

CONAN: TEXT MINING IN THE BIOMEDICAL DOMAIN

RAINER MALIK

SIKS Dissertation Series No. 2006-15 The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.



Copyright ©2006 Rainer Malik

ISBN-10: 90-393-4288-1

ISBN-13: 978-90-393-4288-6

CONAN
Text Mining in the Biomedical Domain

CONAN
Tekst Mining in het Biomedische Domein
(met een samenvatting in het Nederlands)

CONAN
Text Mining in der Biomedizinischen Domäne
(mit einer Zusammenfassung in deutscher Sprache)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus prof. dr. W.H. Gispen, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 11 oktober 2006 des ochtends te 10.30 uur

door

Rainer Malik
geboren op 19 Juni 1978 te Linz, Oostenrijk

Promotor: Prof. dr. A.P.J.M. Siebes

Contents

List of Figures	xi
List of Tables	xiii
Acknowledgments	xv
Part I Introduction	
1. INTRODUCTION	3
2. EXISTING TEXT MINING SYSTEMS	5
2.1 Information Retrieval (IR)	5
2.2 Natural Language Processing	6
2.3 SDI Services	7
2.4 Biological Named Entity Recognition (NER)	8
2.4.1 Problems in NER	8
2.5 Information Extraction (IE) and Text Mining	10
2.5.1 Systems	10
2.5.2 Text Mining	10
2.6 Microarray Analysis	11
2.7 Knowledge Discovery	12
2.8 Conclusions	12
3. PROBLEMS WITH EXISTING SYSTEMS	13
4. RESULTS OF THIS THESIS	15
4.1 Overview of this thesis	16

References	19
Part II Components	
5. BIOLOGICAL DATABASES	27
5.1 PubMed/MEDLINE	27
5.1.1 PubMed	27
5.1.2 MeSH	28
5.2 Ensembl	29
5.3 UniProt	31
5.4 Gene Ontology	32
5.5 UMLS Services	33
5.6 iProLink	34
5.7 SwissProt	35
5.8 Other Services	36
5.8.1 GOA	36
5.8.2 IPI	37
6. CONAN PROGRAM COMPONENTS	39
6.1 BLAST	39
6.2 AbGene	42
6.3 NLProt	43
6.4 MuText	44
6.5 PreBIND	45
7. TECHNICAL COMPONENTS	49
7.1 XML	49
7.2 XPath	51
7.3 Script Languages	52
7.4 Classification Techniques	54
7.4.1 Support Vector Machines	54
7.4.2 Boosting	55

References	59
Part III CONAN	
8. CONAN	65
8.1 Input	65
8.1.1 MEDLINE XML Files	66
8.1.2 GOA	68
8.1.3 IPI	68
8.2 Pre-Processing	69
8.3 Processing	69
8.3.1 Data Integration	70
8.4 Output	71
8.4.1 General	71
8.4.2 BLAST	72
8.4.3 AbGene	73
8.4.4 MuText	73
8.4.5 NLProt	73
8.4.6 PreBIND	75
8.4.7 MeSH	75
8.5 Boosting	75
References	77
Part IV Experiments	
9. MEASURES IN TEXT MINING	81
9.1 Precision and Recall	81
9.2 F-measure	83
9.3 Inter Annotator Agreement	84
10. RESULTS	87
10.1 Experimental Setup	87
10.2 Evaluation	88
10.2.1 Corpora	89

10.2.2	Prodisen	89
10.2.3	BioCreative Corpus	93
10.2.4	YAPEX Corpus	95
10.2.5	LLL Challenge Corpus	95
10.2.6	LDD Corpus	96
10.2.7	Boosting Evaluation	100
10.2.8	Other Corpora	101
10.3	Interesting Results and Usefulness	103
10.3.1	Protein Names	103
10.3.2	Mutations	104
10.3.3	Interactions	104
10.3.4	Keywords	106
References		107
11.	APPLICATIONS	109
11.1	Command Line	109
11.2	Web Server	112
11.2.1	System Architecture	112
11.3	Queries	113
11.3.1	PMID	113
11.3.2	Keyword	116
11.3.3	Protein	116
11.3.4	Mutation	117
11.3.5	Interaction	119
11.3.6	Ensembl, UniProt, Gene Ontology	122
11.4	Gene Interaction Networks	123
11.4.1	Constructing the Network	124
11.4.2	How Literature Can Help	126
References		131

