection of named entities in Spanish. They use the k-NN algorithm in the context of self-training described in section 4.2. The main argument for using k-NN is for them its property of taking into consideration every single training example when making the classification decision. This is particularly useful when training data is insufficient.

PowerBioNe [67] is a named entity recognition system in the biomedical domain. Although the features of the model are integrated through a Hidden Markov Model (HMM) and an HMM-based named entity recognizer, the k-NN algorithm is proposed to resolve the data sparseness problem by estimating the probability parameters of the HMM model.

In [66] a k-NN approach to extraction of protein names from MEDLINE abstracts is described. The study focuses particularly on the effects of unbalanced class distribution in the training set. Different ways of choosing negative training examples in the training set in order to achieve better performance are discussed. Not surprisingly, it is indicated

[2] Sometimes also called *memory-based* or *instance-based*.

that better precision is reached with more negative examples in the training set and better recall is achieved with less negative examples in the training set.

4.1.3 Decision Trees

One of the oldest approaches to machine learning are approaches based on learning decision rules and trees. Decision trees are predictive models that map observations about entities to classes that maximize the probability of seeing that observations. A decision tree is an acyclic oriented graph where terminate nodes (leaves) correspond to classes, inner nodes to attributes (features) and edges to values of the attributes. Every decision tree can be easily transformed to decision rules often regarded as *"if-then" rules*. The rules are found by searching combinations of features in the training set that are discriminative for each class. For example, when our aim is to generate positive rules, we try to cover as much positive examples as possible and none or fewest negative examples.

There are two major ways for accessing the feature space. *General-to-specific* methods search the space from the most general towards the most specific hypothesis. One starts from the most general rule possible, which is specialized at the encounter of a negative example that is covered. The principle is to add features to the rules. *Specific-to-general* methods search the hypothesis space from the most specific towards the general hypothesis and will progressively generalize examples. One starts with a positive example, which forms the initial rule for the definition of the concept to be learned. This rule is generalized at the encounter of another positive example that is not covered. The principle is to drop features. The combination of general-to-specific methods and the specific-to-general methods is the so-called *version space* method [36].

Most common tree algorithms such as ID3 or C4.5 construct the decision tree in a top-down way by selecting the most discriminative feature according to some assumption. For each possible value of this feature descendant nodes are then created. To each descendant node examples from the training set that have the same values of features on the path to the root node are assigned. The process is then repeated until all examples associated with a node correspond to just one particular class or until there exist a feature to select. In the former case, we create a leaf node with the label of the class of all examples, in the latter case, we create a leaf node with the label of the majority class among the examples.

The most difficult step of the algorithm is to determine a good discriminative feature. The basic idea often used to find such feature is based on the fact that features which tend to appear rarely carry more information than features that appear very often. In the most extreme case, features that appear with the probability of 1 carry zero information, because they are not discriminative at all. A good function that fulfills this criteria and express this informativeness is $-\log_2(p_i)$ where p_i is the proportion of a set S of all training examples having the value k for some feature f.

This idea is implemented in *entropy* developed by Claude Shannon who was interested in the problem of maximizing the amount of information that you can transmit over an imperfect communication channel [31]. The entropy is defined as the average uncertainty of a random variable. In our context, it measures the average amount of information in a feature.

$$Entropy(S) \quad = \quad -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{4.8}$$

In the process of building the tree, we want to select the feature f that is the most discriminative. After splitting according to this feature we want to get subsets S_v where $v \in Values(f)$ that are significantly less informative than the whole set S. More precisely, the information *gain* of a feature f is the expected reduction in entropy caused by partitioning the examples according to this feature.

$$Gain(S, f) = Entropy(S) - \sum_{v \in Values(f)} \frac{|S_v|}{S} Entropy(S_v) \tag{4.9}$$

where $Values(f)$ is a set of all possible values of feature f and S_v is a subset of S for which feature f has value v.

In the ID3 method, we always select the feature with the highest value of $Gain(S, f)$. One of the drawbacks of this method is that the value of $Gain(S, f)$ is influenced by the number of possible values of feature f. The C4.5 method therefore normalizes the value of information gain.

Rule and tree learning algorithms were the first algorithms that have been used in information extraction and they are still popular learning techniques for information extraction [36]. The main advantage of decision trees is the ability to understand the solution found by the learning algorithm. As a result of this, they are compatible with man-made rules, which can be very useful when applying information extraction systems to a domain where human-revision of rules can be beneficial.

In [50] decision trees (namely C4.5 method) are applied on the task of coreference resolution of noun phrases. The models were trained on the MUC-6 and MUC-7 coreference corpora. The learning approach achieves accuracy comparable to that of nonlearning approaches. The main contribution in this work probably lies in the field of identification of features that mostly contribute to high level of precision and recall.

In [49] an approach to named entity recognition applied to Japanese texts is described. The presented method is not fully automatic, because dictionaries are used to reduce the number of possible classes, which can be assigned to a given entity. Decision trees, which are constructed automatically from the training set, are then used to resolve states in which more than one class can be assigned to a given entity.

The predictive performance of trees is sometimes not as strong on unseen data as that obtained on the training data. This phenomenon is described as overfitting. The problem is that the tree is too specialized to the training data. Most common approaches try to solve this problem by *pre-pruning* or *post-pruning* of the constructed tree. Another way how to deal with this problem is proposed in [4]. It has been observed that the overfitting problem can be greatly reduced by inducing multiple decision trees from the same data. The classification of an unseen case is then determined by a weighted combination of the classifications assigned by the multiple trees. An important step in inducing multiple decision trees is sampling of the training data. A common approach is *bagging* where each classifier is trained on a sample of documents taken from the training set. An alternative method of sampling, which is applicable also to different methods of classifiers, is *boosting*. It is an incremental approach for inducing the tree from a random selection of examples from the training set. The main idea is to increase the probability of selecting examples that have been misclassified by the trees induced in previous iterations. In [4] these boosting trees are applied to categorization of Reuters-21578 collection. The results show that the more decision trees, the better the performance.

An interesting work in the biomedical information extraction was done in [19]. The authors presented an extraction system for assigning protein, gene or mRNA class labels to biomedical terms. Three different learning techniques, namely naïve Bayes, inductive rule learning and decision trees (C4.5 method), were applied in the experiments and the methods were evaluated on a relatively big corpora. The experimental results show that C4.5 achieved slightly higher accuracy than other mentioned methods. However, decision trees and inductive rule learning proved to be significantly slower than naïve Bayes during both training and testing. Another interesting point is that the importance of different types of features was examined. Although their corpora was quite big, it revealed that incorporating relative positional information according to the examined term actually lowered accuracy.

4.1.4 Support Vector Machines

Machine learning techniques used to aim at creating representations that could be well understood by humans. The goal of machine learning in this paradigm was to automatically output accurate rules that cover all positive examples (we call this property *completeness*) and does not cover any negative ones (we call this property *consistency*). Unfortunately, it is not always possible to discriminate between the positive and the negative class, particularly because of the noisy nature of data. This lead the research in machine learning into development of learning methods that are able to learn mathematical function that discriminates the classes. As a result, there have been developed plenty of statistical and neural network methods that are eligible for these kinds of problems. In this section we investigate the method, which have been successfully applied to many problems in information extraction and text classification.

Support Vector Machines (SVM) are in recent years becoming very popular and well recognized. An SVM is a kind of large-margin classifier: it is a vector space based machine learning method where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise) [32]. Figure 4.1 shows the classification using Support Vector Machine. Unlike other linear machine learning methods such as perceptron algorithm, which can find any other linear separator, or like naïve Bayes, which looks for the best linear separator according to some criterion, the SVM defines the criterion by looking for a decision hyperplane that maximizes the distance from any data point. The distance from the decision boundary to the closest data point determines the margin of the classifier. These points are called *support vectors* and they are the only points that determine the position of the separating hyperplane. This approach makes the SVM classifier particularly robust and well suited for classification with low or unbalanced training data.

For the very high dimensional problems common in text classification, sometimes the data are linearly separable. But in general case they are not, and even if they are, we might prefer a solution that better separates the bulk of data while ignoring a few weird noise documents [32]. A standard approach to cope with this problem is to allow the decision hyperplane to make a few mistakes by leaving the width of the margin as big as if the outliers were not present. We then pay a cost for each misclassified example, which depends on the *slack* (a distance from meeting the maximum margin boundary requirements).

Despite the fact that SVMs are inherently binary classifiers, they can be used for multiclass classification as well. Most common approaches are based on a so-called "one-versus-all" classification. Another possibility is to train $\frac{n(n-1)}{2}$ classifiers and to choose the class of a given document that is selected by the most classifiers. Although we have to build a lot

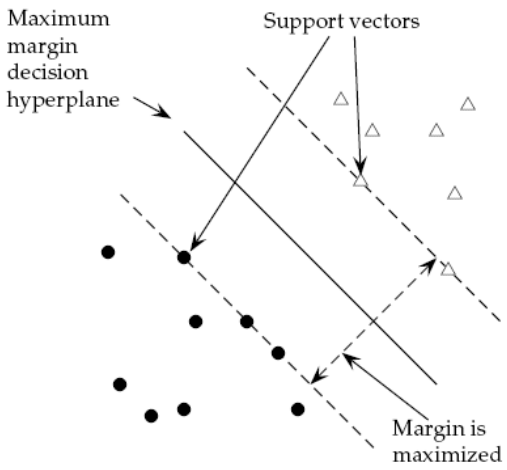


Figure 4.1: The Support Vector Machine classification on the linearly separable task [32].

of classifiers, the time for training may actually decrease, because the amount of training data for each classifier is usually significantly smaller. However, both of these approaches are not the best thing we can do. There exist more sophisticated techniques that can even better deal with multiclass data.

Support Vector Machines are also applicable when the classes are not linearly separable. A general method to solve this problem is to map the feature space on to a higher-dimensional feature space where the training set is linearly separable. Figure 4.2 shows an example of mapping from one-dimensional space, where the points are not linearly separable, on to a two-dimensional space. This approach makes a linear classification in the higher-dimensional space equivalent to non-linear classification in the original space. However, the mapping function has to preserve the relatedness between data points in the higher-dimensional space. This idea is often regarded as *kernel trick*. Function of SVMs relies on a dot product between data point vectors. A kernel function is a function that corresponds to a dot product in some feature space and satisfies the *Mercer's condition*. We will not go in details here and refer the reader for example to [32] for more comprehensive description. Some common kernels are Radial Basis Function (RBF), Polynomial and Sigmoid. Recently, there have been developed special kernels that can work over the syntactic tree of a sentence. They are called dependency tree kernels.