Part V Discussion

12. DISCUSSION AND CONCLUSIONS	135
12.1 Summary of the Results	135
12.2 Conclusions	135
12.3 Further Research	137
References	139

Part VI Appendix

13. APPENDIX	143
13.1 Definitions	143
13.2 Example Queries	145
13.3 DTD	146
13.3.1 Medline DTD	146
13.3.2 CONAN DTD	149
13.3.3 Regular Expressions used in Protein-Protein Interaction Extraction	150
13.3.4 Databases used in BLAST search	153
Index	155

List of Figures

5.1	Screenshot of the MeSH browser	28
5.2	Screenshot of the Ensembl website	30
5.3	Screenshot of Gene Ontology	32
5.4	Screenshot of UMLS Metathesaurus	34
5.5	Screenshot of SwissProt	35
5.6	Screenshot of the IPI service	38
7.1	Support Vector Machine	55
7.2	Overview of the Boosting classifier	57
8.1	Flow Diagram of CONAN	66
9.1	Definition of Recall and Precision	82
9.2	Recall-Precision Graph	83
11.1	CONAN Website Layout	112
11.2	CONAN Input Form	113
11.3	CONAN Webserver Communication	114
11.4	CONAN Result: PMID Search	115
11.5	PMID List	118
11.6	Interaction / All	120
11.7	Interaction / PMID	121
11.8	Gene Interaction Graph	123
11.9	www.genenetwork.nl	126

List of Tables

6.1	Sample of Databases used in keyword search	40
6.2	Translational Table used in the BLAST search	41
6.3	Parameters used in BLAST-search	42
9.1	Example for Inter-Annotator Agreement	84
9.2	Guideline for the Strength of the <i>kappa</i> -value	85
10.1	Prodisen corpus in numbers	91
10.2	PubMed article usage overlap between different biological databases.	92
10.3	Agreement indices of the polytomous ratings of the two Prodisen data sets.	92
10.4	Overview of Results on BioCreative Corpus	94
10.5	Quartile Segments in BioCreative	94
10.6	Overview of Results on YAPEX-Corpus-SLOPPY	95
10.7	Overview of Results on YAPEX-Corpus-STRICT	95
10.8	Overview of Results on LLL-Challenge Corpus as reported in [14]	96
10.9	Inter-Annotator Agreement Results	98
10.10	Evaluation of CONAN on LDD Corpus	99
11.1	Accuracy of the Gene Interaction Networks	125
11.2	Likelihood Ratios for PubMed Bins	128
11.3	Extracted Interactions	129
13.1	Databases used in BLAST search	153

Acknowledgments

As the Acknowledgments of every Ph.D. thesis start with some self-pity, here is mine: Moving to a different country at a relatively young age is not easy. Conducting research and writing a Ph.D. thesis is definitely even harder. So here's to the people without whom this thesis would not have been written.

First and foremost, I want to thank my supervisor Arno Siebes for the courage of employing an unknown Austrian biologist as a Ph.D. student in computer science. Without his enthusiasm and his support, nothing could have been done. Even when times were rough, he always understood to boost my confidence.

Secondly, I want to thank my family back in Austria who (although never quite understanding what I'm doing exactly), always supported me in every situation possible.

Thirdly, the LDD (Large and Distributed Databases) group at the University of Utrecht became sort of a second family for me over time. My thanks go to Ad Feelders, Hans Phillipi and Lennart Herlaar for being (most of the time) the best colleagues one can ask for. Carsten Riggelsen, Ronnie Bathoorn and Jeroen de Knijf, my fellow Ph.D. students, always shared my vision of supporting each other and discussing (not only) research. A special thanks goes out to Arno Knobbe, who has been invaluable in giving tips and hints how to make a good thesis. Last, but not least, the three "Benjamins" of our group, Matthijs, Arne and Jilles, who had not only the pleasure to share a room with me for the last weeks/months, but also had the (arguable) pleasure of being the first ones to proof-read this thesis.

Special thanks go out to the people without whom this research would have been impossible: Edwin Cuppen, Victor Guryev, Lude Franke, Philip Lijnzaad and coworkers. Edwin has been supportive of my project from the start, even when it did not really fit into the project description. Victor has been a good source of advice and helped a great deal in implementing and setting up the CONAN webserver. Lude Franke, also a fellow Ph.D. student, has been a

great source of discussions and ideas and has helped me not only throughout this theses. Philip Lijnzaad and coworkers from the UMC Utrecht have to be thanked because they let me use their CPU power without any problems whatsoever.