Although one would expect more complex kernel to perform better than, for example, the linear one, it is difficult to assess such behavior in practice. In [61] was performed an experimental comparison of SVMs with linear and non-linear kernels on the task of text categorization of Reuters articles. They obtained slightly better result with the linear SVM than with the non-linear models. Sometimes it is therefore argued that the linear kernels seem to be more appropriate for text classification tasks. Moreover, it should be also taken into account that the linear models are usually substantially faster to train.

The time to train an SVM is actually a weakness of this classifier. More precisely, the theoretical complexity of training SVMs is cubic with respect to the size of the training set. All the recent work on SVM training has work to reduce that complexity, often by being satisfied with approximate solutions. Standard empirical complexity is about $O(|D|^{1.7})$ [32] where $|D|$ is the number of instances in the training set. On the other hand, in [23]

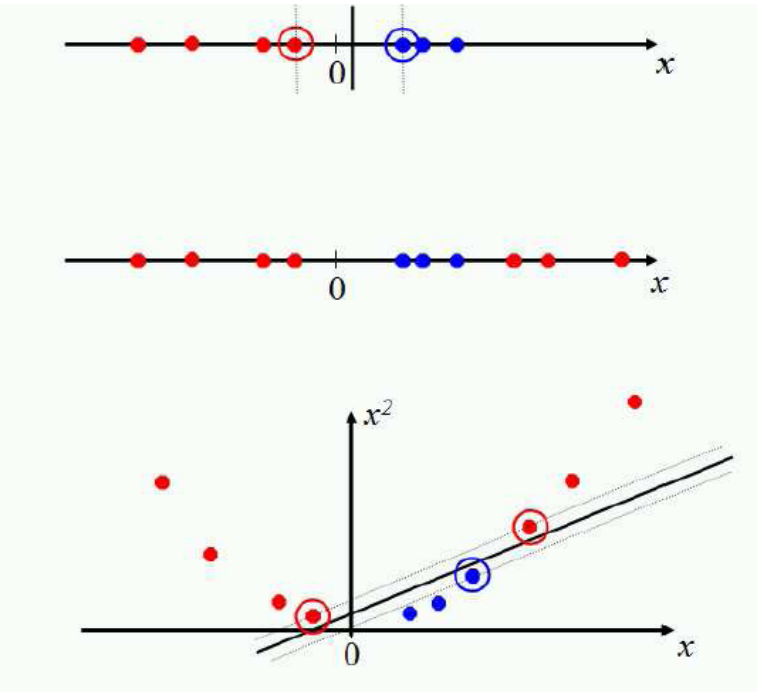


Figure 4.2: Projecting data that is not linearly separable into a higher dimensional space can make it linearly separable [32].

they conclude that SVMs consistently achieve good performance on classification tasks, outperforming existing methods substantially and significantly. Their ability to generalize well in high-dimensional feature spaces eliminates the need for feature selection, which makes their application considerably easier. Their advantage over conventional methods is their robustness. Furthermore, SVMs do not require any parameter tuning, since they can find good parameter settings automatically.

Because of the fact that the solution found by SVMs is not readable by humans, there is currently much effort in rule extraction from SVMs. In [13] we can find five main points describing the motivation behind rule extraction of comprehensible rules from SVMs. The first point refers to the *provision of a user explanation capability*, which can be used internally for reasoning and learning, and externally for the explanation of results to a user. The second point claims that the internal states of the machine learning system are both accessible and can be interpreted unambiguously. This property is called *transparency* and it would allow the exploration of regions in generalisation space that may lead to erroneous or sub-optimal states. The next point refers to the fact that rule extraction algorithms currently do not allow for verification, i.e. they do not prove that a machine learning system behaves according to some specification. However, rule extraction algorithms provide a mechanism for either partially or completely *decompiling* an SVM. This is about half-way to software verification because it allows for a comparison between the extracted rules and the specification. One of the main points is that if a limited or unrepresentative data set has been used in the training process, it is difficult to determine if and when generalisation fails for specific cases. By expressing learned knowledge as a set of rules, an experienced user can anticipate or predict a generalisation failure. The last point that has been one

of the primary objectives for rule extraction is essential for data mining and knowledge discovery. The idea is that a knowledge system may discover hidden features in the input data whose importance was not previously recognized.

Generally, SVMs are applied in information extraction very often. In biomedical domain they were used, for example, for gene and protein name recognition in [35, 54, 22]. Relevant work on biomedical relation information extraction can be found in [64, 28, 14]. Some of this work also deals with special kernels proposed for relation extraction. Overall, the research in SVMs indicates that SVMs usually outperform other classification methods.

4.2 Weakly-supervised Techniques

Since creation of large annotated corpora for every new domain requires a lot of manual labour, techniques that are able to generalize from a very small set (*seed*) of examples are needed. These techniques rely on acquisition of new patterns from unannotated corpora. In the best case, these techniques work fully automatically and during the learning process we are consistently discovering valuable patterns. These techniques involve *expansion* (can be sometimes regarded also as *self-training*) and *co-training*. If this is not our case, we can still apply *active learning*. In active learning the algorithm itself decides what are the best candidates for annotation in order to maximize the speed of the learning process.

Bootstrapping refers to a technology that starts from a small initial effort and gradually grows into something larger and more significant [36]. The main principle of *expansion* is that we apply a small seed of extraction patterns (learned from a few examples) on the unannotated corpora in order to discover new extraction patterns. If the new member is considered sufficiently similar to other members of the training set, it is added to the training set. The process can be iterated until no valuable patterns are discovered.

Co-training is based on the idea that more experts may be better than one if their judgments are combined appropriately. Following this basic idea, n classifiers are trained using the same seed of extraction patterns, however, with disjoint, and best conditionally independent features. Another possibility is to train them using different classification methods. At each iteration, the classifiers are applied on the same set of unlabeled examples. The examples labeled with the highest confidence, concerning various voting strategies such as *majority voting* or *adaptive classifier combination (ACC)*, are then added to the training set. This process is iterated until a certain level of accuracy is reached.

Machine learning algorithms are generally applied using a randomly selected training set. However, in many settings, we also have the option of using *pool-based* active learning. In *active learning* the classifier has access to a pool of unlabeled data and can request the class label for a certain number of instances in the pool. The main issue with active learning is finding a way to choose good requests or queries from the pool [57].

Chapter 5

Evaluation of Information Extraction Systems

An important feature of each information extraction system is its ability to provide quantitative measure of its performance. Evaluating and comparing information extraction systems is not an easy task and it should not be underestimated. A traditional evaluation method is to compare the results of the system to some *golden standard*, which is often hand-crafted by a human annotator. Not only is this task tedious, but on many information extraction tasks the annotations assigned by human can be hardly considered correct. Problems can arise, for example, with subjective language. In [60] was compared the performance of four highly skilled and experienced human annotators with three top information extraction systems applied to the MUC-5 task. The results indicate that humans achieved up to 82% precision and 79% recall. Machines were able to reach 57% precision and 53% recall on the same task. When we evaluate to a golden standard in a domain where it is difficult to assign classes objectively, we expect that a sufficiently high level of the so-called *inter-annotator* agreement was reached.

All these problems, however, do not imply that information extraction systems cannot be evaluated with high reliability. First of all, different criteria should be applied depending on the application domain of an information extraction system. For instance, in some cases, especially when huge amount of data is available, we want to find out whether there is an evidence that a certain disease can be treated by a certain drug. In this case, system that has a high precision (explained in 5.1) and low recall is appropriate. To the contrary, when the goal of the extraction system is only to filter out the information that is not important for us and the rest of the information is to be revised by a human curator, high recall is more needed. Moreover, we can often evaluate on different levels of abstraction. If we are searching for information that applies generally, we can evaluate at a document or even higher level. This is desirable when we are asking general questions such as whether a given gene and a given disease are related to each other. On the other hand, when our aim is to precisely locate pieces of information to provide a user with proofs for our claim that a given gene can actually cause a disease, it is better to evaluate at a sentence level. To summarize, it is extremely important to understand well the goals of the systems and to define the evaluation task precisely.

Many of the information extraction metrics were defined during the *Message Understanding Conferences (MUC)* in 1990s [36]. Overall, although all the participants of these conferences felt strongly that evaluation is extremely important, they conclude it is costly

	Yes is correct	No is correct	
System decides yes	true positives (tp)	false positives (fp)	$tp + fp$
System decides no	false negatives (fn)	true negatives (tn)	$fn + tn$
	$tp + fn$	$fp + tn$	$tp + fp + fn + tn$

Table 5.1: Contingency table for a set of binary decisions.

and requires a very large amount of time and effort [8]. The *Automatic Content Extraction (ACE)* competition currently develops its own metrics. Since 1992, *Text Retrieval Conference (TREC)* has been also operating hosted by National Institute of Standards and Technology (NIST), which has developed several metrics for large-scale evaluation of text retrieval methodologies. TREC has accelerated the transfer of research ideas into commercial systems and helped to boost the effectiveness of retrieval systems.

It is important to stress that information extraction is usually not a final product, but more often a component, which can be used in information retrieval, text summarization or data mining. Because of this, we distinguish between *intrinsic* evaluation, where only the performance of the extraction task is measured, and *extrinsic evaluation*, where is measured the performance of the whole system in which information extraction component is involved [36]. In this work, our aim is to evaluate only information extraction components, thus we will always do intrinsic evaluation.

Besides considering the quality of information extraction there are also other criteria for judging the performance. First of all, information extraction is computationally expensive and even if the computer power has grown substantially, we should still take care of *computational complexity*. Another criterion to consider is the *domain-coverage*, which refers to the fact that some extraction systems might work well in a limited domain, but it may be hard for them to generalize well in case they are applied to previously unseen observations. It is, however, often very difficult to specify the concept of the domain in order to be able to evaluate this coverage. It is natural that we can think of more criteria, such as *interoperability*, referring mainly to input and output format of the extraction system, *portability* and *extensibility* of the extraction system, *sensitivity* to the linguistic quality of the input etc.

5.1 Standard Metrics

Information extraction adopts the standard evaluation metrics well-known from text classification, which are mainly *precision*, *recall* and *accuracy*. Each classification task can be divided into n binary tasks. The result of these n decisions can be summarized in a *contingency table* 5.1. Each entry in the contingency table specifies the number of decisions with the corresponding result. Columns of the table refer to decisions of an expert and rows to decisions of a given information extraction system that we are evaluating. Information extraction systems are often used in the retrieval context where this table is applicable only with a slight change. The row "System decides yes" only corresponds to the number of retrieved items and the row "System decides no" corresponds to the number of entries that were not retrieved by the system. In the same fashion, the column "Yes is correct" and the column "No is correct" correspond to the number of relevant and nonrelevant items in the corpus.

Given the binary contingency table we can now define precision, recall and accuracy.

$$precision = \frac{tp}{tp + fp} \tag{5.1}$$

$$recall = \frac{tp}{tp + fn} \tag{5.2}$$

$$accuracy = \frac{tp + tn}{tp + fp + fn + tn} \tag{5.3}$$

In information extraction, precision refers to the proportion of items that the system assigns their class correctly. Recall is the proportion of all correct items to the number of items retrieved by the system. In the extreme case every system can achieve perfect recall by never deciding "No" or always retrieving everything. Perfect precision can be achieved by never deciding "Yes" or never retrieving anything. An alternative is accuracy, which is a good measure in classification tasks when we are assigning to multiple disjoint classes.

5.2 Cross-validation

If we use only one static division of the train and test set, we cannot be really sure that our results are reliable and that they are not only an artifact of random fluctuations. Besides that, we are often suffering from a lack of training data. To make our result more reliable and to improve the size of both train and test set, we should adopt cross-validation.

In *k-fold* cross-validation, sometimes called rotation estimation, the data set D is split into k mutually exclusive subsets (the folds) $(D_1, D_2, \ldots, D_k)$ of approximately equal size. The classifier is trained and tested k times; each time $t \in \{1, 2, \ldots, k\}$, it is trained on $D \setminus D_t$ and tested on D_t. The cross-validation estimate of accuracy is the overall number of correct classifications, divided by the number of instances in the data set [25]. *Leave-one-out* cross validation is a special case of k-fold cross validation where k is equal to the number of instances.

Chapter 6

Extraction of Relations from General Texts

Information extraction systems used to rely mainly on symbolic, handcrafted knowledge. This knowledge is usually represented by a set of handmade rules. Although these systems provide satisfactory results and are often able to be used in practical applications, significant effort is needed to build them. Another disadvantage is that their rules are often very specific for a target application and they are therefore not easily portable to different domains.

Our aim in this work is to work mainly with information extraction methods in the biomedical domain. However, this chapter can rather be considered as a motivation example of an extremely simple extraction technique that is applicable in general domain. The aim of this chapter is to develop a simple method for automatic extraction of lexical patterns from a selection of Wikipedia articles.

Wikipedia articles consist mainly of unstructured text, but structured information such as different kinds of infobox templates is also present. Consider that we would like to know what is the capital city, the area or population of the Czech Republic. The structured information in the infobox in Figure 1.1 can help us to find the solution with less effort than finding it in the unstructured text. As a matter of fact, if the population of the state has changed, it is highly inconvenient to change the information on two places (both in the infobox and in the unstructured text). Information extraction system that can extract and synchronize the information in infoboxes with the information in text would be helpful.

6.1 Data set

We created a new data set from a selection of Wikipedia articles in order to evaluate the performance of a method which we propose later in this chapter. The data set contains 1.6 MB of text stored in 50 files. Each file corresponds to a particular country. Only articles that at least once mention the name of their national capital city were selected. By the word "mention" we refer to all parts of the article from which a human can recognize the name of a certain capital city.

To be able to provide the data set with labels, we took the advantage of DBpedia project. DBpedia provides *Resource Description Framework (RDF) triples* that can be easily used to determine the capital of a particular country (current goal or label). The DBpedia infobox extraction algorithm detects infobox templates and recognizes their structure using pattern

matching techniques. The templates are then parsed and transformed to RDF triples[1] [5]. To summarize, DBpedia extracts the knowledge from structured templates and transforms it to RDF triples that can be used, for example, to answer sophisticated queries. To the contrary, our work focuses on extraction of structured information from unstructured text, so the RDF triples can be used to provide our database with labels quite easily.

Note that all the structured information appearing in the articles was naturally discarded from our data set, so it contains plain text only. The articles were tokenized and sentences were split using GATE[2]. Furthermore, all files were manually inspected and sentences, which can serve as a clue to determine the name of a capital city, were annotated. Finally, these capital city entities together with the names of their corresponding countries form a set of positive examples. Although the whole data set contains only 99 of these positive example pairs, it also contain a range of sentences that seem quite problematic. We will discuss this in detail in section 6.4.

6.2 Task Specification

For better understanding of the actual task, assume that we would like to find all sentences that precisely state what is the capital city of a certain country. This task looks similar to one defined in [2], where the goal is to extract relation pairs, such as *organization-location* from plain text. In [2] the proposed Snowball system is expected to extract these relations between two entities that are both identified in a text. In opposition to Snowball, we are here interested in finding relations between the whole article, which refers to a certain country, and possibly a few capital city entities.

To give an example of the Snowball task consider the text, *"computer servers at **Microsoft**'s headquarters in **Redmond**,"* from which the pair Microsoft - Redmond should be extracted. To the contrary, the presented system should extract for instance the pair Belize - Belmopan from the file **Belize** when the sentence, *"Over several years, the British colonial government designed a new capital, **Belmopan**, at the exact geographic centre of the country, and in 1970 began slowly moving the governing offices there."* is encountered.

6.3 Method

The system is initially provided with a number of example pairs *country - capital*. The expected output are new country - capital pairs extracted from the data set. We assume that we can decide if a word is a capital city for a given country by exploring its context. The presented approach uses lexical features weighted according to their importance for the task.

The **first step** is to search for all occurrences of a capital, provided by an example from the training set, in a corresponding text. Its context words are then explored. Note that the size of a context window is fixed during an experiment.

We distinguish context words appearing on the left hand side of the entity from context words appearing on the right hand side. Context words are also weighted using a simple idea similar to *term frequency-inverse document frequency (tfidf)*: most indicative words tend to appear often in a context of capital city, but they rarely appear in different contexts. Using this approach it was for example quite easy to automatically detect that the most

[1] The DBpedia database is freely available on the project website `http://dbpedia.org/About`.

[2] http://gate.ac.uk/

indicative word for this task is the word "capital." Summarizing the paragraph, weighted words on the left and right hand side of a capital city entity form a pattern. We refer to those patterns as candidate patterns.

The **second step** is to prune the set of candidate patterns. Our candidate patterns can contain patterns extracted from sentences that mention the name of a capital city A (A is a member of the training set), but it is impossible to conclude from them that A is the capital city. For example, it is incorrect to derive that Prague is capital city from the sentence: *The occupation ended on 9th May 1945 with the arrival of Soviet and American armies and the **Prague** uprising.* Patterns extracted from such sentences can cause errors while identifying new entities. Therefore, minimal confidence threshold is used to discard them. Candidates with low confidence are considered unreliable and are eliminated from further evaluation. Other candidates are considered reliable and form a pattern set.

The **third step** is to use the learned patterns in the pattern set for identification of new relations from unseen articles. The algorithm scans the text of an article word by word. All words beginning with a capital letter are considered as potential candidates for new capital city entities. Their left hand side context words together with the right hand side context words form a pattern as in the first step. This pattern is compared with all patterns in the pattern set. A pattern similarity threshold controls how flexible the patterns are in identifying new entities.

6.4 Evaluation Methodology

This section describes how the proposed algorithm was evaluated. Firstly, we define the *Ideal*[3] set, which is a set of country - capital city pairs we want to retrieve from the whole data set. Finally, we evaluate the system using standard precision and recall metrics and discuss the results.

6.4.1 Creating the *Ideal* set

For small text collections it is possible to inspect documents manually and create the *Ideal* set by hand. This is exactly what we did in section 6.1 by annotating all sentences from which a correct relation can be derived. The *Ideal* set then contains all country - capital city pairs extracted from these sentences. It is important to stress that each relation pair relates to a particular sentence or instance of a capital city entity.

Another way of defining the *Ideal* set is that it contains for each country exactly one country - capital city pair. In the first definition the *Ideal* set can contain the same pair more than just once, because each pair is related to one sentence from which the relation can be derived. To the contrary, following the latter definition we have for each country, which is represented by some article, exactly one pair. Roughly speaking, if we use the former definition of the *Ideal* set, we are also interested in sentences from which the knowledge is derived, however, in the latter, we are only interested in deriving correct pairs for each country.

To demonstrate the difference between these two approaches, consider these sentences from article about Afghanistan.

*In 1504, Babur, a descendant of both Timur Lang and Genghis Khan, established the Mughal Empire with its capital at **Kabul**.*

[3]The *Ideal* set serves here as a golden standard.

He was succeeded by his son, Timur Shah Durrani, who transferred the capital from Kandahar to ***Kabul****.*

When evaluating the results of the proposed method using the first definition of the *Ideal* set, both relations *Afghanistan - Kabul* from the first sentence and *Afghanistan - Kabul* from the second one must be extracted, and no other relations should be found. On the other hand, using the second definition, we aim to extract the relation pair *Afghanistan - Kabul* and it does not matter from which sentence we actually extract it. We even consider the extraction correct in case it derives the relation from the following sentence.

Several important centers of Khorasan are thus located in modern Afghanistan, such as Balkh, Herat, Ghazni and ***Kabul****.*

In our evaluation we refer only to the first "strict" definition of the *Ideal* set.

To show how this task is, in fact, difficult it should be taken into account that for any method based on pattern matching it is unlikely to discover that the relation that can be extracted from the following sentence is not correct.

Ahmad Shah Durrani created a large empire with its capital at ***Kandahar****.*

The relation *Afghanistan - Kandahar* is not considered correct, because Kandahar is not any more the capital of Afghanistan. Although for human it is very easy to find out that this relation is not valid, the algorithm would have to be able to work with time dependencies.

6.5 Results

Unfortunately, our data set is quite small. *Leave-one-out* cross validation is therefore a good estimation method in order to improve the size of the training set.

A couple of tests with a different size of the context window were performed and evaluated using standard metrics. Given the *Ideal* set from section 6.4.1 we can define precision and recall more formally. We define *Precision* as

$$Precision = \frac{\sum_{i=0}^{|Extracted|} |l_i = l'_i|}{|Extracted|} \tag{6.1}$$

where $Extracted$ is a set of relation pairs that were extracted by the system and $[l_i = l'_i]$ is equal 1 if the test value l_i matches the extracted value l'_i and 0 otherwise. Similarly we define *Recall* as

$$Recall = \frac{\sum_{i=0}^{|Extracted|} |l_i = l'_i|}{|Ideal|} \tag{6.2}$$

Thus, *precision* refers to a proportion of correctly identified relation pairs to the size of the extracted pairs and *recall* to a proportion of correctly identified pairs to the whole *Ideal* set.

The results are reported in Figures 6.1 and 6.2. The algorithm performed quite well on the first one hundred extracted items. Then the precision is going down and it is evident that it is quite hard for the method to identify new entities. This may be caused by a limited

size of the context window that may still be too small to cover all important context words. The results also indicate that we were able to discover more entities using a bigger size of the context window.