A greatly appreciated effort was done by Martin Krallinger (Madrid), who is a fellow Text Miner. He always provided me with news about papers and conferences and always was ready to start a new project or a discussion.

What would be an Acknowledgments section without the shout-outs to friends. Big thanks to Ercole, Marcella, Nathalie and Robert for showing me some good times in Utrecht. Christof van Nimwegen, Marco Wiering and Wim de Jonge also showed me that Dutch people are not that bad after all ;-)

Finally, my friends back in Austria always were of big support: Jochen, Gregor, Michl, Iris, Opa, Michelle, Georgia, Petra, Mike, Hupsi, Thomas and Claus. All the trips back to the mothership were invaluable and helped me a great deal in retaining my sanity.

This study was financially supported by the Dutch Ministry of Economic Affairs through the Innovation Oriented Research Program (IOP) on Genomics, grant IGE01017.

PART I

INTRODUCTION

Chapter 1

INTRODUCTION

This thesis is concerned with text mining; extracting useful information from text data. When conducting research, there are always many problems to think about. Research problems can be categorized in two different fields: World Problems and Knowledge Problems [27].

World Problems consist of a difference between how the world is and the way we think it should be. We solve world problems by changing the state of the world itself.

Knowledge Problems are considered to be caused by lack of knowledge of the world. When solving a knowledge problem, we try to change the state of our knowledge and not try to change the world itself. We want to solve the problem by following a sound research method.

When we keep the definition of those problems, we can formulate research questions specific to text and literature mining.

Over the last decades, the use of large-scale experimental techniques has led to an increased pace at which scientific information is produced. Hence, the biomedical text presenting this information and the text databases storing this information, namely PubMed/MEDLINE, are growing at an equal rate. This often quoted fact [21] poses a big problem for every experimentalist. Interesting and useful information, like interaction data and mutation data, could appear in papers they have not read. In this way, important facts might get overlooked and scientific work might be affected.

So we can specify the definition of our world problem and knowledge problem in the following way:

The world problem is that there are too many scientific papers published in the biomedical field at the moment and a researcher has to read too many papers to perform quality research. At another level, there is a problem that

biological entity names (e.g. protein names) are not unified in the community. Later on, we will see how this affects literature and text mining.

The knowledge problem we are confronted with is that important information is often not clearly visible and biological information might be completely lost in the text. Although there are many other ways to gather information (conferences, books, etc.), scientific publications and papers are still the foremost way of getting the newest information available in the field. When not having enough knowledge about the world, experimental biological and medical research might be hindered or even made impossible.

It has to be said that regarding these problems, research can only solve the knowledge problems. It will take many years to solve the world problems that were stated above. From my own point of view, things like the entity nomenclature and the highlighting or tagging of biomedical entities in text could be achieved in the course of time, but an enormous effort has to be taken by publishers, editors, reviewers and authors to solve those problems in the future.

In other words, the research question that is fundamental to this thesis is

Is it possible to construct a system which is suited to extract hidden information out of text while being as complete as possible?

In this part of the thesis, I firstly describe which text mining systems exist and also give examples of the different text mining categories. Secondly, I give an overview of what the main problems in text mining are at the moment. Finally, I provide a solution to these problems: CONAN, a system developed by me, which forms the main part of this thesis. Moreover, I give an overview of the whole thesis.

Chapter 2

EXISTING TEXT MINING SYSTEMS

In this chapter, I give an overview of the methods used in text mining and information extraction in the biomedical field nowadays and also what the problems with these systems are.

In the last years, several text analysis systems and algorithms have been developed for the biomedical community. Although the principal goal of each of those services is to serve the biological community with information, the way how information is extracted and presented and the type of information is quite different from system to system.

As Martin Krallinger writes in his review of Text mining and Natural Language Processing (NLP) services [14], we can distinguish several different types of Text Mining/NLP systems with regard to what information is extracted. These types include Information Retrieval (IR), Information Extraction (IE) and Knowledge Discovery (KD). More or less the same structure is given in the review of Lars Juhl Jensen [10].