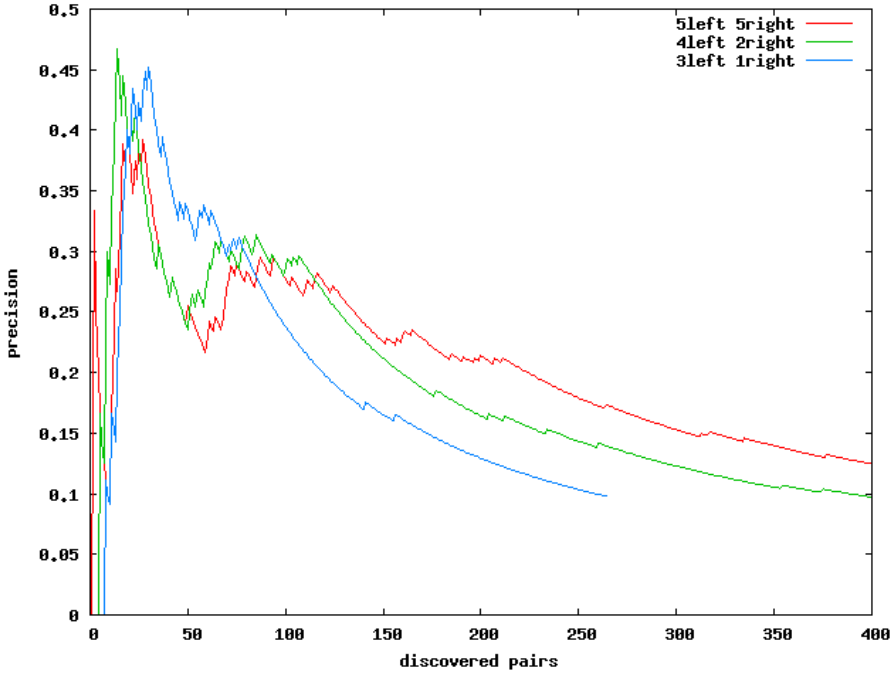


Figure 6.1: Precision of the proposed method

Here we present some of the sentences from which the relations were correctly identified:

The Parliament of Austria is located in ***Vienna*** *the nations largest city and capital*
The Government of Kazakhstan transferred its capital from Almaty to ***Astana*** *on December 10 1997*

On the contrary typical mistakes were derived from sentences such as:

Homel *with 481000 people is the second largest city of Belarus and serves as the capital of the* ***Homel*** *Oblast*
Cordoba*, Muslim Spain's capital, was the largest, richest and most sophisticated city of medieval Europe.*

From the first sentence the incorrect relation *Belarus - Homel* was discovered and from the second one *Spain - Cordoba*.

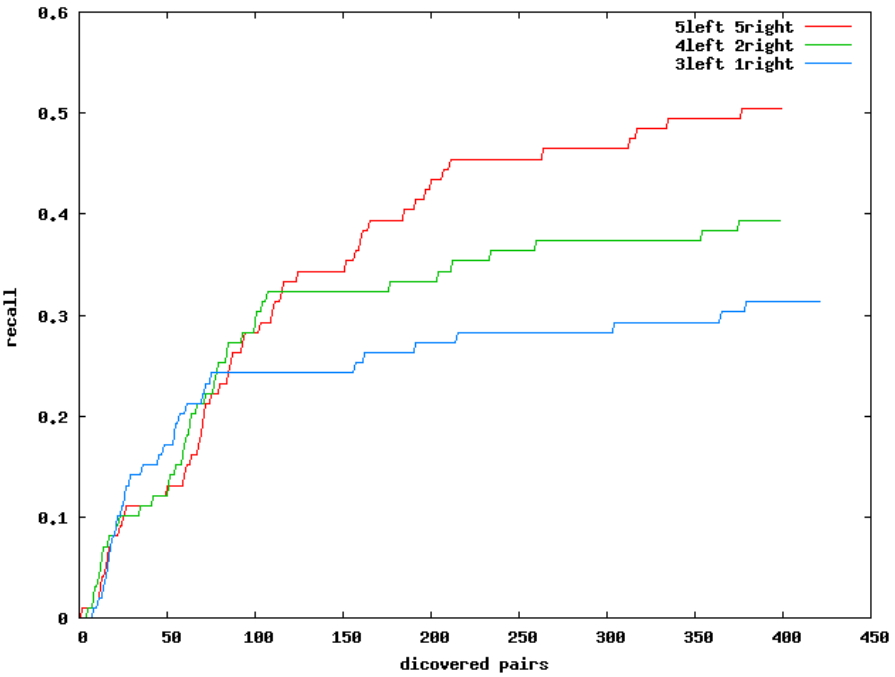


Figure 6.2: Recall of the proposed method

Chapter 7

Extraction of Relations from Biomedical Texts

The amount of biomedical information is growing explosively and new and useful results are appearing every day in research publications. Many of these publications are stored in online databases such as *MEDLINE*, *Online Mendelian Inheritance in Man (OMIM)* [17], *Yeast Protein Database (YPD)* or *HIV-1 Human Protein Interaction Database*. These databases should ideally provide advanced query capabilities to help the user in getting the desired information as fast as possible. As a matter of fact, automatic extraction of useful information from these online resources remains a challenge [53], and biomedical resources such as *MEDLINE* records can still be viewed as a greatly underutilized source of information. Current systems for accessing *MEDLINE* (for example *PubMed*) accept keyword-based queries to text sources and return documents that are hopefully relevant to the query [11]. Our goal is, in contrast, to extract valuable semantic relations in order to support more complex queries. As a result, a retrieval system based on this technology should be able to return precise answers, rather than just a list of relevant documents.

First of all, in section 7.1 we identify the state-of-the-art in information extraction of relations from biomedical text, then in section 7.2 we show how information retrieval systems in biomedical domain can be improved using machine learning techniques. Finally, in section 7.3 we present results in classification of different types of *protein - protein* interactions into multiple classes.

7.1 Introduction and Problem Specification

In chapter 6 we developed a system that can identify relations between a country and its capital city. Although we call the task *relation extraction*, because we were interested only in capital city entities which belong to the country specified by the name of a given article and not in capital city entities which could occur in the article accidentally, the task has many similarities with *named entity recognition*. To the contrary, in this chapter we are interested in semantic relations between two named entities appearing in a single sentence.

In particular, the problem can be divided into two parts:

- **Named entity recognition**
  The entities of interest are *proteins* and *subcellular locations*. For example, given the sentence:

Vam6p$_{PROTEIN}$ *fused with a green fluorescent protein were highly accumulated in a few specific regions of the vacuolar*$_{SUBCELLULAR\ LOCATION}$ *membranes.*

We want to extract strings corresponding to the semantic roles $PROTEIN$ and $SUBCELLULAR\ LOCATION$.

- **Relation recognition**
  The system should identify the type of the relation that holds between the semantic roles in a sentence. For example, given the sentence above, we want the system to classify the sentence as containing the *accumulated in* relationship.

The methods we are developing are using dictionaries that were automatically extracted from annotated corpora and contain a list of *proteins* and *subcellular locations*. In this part of my work, we make the assumption that named entity recognition can be based only on these dictionaries. Of course, this assumption does not always hold in practice. Overall, we can make the following types of mistakes. In our corpora may be *proteins* that are not present in our dictionary. Besides that, we can also make a mistake by misclassifying some other named entity as a *protein*.

Relation recognition task is necessary for answering precise queries about entities that may interact. For example, we would like to know if some disease can be treated by a certain drug. A very common approach to this task is to perform named entity recognition and then imply the existence of a semantic relation by co-occurrence assumption. This assumption is based on the idea that two instances of different named entities tend to appear together in a document more likely in case they are related. This means that we are deciding whether in a document D containing two named entities A and B there is a relation between A and B or not. Although this assumption is not very useful in case entities A and B appear in the training set relatively rarely, it works considerably well on large collections. Unfortunately, the rare events are those that a user is often mainly interested in, for example, when searching if a new drug for a certain disease has been developed.

7.2 Improving Information Retrieval by Extraction of Semantic Relations

In this section, we describe our experiments performed on the task of relation extraction from biomedical databases. We briefly cover the methods that can be used for this task and compare a simple co-occurrence predictor method against statistical machine learning methods. In the rest of the introduction to this section, we present the related work in this field. In section 7.2.1 the databases used in the experiments are described. In section 7.2.2 and 7.2.3, we show the relations of interest and present the extraction method. In sections 7.2.4 and 7.2.5, we evaluate the system and discuss the results we have achieved.

In [51] was developed a prototype system for retrieving and visualizing information from literature and genomic databases using gene names. The premise of the work is a hypothesis that if two genes have a related biological function, then their co-occurrence within the biomedical literature is more likely. From the collection of *MEDLINE* documents the authors construct a graph with edges between pairs of genes. The edge inclusion is determined by a user-defined threshold. The length of the genes is a function of the occurrence of the two genes within the literature.

In [52] a knowledge discovery method that can identify related genes is presented. The approach relies on multiple Thesauri, representing domain knowledge as gene names and terms describing gene functions. The method is another variant of a statistical co-occurrence, using an idea similar to traditional $tfidf$. When a relation is predicted, it is classified to classes, such as *activates*, *binds*, *releases*, *regulates* etc. Unfortunately, this method can easily suffer from lack of data.

In [11] is presented an approach in which relation between two named entities is implied in case they appear in a single sentence. Relations between two terms that occur more often are considered more likely to hold. In our experiments, we use this approach to compare with a real baseline. Next, we develop machine learning algorithms, which should approve better performance than this relatively simple method.

7.2.1 Data Set

For our experiments we have used annotated data set created from a selection of articles retrieved from *Yeast Protein Database (YPD)*.

YPD is a database for the protein of the budding yeast, *Sacharomyces cerevisiae* [21]. By the way, it is the first annotated database for the complete proteome of any organism. The most rapidly growing part of *YPD* is the textual annotation. Approximately 25 000 lines of text now describe the functions, mutant phenotypes, physical interactions, domain structures, similarities to other proteins etc. Annotations are drawn from approximately 3500 yeast papers and abstracts, and more than 8700 yeast papers are cited in the reference list [21]. Our selection of articles from YPD database contains 7128 sentences from which 780 describe a semantic relation of interest. The size of the data set is 4.0 MB including annotations.

7.2.2 Relations of Interest

In our task we are interested in extraction of the following relation

- **subcellular-localization (Protein, Subcellular-Structure):**
  The instances of this relation represent proteins and the subcellular structures in which they are found.

The sentence from which a subcellular-localization relation can be extracted may, for example, look like the following one

Subcellular fractionation demonstrated that $\textbf{\textit{Gtt1p}}_{PROTEIN}$ *associates with the* $\textbf{\textit{endoplasmic reticulum}}_{SUBCELLULAR-STRUCTURE}$.

Here, the relation pair (GTT1, endoplasmic reticulum) should be extracted.

7.2.3 Extraction via Text Classification

In this section we present the method we have used in our experiment. The method is similar to the one used in [11] and [9]. An example of the extraction process is given in figure 7.1.

First of all, we are addressing the task of extracting instances of a binary relation $R \subseteq X \times Y$, where X and Y are sets of concepts. We assume that we are given semantic lexicons l_X and l_Y that can be defined as a mapping from a set $W = T \cup C$ of terms T and

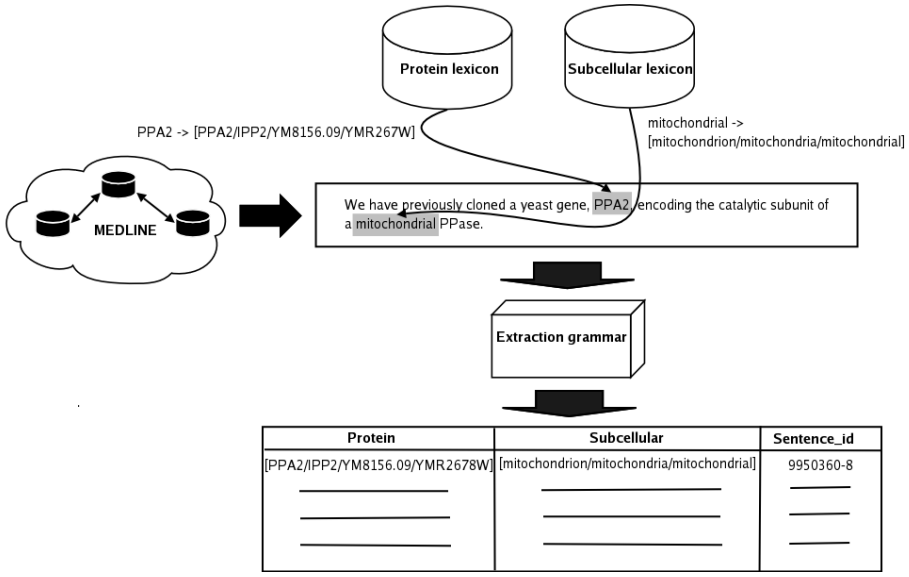


Figure 7.1: The extraction process on the task of *protein-subcellular location* relation extraction.

collocations C, appearing in the data set, to concepts $l_X : W \rightarrow X$ and $l_Y : W \rightarrow Y$. One of the reasons why we define lexicons in this way is that, for example, one protein has often more different lexical representations. Figure 7.1 shows a case where protein *PPA2* can have four different lexical representations *PPA2, IPP2, YM8156.09, YMR267W*. Thus, l_X and l_Y are mapping each term and collocation, which denotes a named entity of interest, to a specific concept.

Given such lexicons, the next step is to identify instances that could possibly express a relation in a document. In our work the instances correspond to sentences. This implies that we are not able to detect relations between two entities that are not appearing in a single sentence. This can result in a slight decrease in recall. However, this decrease is not substantial at all and therefore we neglect it. When we scan the input text, we identify all entities corresponding to members of the set X and Y using our lexicons. We extract a relation instance $(x, y) \in R$ from a sentence S in case:

(i) $\exists w_1, w_2 \in S : l_X(w_1) = x \wedge l_Y(w_2) = y,$ where $x \in X \wedge y \in Y$
Note that S is a set containing all terms and collocations of the sentence.

(ii) Sentence is classified as positive by a statistical machine learning technique.

The first condition says that we first identify candidate concepts x and y. Then, we classify the sentence using a classifier, which can be seen as a filter of sentences that are not describing any relation. If the sentence is classified as positive, the relation is extracted. The statistical machine learning model can be learned from labeled positive and negative instances in the training set.

As we stated in the foregoing paragraphs, we make the assumption that relation pairs always appear within a single sentence and therefore we are not able to find relations described over a couple of sentences. Unfortunately, we are also facing an opposite problem. Consider, for example, a sentence that mentions multiple terms corresponding to the set X

and multiple terms corresponding to the set Y. The sentence may specify that the relation holds only between some of the terms of the set X and Y. However, we can only classify the sentence as being a member of the positive class, in which case we extract all possible relation pairs, or we classify the sentence as being a member of the negative class, in which case we extract no relation pairs. Generally, we are in this work limited to instances that correspond to sentences. If we worked with a smaller instances such as in chapter 6, we would have to deal with overlapping relationships[1].

In order to learn statistical models for classification of sentences, we represent each sentence using a *bag-of-words* approach. In this approach, one looks at the frequency of words in a text, without considering their order.

In our experiments we have worked with naïve Bayes 4.1.1, decision tree 4.1.3 and Support Vector Machine classifiers 4.1.4. In addition, we also worked with a k-Nearest Neighbour classifier 4.1.2 in many different types of document representations investigating also a similar approach to the one described in Snowball [2]. In Snowball not only whole sentences are classified, but also smaller patterns that divide words in a sentence to the left hand side of the first candidate term, middle context and to the right hand side of the second candidate term. Unfortunately, the k-Nearest Neighbour classifier in all different settings proved to be quite inefficient. This is because it always needs to compare the pattern we want to classify with all other patterns in the training set. In general, the scalability of this classifier is therefore not very good. As a result of this, with k-NN we were not able to run enough experiments to achieve significant results.

7.2.4 Evaluation

In chapter 6 we considered an extracted relation to be correct only if it was retrieved from a sentence that indeed describes it. In this evaluation, we set the task in a different way. Since our data set is relatively big, in comparison to the one used in the previous experiment, and we would like to compare our results to a co-occurrence predictor, described in the beginning of this section, we are satisfied with a table of relation pairs. This table can be useful, for example, when there is a query for all proteins appearing in the endoplasmatic reticula.

An extracted relation is therefore considered as being correct, in case there exists a sentence in the corpus in which the relation holds. It should be noted that as in chapter 6 we evaluated correct decisions made on the sentence level, here we are evaluating on the relation level.

In order to be able to evaluate the results of the system in terms of standard precision and recall measures, we sort the relation pairs according to their confidence in descending order. The confidence of a relation pair is determined by the number of times this relation pair has been extracted. For example, when a relation pair is extracted multiple times, it has a bigger confidence than a relation pair that has been extracted only once. Once the relation pairs are sorted using this method, they are compared to the correct table of relation pairs. This table is created from annotations that have been assigned by an expert with biological background. We refer to this set as $Ideal$.

In comparison to the previous task (chapter 6), the precision and recall is defined in the following way. Let $L = l_1, l_2, \ldots, l_n$ be a sequence of all relation pairs that have been extracted by the system sorted in descending order according to their confidence, and let

[1] An approach based on frame theory could be possibly used, but we did not investigated it in this work.

Ideal be the set of all correct relation pairs in the data set. The precision for some $k < n$ is than defined as

$$Precision = \frac{\sum_{i=0}^{k} |l_i \in Ideal|}{k} \tag{7.1}$$

where $[l_i \in Ideal]$ is equal 1 if relation pair $l_i \in Ideal$ and 0 otherwise. Similarly we define *Recall* as

$$Recall = \frac{\sum_{i=0}^{k} |l_i \in Ideal|}{|Ideal|} \tag{7.2}$$

7.2.5 Results

To improve the reliability of the results and to avoid random fluctuations, the data set was divided into 5 folds preserving the proportion of positive and negative sentences in each fold. After that, we run the experiments[2] and cross-validated across the folds. Pictures 7.2 and 7.3 show the precision and recall of the system on the subcellular-localization task.

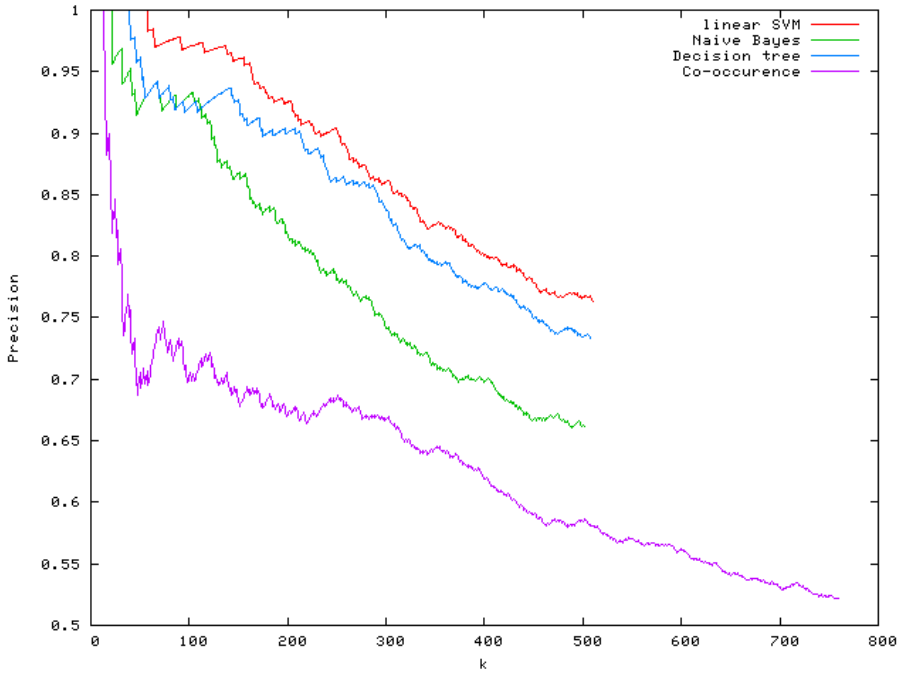


Figure 7.2: Precision of the system on the extraction of subcellular-localization relation from the YPD database

The graphs indicate that we were able to improve the results substantially in comparison to the co-occurrence predictor. Moreover, with decision tree we achieved 94% precision in comparison to 68% precision of the co-occurrence predictor at $k = 143$. This corresponds to approximatelly 25% recall. The linear SVMs performed best. Naïve Bayes was the fastes to train, but has been outperformed by both decision tree and SVM.

[2]The extraction system is implemented in Python and uses classifiers from the Orange toolkit `http://www.ailab.si/orange/`.

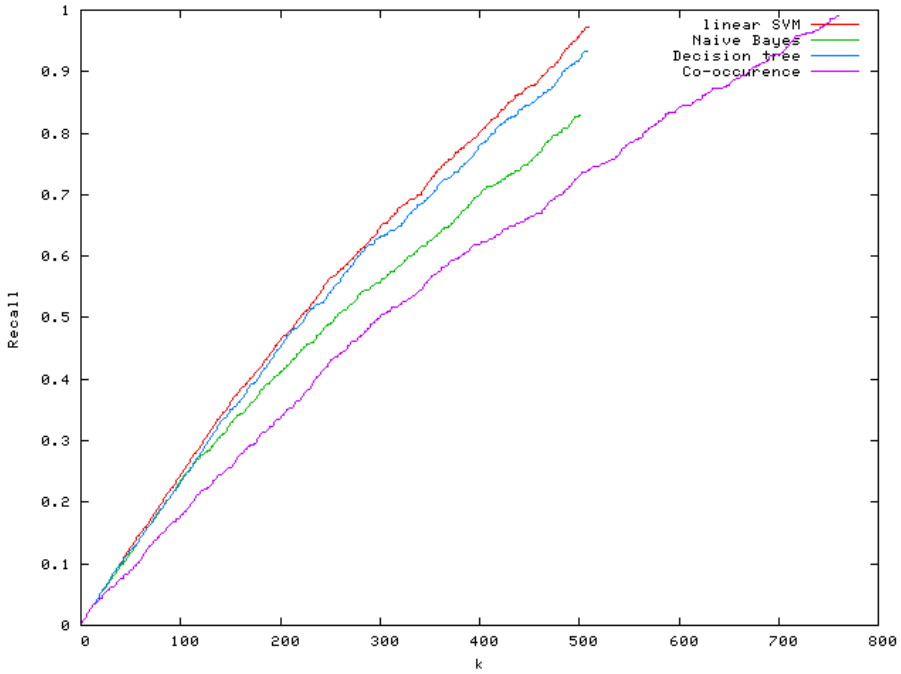


Figure 7.3: Recall of the system on the extraction of subcellular-localization relation from the YPD database

Although the improvement in precision over the co-occurrence method is significant, all three presented machine learning methods achieved lower recall than the co-occurrence predictor. The recall of SVMs was, however, only slightly lower[3]. This is caused by the fact, that we either classify the sentence as being a member of the positive or being a member of the negative class. This problem can be solved by setting a task more like a regression, rather than classification. For example, the model should, instead of strictly accepting or rejecting instances, return the estimated probability that the instance is positive. In [11] it is shown that the naïve Bayes estimated posterior probability that each sentence is in the positive class can be combined with the class probabilities using the *noisy or* function [40].

$$confidence = 1 - \prod_{k}^{N}[1 - Pr(c = pos|s_k)] \tag{7.3}$$

Here, $Pr(c = pos|s_k)$ is the probability estimated by naïve Bayes for the k-th element of our set of sentences. Although in [11] naïve Bayes in this setting proved to be less precise than classical naïve Bayes, it naturally provided better recall. In particular, the recall was identical to the co-occurrence predictor. An interesting possibility would be to combine the high precision of the former method with the high recall of the latter one. In [42] the authors show that it is possible to build a hybrid classifier that will perform at least as well

[3]Note that the recall in our experiments is slightly bigger than the one measured in [11]. The reason for this is that we create the *Ideal* set from the annotated corpora where only relation pairs appearing in a single sentence are mentioned. To the contrary, in [11] they took into account also the relation pairs appearing in a broader context. As a result, our 100% recall corresponds approximately to 78% recall mentioned in [11].

as the best available classifier for any target conditions. In some cases, the performance of the hybrid can actually surpass that of the best known classifier.

7.3 Semantic Classification of Protein-Protein Interactions

Protein-protein interactions refer to the association of protein molecules and the study of these associations from the perspective of biochemistry, signal transduction and networks. Identification of these interactions remains one of the greatest challenges in modern genomics. In this section we address the problem of multi-way relation classification, applied to identification of the interactions between proteins from bioscience text. This section builds on the work of [47].

The main difference between the relation extraction task specified in section 7.2 and this one is that this section not only detects whether there is a relation or not, but rather specifies the type of the semantic relation. As it is shown in [46], some approaches in this field simply report that a relation exists between two proteins, but they do not determine which relation holds (for example the task specified in section 7.2), while most others start with a list of interaction words and label only those sentences that contain these trigger verbs. In order to improve the accuracy of both of these approaches, we will use machine learning techniques to classify the relations into multiple semantic classes based on their semantics.

7.3.1 Data Set

In our experiments we use a collection of sentences from *HIV-1 Human Protein Interaction Database*. This database provides scientists in the field of HIV/AIDS research a concise, yet detailed, summary of all known interactions of HIV-1 proteins with host cell proteins, other HIV-1 proteins, or proteins from disease organisms associated with HIV/AIDS [1]. This database is manually curated. The main reason for using this database is that it contains information about the type of protein interactions and it can therefore serve as a good source of labeled data.

As in [47], we refer to the combination of a protein pair PP and an article A as a *"triple."* The goal is to automatically determine the interaction type of PP pairs in each triple and use this information to assign correct interaction type to the whole triple. For example, the following sentence, which mentions the PP pair (*cdk7, tat*) found in article A, is associated with some interaction type.

These results are consistent with a recent report by (Chen and Zhou 1999) , who observed that $\boldsymbol{Tat}_{PROT2}$ *can still stimulate transcription elongation after the removal of* $\boldsymbol{CDK7}_{PROT1}$ *from TFIIH by treatment with high salt and immunodepletion.*

Decisions about all such sentences in a triple are combined in order to assign correct interaction type *(stimulates)* to the whole triple.

Although the database in some cases reports multiple different interactions for a given triple, these triples have been excluded from our experiments. However, PP pairs with multiple interactions are included in case these interactions are reported by different articles.

In our experiments, we are using a data set[4], which has been created from full-text journal papers referring to a random subset of the PP pairs. In addition, our data set

[4]Available online at `http://biotext.berkeley.edu/data.html`.

Interaction	Papers		Citances	
	sentences	triples	sentences	triples
Degrades	60	5	63	6
Synergizes with	86	4	101	10
Stimulates	103	11	64	13
Binds	98	10	324	36
Inactivates	68	12	92	10
Interacts with	62	10	100	21
Requires	96	6	297	9
Upregulates	119	11	98	16
Inhibits	78	12	84	23
Suppresses	51	5	99	7
Total	821	82	1322	151

Table 7.1: Number of interaction sentences and triples

also contains sentences that have been extracted from other articles, which explicitly cite a given article A. In [41] is introduced a neologism, *citances*, to mean sentence(s) surrounding the citation within a document. The main idea behind this is that citances in bioscience literature are more likely to state biological facts than arbitrarily chosen sentences in an article. They also tend to be more conscious, since the authors try to summarize previous related work, which has already been described in detail in the original paper.

We selected 10 interactions of Table 7.1 that fulfills the requirement of minimum number of 40 sentences for each type of interaction. The sentences that contain PP pairs describing one of these interactions have been collected and form our data set. In [47] it is argued that a hand-assessment of the individual sentences shows, that there are some sentences in the data set that mention the target proteins PP, but do not, in fact, describe the interaction I. Thus, the labels to the PP pairs are assigned at the document level (to determine if the algorithm can predict the interaction that a curator would assign to a document as a whole given the protein pairs).

7.3.2 Classification Methods

As we have discussed in the previous section, our goal is to assign interaction I to a given PP pair. In [47] two generative (naïve Bayes and dynamic graphical model based on previous work on relation extraction) and one discriminative (feed-forward Neural Network with hyperbolic tangent activation function trained with a gradient descent method) models were used. On the contrary, we are using naïve Bayes, k-Nearest Neighbours, decision trees and Support Vector Machines.

Our problem can be described as a 10-way classification problem, which is more complex than most of the related relation extraction work where the task is usually to make binary predictions.

In our experiments we want to:

(i) Compare the results of the mentioned machine learning techniques on this task.

(ii) Bring down the amount of labeled data needed to train the models.

First, we train and evaluate different learning methods on the data set using 5-fold cross-validation described in 5.2. The evaluation is measured at a document level referring to the fact that sentences in each triple always describe the same relation type. We always ensure that sentences from one triple do not appear in both training and testing set[5]. This way of division is very important. We tried evaluating relations when the sets were not divided in this way, which resulted in almost 10-20% boost in accuracy. Unfortunately, this boost was caused by overfitting of the classifiers that were, in fact, doing document identification instead of relation classification.

This time, we do not use precision and recall measures for evaluation. Our data set consists of sentences containing PP pairs and therefore classification of a single sentence can only result in correct or incorrect association of a particular protein interaction to a given PP pair. That is why, accuracy is more appropriate in this case and refers to a proportion of correctly assigned interaction labels to the size of the whole data set.

Second, we investigate the idea of active learning described in 4.2 and initially train the models from a very small seed of example sentences. In particular, we train in our experiments the models on a seed that contains for each interaction type only one example triple. The seed was chosen manually, however, no special care was taken to the selection process. In the future, it would be interesting to see, what impact on the results can have a good selection of seed sentences. The learned model is used to classify the rest of sentences in the pool. The sentence which is assumed to help us most is in turn selected from the pool and we ask a qualified user to associate an interaction type to it. We will explain how to choose this sentence later. The sentence with its corresponding triple sentences is then added to our seed and the whole process is repeated until a sufficient level of accuracy is reached. The evaluation of the second approach can be measured as a level of accuracy at a given number of labeled triples. In this way, we would like to demonstrate that a proper selection of training sentences indeed improves the accuracy with respect to the number of labeled sentences.

For the active learning task we used Support Vector Machine learners. Our *SVM active learner* contains a query component that, given a current training set, decides which instance from the pool to query. Because SVMs operate on the *version space*, we shall choose to query instances that reduce the version space as much as possible. Intuitively, one good way of doing this is to choose a query that halves the version space. Since it is not practical to explicitly compute the sizes of the new version space, we should approximate this procedure. In [56] a *simple margin* method is explained, which results in a natural rule: learn an SVM on the existing labeled data (seed) and choose as the next instance to query the instance that comes closest to the hyperplane.

7.3.3 Results

First of all, for each sentence corresponding to a certain triple we find the interaction that maximizes the probability of interaction given the features and then assign to all sentences of this triple the most frequent interaction[6]. We do not use protein names as features for

[5]Note that in [47] roughly 75% of the data set was used for training and the rest for testing. In our experiments, we use k-fold cross-validation where the folds are divided in such a way that each fold contains approximately the same proportion of interaction types with respect to the whole data set. Thus, our results can slightly differ and comparisons should be taken with care.

[6]Note that in [47] it is indicated that this leads to a slightly better results than to choose the interaction that maximizes the sum over all the triple's sentences.

Method	Accuracy	
	Papers	Citances
linear SVM	63.4%	68.2%
Decision tree	59.8%	54.9%
k-NN	57.3%	58.9%
naïve Bayes	57.3%	62.2%
Base	11.0%	9.9%

Table 7.2: Accuracies for the task of 10-class labeling of *protein-protein* interactions

Truth	Predictions										
	D	SyW	St	B	Ina	IW	R	Up	Inh	Su	Acc. %
Degrades (D)	6	0	0	0	0	0	0	0	0	0	100.0
Synergizes with (SyW)	0	3	0	0	0	1	0	2	4	0	30.0
Stimulates (St)	0	0	5	3	0	1	4	0	0	0	38.5
Binds (B)	0	0	1	28	1	1	1	0	4	0	77.8
Inactivates (Ina)	0	0	0	0	10	0	0	0	0	0	100.0
Interacts with (IW)	0	2	0	6	0	5	2	2	3	1	23.8
Requires (R)	0	1	0	0	0	0	8	0	0	0	88.9
Upregulates (Up)	0	0	0	0	0	0	0	14	2	0	87.5
Inhibits (Inh)	0	1	0	2	1	0	1	1	17	0	73.9
Suppresses (Su)	0	0	0	0	0	0	0	0	0	7	100.0

Table 7.3: Confusion matrix for the linear SVM for citances. The members of the matrix correspond to the number of triples.

the learned models, although this can help to reach a better performance. This is because we consider this approach unfair since the classifiers can overfit on the data set.

The results on the first task are reported in picture 7.2. The linear SVM classifiers performed best (the confusion matrix for citances is reported in table 7.3)[7]. However, other methods also worked relatively well in comparison to the baseline. The baseline is measured using cross-validation and indicates the accuracy we achieve in case, we always classify to the majority class determined from the training set. The results present that on papers we achieved 63.4% with the linear SVMs, which is about 5.6% more than the dynamic model presented in the work of Rosario [47]. In this experiment, our baseline seems to be comparable (we measured 11.0% and they 11.1%), the results for naïve Bayes seem to be very similar as well. To the contrary, on citances we have measured quite different level of the baseline and therefore we are not able to make a comparison. The reason for this is probably, as it was stated above, different division of the training and testing set. In classification with decision trees, it is possible to work with many different variants and parameters. As a result of this, we cannot conclude that decision trees are not able to reach slightly better accuracy than it is reported in table 7.2.

Another interesting point is that we generally achieved slightly higher accuracies with citances. A possible explanation for this may be that citances really provide more conscious description of semantic relations, as it is claimed in [41]. On the other hand, we can also argue that the higher accuracy of citances could have been caused by having more training data in each fold. However, there may be some other possible explanations.

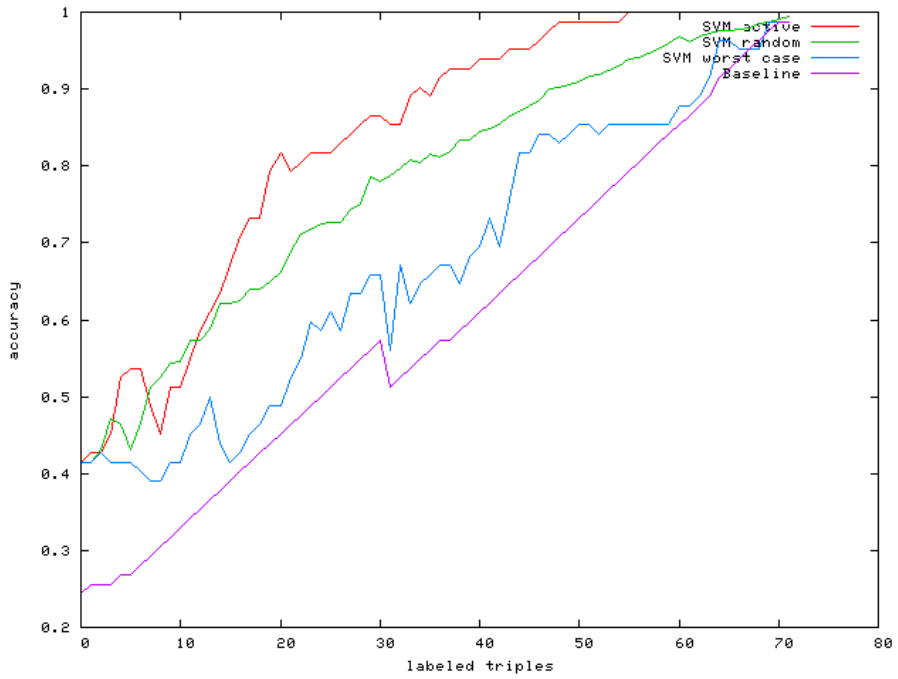


Figure 7.4: Accuracy of the system with respect to the number of labeled sentences from papers

[7]We also experimented with nonlinear variants of SVMs, but slightly better results were obtained with the linear kernel. This conforms with the study of [61].

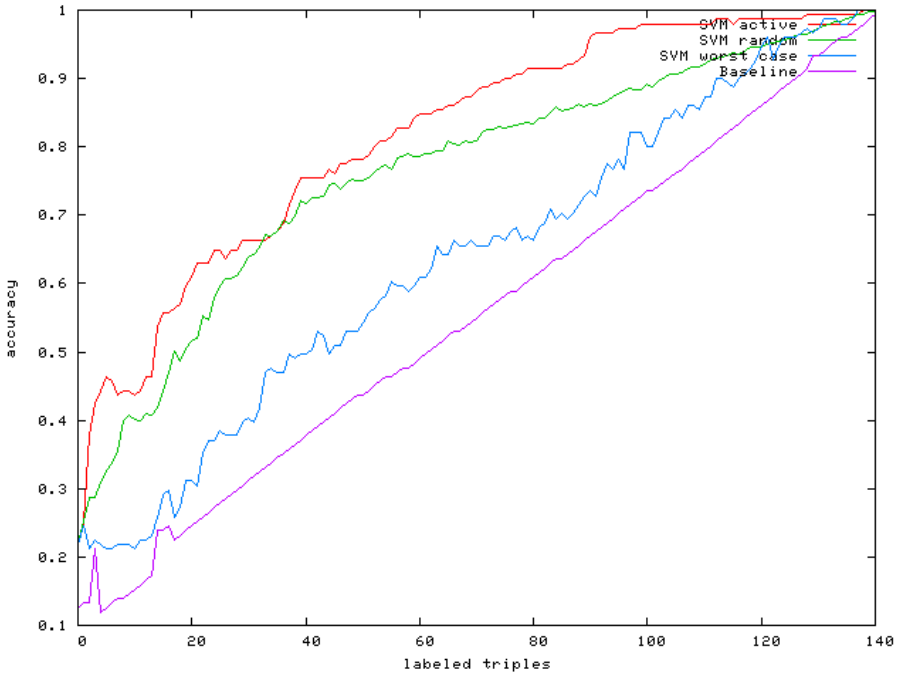


Figure 7.5: Accuracy of the system with respect to the number of labeled sentences from citances

The results of the active learning experiment are reported in figures 7.4 and 7.5. We have experimented only with linear SVMs, which proved to have the best performance in our previous experiments. The baseline shows the accuracy of the system, if we always assign a given sentence to the majority class according to the composition of the training set. Note that in the baseline method, we always ask the expert to label the sentence (triple) about which interaction type we are most unsure (which corresponds to a class that has fewest examples in the training set).

Our results indicate that the method we use to choose which sentences to label indeed approximately minimizes the number of labeled sentences needed to achieve a reasonable level of accuracy. We say approximately minimizes, because we are not able to prove it. It is possible and quite probable that there exist sequences of sentences to query from the pool that lead to better results. To prove that our method chooses the best sequence we would have had to first try all of them, but this is computationally infeasible. As a matter of fact, our query function can be considered only as a rough approximation of the best query function. The same holds for our approximation of the worst case method. In this method we always ask the expert to label the sentence from the pool about its class we are most sure. This corresponds to an instance that does not lie in the margin of the SVM. The random method chooses the instances from the pool randomly. Therefore, the results of this method were averaged over five runs. During our experiments, the random method performed substantially better than the worst case method, but still considerably worse than the active learning method.

Chapter 8

Conclusion and Future Work

In this work, we studied methods for automatic extraction of semantic information from text. In the theoretical part, we presented the current state-of-the-art and discussed the advantages and disadvantages of several methods applicable to information extraction. It was presented how information units can be represented by feature vectors and then used in turn by machine learning techniques. We also indicated a wide range of research studies, taking into account applications in the biomedical domain, that adopted some of these techniques in their experiments.

Later, we have developed a system for pattern extraction in a general domain and evaluated its performance on the selection of Wikipedia articles. This system served us as a basic experiment that inspired us to try more sophisticated techniques. Finally, we successfully applied these techniques in the biomedical context and discussed their results in detail. One of the main contributions can be seen in application of active SVM learners to the problem of 10-way protein interaction type classification.

Our research indicates that information extraction systems can achieve a sufficient level of performance to be applicable in practice. Particularly, information extraction from biomedical texts is currently a domain where actually almost every system which achieves better results than simple statistical techniques is of real value.

On the other hand, there is still a lot of space for future work. In our experiments, it would be interesting to use more advanced features in both relation detection and relation classification. A good step forward would be to experiment with the grammatical structure of sentences. The grammatical structure can be, for example, represented by dependency trees. Fortunately, SVMs offer a promising solution to deal with features in this form using dependency tree kernels. We would like to experiment with different features and find, which features contribute most to the performance of the extraction.

Overall, one of the most interesting issues would be to compare our results for both detection and labeling to results that can be achieved by humans and by hand-made rules. Another thing to consider is to investigate the approach of induction of decision rules from SVMs. The solution found by SVMs can be then inspected from the linguistic point of view.

Finally, and more generally, although current research in information extraction is often focusing on quite difficult extraction tasks, these tasks are mostly designed to extract very specific entities or relations. On one hand, it is extremely important to have a great number of comparisons of different approaches to information extraction applied to many different tasks. An advantage of such research studies is that it is possible to measure and discuss the performance of extraction systems on such limited tasks, which is becoming considerably

more difficult in more complex tasks. On the other hand, there is still no firm evidence on applications of the more difficult extraction tasks in complex systems that take information extraction only as a necessary input component for reasoning, advance querying, semantic web etc.

Bibliography

[1] National institute of allergy & infectious disease: Hiv protein-interaction database, 2008. [Online; accessed 2-April-2008].

[2] Agichtein, E., and Gravano, L. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries* (2000).

[3] Aitken, J. Learning Information Extraction Rules: An Inductive Logic Programming approach. In *Proceedings of the 15th European Conference on Artificial Intelligence* (2002), pp. 355–359. http://citeseer.ist.psu.edu/586553.html.

[4] Apte, C., Damerau, F., and Weiss, S. Text mining with decision trees and decision rules.

[5] Auer, S., Bizer, C., Lehmann, J., Kobilarov, G., Cyganiak, R., and Ives, Z. Dbpedia: A nucleus for a web of open data. In *Proceedings of ISWC 2007 (To Appear)* (2007).

[6] Biemann, C. Ontology learning from text: A survey of methods. *LDV Forum 20*, 2 (2005), 75–93.

[7] Burget, R. Visual html document modeling for information extraction. In *RAWS 2005* (2005), Faculty of Electrical Engineering and Computer Science, VSB-TU Ostrava, pp. 17–24.

[8] Chinchor, N., Lewis, D. D., and Hirschman, L. Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). *Comput. Linguist. 19*, 3 (September 1993), 409–449.

[9] Chun, H. W., Tsuruoka, Y., Kim, J. D., Shiba, R., Nagata, N., Hishiki, T., and Tsujii, J. Extraction of gene-disease relations from medline using domain dictionaries and machine learning. *Pac Symp Biocomput* (2006), 4–15.

[10] Cimiano, P., and Völker, J. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)* (2005), pp. 166–172.

[11] Craven, M., and Kumlien, J. Constructing biological knowledge-bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology* (Germany, 1999), pp. 77–86.

[12] Culotta, A., and Sorensen, J. S. Dependency tree kernels for relation extraction. In *ACL* (2004), pp. 423–429.

[13] Diederich, J. Rule extraction from support vector machines: An introduction. In *Rule Extraction from Support Vector Machines*, J. Diederich, Ed., vol. 80 of *Studies in Computational Intelligence*. Springer, 2008, pp. 3–31.

[14] Eom, J.-H., and Zhang, B.-T. Mining protein interaction from biomedical literature with relation kernel method. In *ISNN (2)* (2006), J. Wang, Z. Yi, J. M. Zurada, B.-L. Lu, and H. Yin, Eds., vol. 3973 of *Lecture Notes in Computer Science*, Springer, pp. 642–647.

[15] Feldman, R., and Sanger, J. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, December 2006.

[16] Fleischman, M., and Hovy, E. Fine grained classification of named entities. In *Proceedings of the 19th international conference on Computational linguistics* (Morristown, NJ, USA, 2002), Association for Computational Linguistics, pp. 1–7.

[17] Hamosh, A., Scott, A. F., Amberger, J., Bocchini, C., Valle, D., and Mckusick, V. A. Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders. *Nucl. Acids Res. 30*, 1 (January 2002), 52–55.

[18] Harabagiu, S., and Maiorano, S. Acquisition of linguistic patterns for knowledge-based information extraction, 2000.

[19] Hatzivassiloglou, V., Duboue, P. A., and Rzhetsky, A. Disambiguating proteins, genes, and rna in text: A machine learning approach.

[20] Hearst. Automated discovery of wordnet relations.

[21] Hodges, P. E., Payne, W. E., and Garrels, J. I. The yeast protein database (ypd): a curated proteome database for saccharomyces cerevisiae. *Nucleic Acids Research 26*, 1 (1998), 68–72.

[22] Ji, K., Ohta, M., and Tsujii, Y. Tuning support vector machines for biomedical named entity recognition, 2002.

[23] Joachims, T. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning* (Chemnitz, DE, 1998), C. Nédellec and C. Rouveirol, Eds., no. 1398, Springer Verlag, Heidelberg, DE, pp. 137–142.

[24] Kavalec, M., and Svatek, V. Information extraction and ontology learning guided by web directory, 2002.

[25] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI* (1995), pp. 1137–1145.

[26] Kozareva, Z., Bonev, B., and Montoyo, A. Self-training and co-training applied to spanish named entity recognition. In *MICAI* (2005), A. F. Gelbukh, A. de Albornoz, and H. Terashima-Marín, Eds., vol. 3789 of *Lecture Notes in Computer Science*, Springer, pp. 770–779.

[27] Lee, L. "i'm sorry dave, i'm afraid i can't do that": Linguistics, statistics, and natural language processing circa 2001, 2004.

[28] Li, J., Zhang, Z., Li, X., and Chen, H. Kernel-based learning for biomedical relation extraction. *J. Am. Soc. Inf. Sci. Technol. 59*, 5 (2008), 756–769.

[29] Loper, E., and Bird, S. Nltk: The natural language toolkit, 2002.

[30] Maedche, A., Neumann, G., and Staab, S. Bootstrapping an ontologybased information extraction system, 2002.

[31] Manning, C. D., and Schütze, H. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 1999.

[32] Manning, Ch. D., R. P. S. H. *Introduction to Information Retrieval*. Cambridge, July 2008.

[33] McDowell, L., and Cafarella, M. J. Ontology-Driven Information Extraction with OntoSyphon. In *International Semantic Web Conference* (2006), pp. 428–444.

[34] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. Introduction to wordnet: An on-line lexical database*. *Int J Lexicography 3*, 4 (January 1990), 235–244.

[35] Mitsumori, T., Fation, S., Murata, M., Doi, K., and Doi, H. Gene/protein name recognition based on support vector machine using dictionary as features. *BMC Bioinformatics 6*, Suppl 1 (2005).

[36] Moens, M.-F. *Information Extraction: Algorithms and Prospects in a Retrieval Context (The Information Retrieval Series)*. Springer, October 2006.

[37] Mohammad, S., and Pedersen, T. Combining lexical and syntactic features for supervised word sense disambiguation. In *Proceedings of CoNLL-2004* (2004), Boston, MA, USA, pp. 25–32.

[38] Mukherjea, S., Subramaniam, L., Chanda, G., Sankararaman, S., Kothari, R., Batra, V., Bhardwaj, D., and Srivastava, B. Enhancing a biomedical information extraction system with dictionary mining and context disambiguation. *IBM Journal of Research and Development 48*, 5/6 (2004), 693–701.

[39] Pantel, P., and Pennacchiotti, M. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (July 2006).

[40] Pearl, J. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988.

[41] Preslav, N. I., Ariel, S. S., and Marti, H. A. Citances: Citation sentences for semantic analysis of bioscience text. In *Workshop on Search and Discovery in Bioinformatics*.

[42] Provost, F., and Fawcett, T. Robust classification systems for imprecise environments. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence* (Menlo Park, CA, USA, 1998), American Association for Artificial Intelligence, pp. 706–713.

[43] Riloff, E. Automatically constructing a dictionary for information extraction tasks. *Proceedings of the Eleventh National Conference on Artificial Intelligence* (1993), 811–816.

[44] Riloff, E. Automatically generating extraction patterns from untagged text. In *AAAI/IAAI, Vol. 2* (1996), pp. 1044–1049.

[45] Riloff, E., Wiebe, J., and Phillips, W. Exploiting subjectivity classification to improve information extraction. In *Proc. 20th National Conference on Artificial Intelligence (AAAI-2005)* (2005), pp. 1106–1111.

[46] Rosario, B. Extraction of semantic relations from bioscience text.

[47] Rosario, B., and Hearst, A. Multi-way relation classification: Application to protein-protein interaction. In *Human Language Technology Conference on Empirical Methods in Natural Language Processing* (2005).

[48] Sebastiani, F. Machine learning in automated text categorization. *ACM Computing Surveys 34*, 1 (2002), 1–47.

[49] Sekine, S., Grishman, R., and Shinnou, H. A decision tree method for finding and classifying names in japanese texts, 1998.

[50] Soon, W. M., Ng, H. T., and Daniel. A machine learning approach to coreference resolution of noun phrases. *Comput. Linguist. 27*, 4 (December 2001), 521–544.

[51] Stapley, B., and Benoit, G. Biobibliometrics: Information retrieval and visualization from co-occurrences of gene names in medline asbtracts, 2000.

[52] Stephens, M., Palakal, M., Mukhopadhyay, S., Raje, R., and Mostafa, J. Detecting gene relations from medline abstracts. *Pac Symp Biocomput* (2001), 483–495.

[53] Subramaniam, V. L., Mukherjea, S., Kankar, P., Srivastava, B., Batra, V. S., Kamesam, P. V., and Kothari, R. Information extraction from biomedical literature: methodology, evaluation and an application. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management* (New York, NY, USA, 2003), ACM Press, pp. 410–417.

[54] Takeuchi, K., and Collier, N. Bio-medical entity extraction using support vector machines.

[55] Tanev, H., and Magnini, B. Weakly supervised approaches for ontology population. In *EACL* (2006), The Association for Computer Linguistics.

[56] Tong, S. *Active learning: Theory and Applications*. PhD thesis, August 2001.

[57] Tong, S., and Koller, D. Support vector machine active learning with applications to text classification. In *Proceedings of ICML-00, 17th International Conference on Machine Learning* (Stanford, US, 2000), P. Langley, Ed., Morgan Kaufmann Publishers, San Francisco, US, pp. 999–1006.

[58] Wactlar, H. New directions in video information extraction and summarization. In *10th DELOS Workshop* (June 1999).

[59] Wikipedia. Wikipedia — wikipedia, the free encyclopedia, 2007. [Online; accessed 29-December-2007].

[60] Will, C. A. Comparing human and machine performance for natural language information extraction: results from the tipster text evaluation. In *Proceedings of a workshop on held at Fredericksburg, Virginia* (Morristown, NJ, USA, 1993), Association for Computational Linguistics, pp. 179–193.

[61] Yang, Y., and Liu, X. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1999), ACM Press, pp. 42–49.

[62] Yangarber, R., and Grishman, R. Machine learning of extraction patterns from unannotated corpora: Position statement, 2001.

[63] Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics* (1995), pp. 189–196.

[64] Yeh, A. S., Hirschman, L., and Morgan, A. A. Background and overview for kdd cup 2002 task 1: Information extraction from biomedical articles. *SIGKDD Explorations 4*, 2 (2002), 87–89.

[65] Yildiz, B., and Miksch, S. ontox - a method for ontology-driven information extraction. In *ICCSA (3)* (2007), O. Gervasi and M. L. Gavrilova, Eds., vol. 4707 of *Lecture Notes in Computer Science*, Springer, pp. 660–673.

[66] Zhang, J., and Mani, I. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets* (2003).

[67] Zhou, G., Zhang, J., Su, J., Shen, D., and Tan, C. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics 20*, 7 (May 2004), 1178–1190.