2.1. Information Retrieval (IR)

In IR, relevant articles have to be retrieved from large collections of data. This form of analysis is also known as Article Retrieval. IR in the biomedical field wants to provide a Google-like service for biomedical articles. The user queries the database either with a set of keywords or with a document. Interesting articles that contain the keywords or articles which are similar to the given article are retrieved by the system. Although many services (like Entrez [23, 26]) are already heavily used by scientists, they need lots of database and program updates to keep the content up-to-date. IR methods are not the same as text mining methods, although they share the same tools and techniques, Natural Language Processing (NLP) being one of the most important.

2.2. Natural Language Processing

Natural Language Processing (NLP) is a range of computational techniques for analyzing and representing naturally occurring text (free text) at one or more levels of linguistic analysis (e.g., morphological, syntactical, semantical, pragmatic). The ultimate goal is to achieve human-like language processing for knowledge-intensive applications. This goal is still far from reached, the higher the level of analysis, the more difficult the problem is. Moreover, the different levels of analysis are not disjunct. For instance, semantics plays an important role in the syntactic analysis. NLP is a subfield of artificial intelligence and linguistics. In IR, NLP is often used as a pre-processing step. When a system wants to find the most important information in text and then wants to retrieve the information found, it first has to define the most important parts. The two primary aspects of natural language are syntax (or grammar) and the lexicon.

Syntax, or the patterns of language, defines structures such as the sentence (S) made up of noun phrases (NPs) and verb phrases (VPs). These structures include a variety of modifiers such as adjectives, adverbs and prepositional phrases.

A noun phrase consists of a pronoun or a noun with any associated modifiers, including adjectives, adjective phrases, adjective clauses, and other nouns in the possessive case. A verb phrase consists of a verb, its direct and/or indirect objects, and any adverb, adverb phrases, or adverb clauses which happen to modify it. An example for a noun phrase is:

the membrane-bound protein

In this phrase, “protein” is the noun and “membrane-bound” is the adjective describing the noun. An example for a verb phrase is:

is ubiquitously expressed

In this phrase, “is expressed” is the verb and “ubiquitously” is the adverb connected to it.

A **lexicon** is a machine-readable dictionary which may contain a good deal of additional information about the properties of the words, notated in a form that parsers can utilize. It shows what terminal symbol a word in the language belongs to e.g. eat = verb, duck = noun and duck = verb.

The determination of the syntactical structure of a sentence is done by a parser. A parser is an algorithm that uses the grammar and lexicon to find the structure in a language fragment (usually a sentence). The input would

be the sentence (for example) and the output would be some representation of the structure.

Modern parsers perform reasonably well in determining the syntactic structure of a sentence. Unfortunately, in any real sentence there are notorious ambiguity problems, often caused by the fact that a word can have different meanings and syntactic roles. There are two main kinds of ambiguity:

- Global ambiguity: the whole sentence can have more than 1 interpretation.
- Local ambiguity: part of a sentence can have more than 1 interpretation.

Consider the simple sentence,

"Voltage-gated sodium and potassium channels are involved in the generation of action potentials in neurons." (Science, v219, p1337, Human Genome issue).

To a biologist, this sentence is clear and unambiguous. It means that both sodium and potassium channels are voltage-gated, meaning that they are activated by the surrounding electric potential difference near the channel. A parser faces many difficulties when analyzing the sentence. For example, a parser may group the constituents to form the noun phrase "Voltage-gated sodium", when it is the channels that are voltage-gated. There is also the issue of whether there are single channels for both sodium and potassium or separate channels for the two ions - the structure of the English in the sentence leaves this open. These ambiguities are classic ones in parsing and there are no simple ways to resolve them on the basis of sentence syntax alone. In biomedical text analysis, and especially in CONAN, these ambiguities do not pose a big problem, because mostly simple noun phrases have to be extracted.

It is important to note that text mining, IR and NLP are different fields. Sophisticated NLP techniques are frequently used in IR to represent the content of text in an exact way (e.g. noun and verb phrases being the most important ones), extracting the main points of interest, depending on the domain of the IR service. However, NLP is not only used in parsing the documents, but also for handling the user queries. The important information has to be parsed from the user queries in a similar way. NLP techniques are used in almost every aspect of the text mining process, namely in Named Entity Recognition (see Section 2.4), Information Extraction (see Section 2.5) and Knowledge Discovery (see Section 2.7).

2.3. SDI Services

SDI services (selective dissemination information services, like Pubcrawler ([9, 24])) are related to IR services. They retrieve relevant articles and notify the user when these articles are available. The big advantage of SDI systems is that they are fully automated and the user only has to specify the area

of interest once. These SDI Services can be seen as a “news service” for the subscriber.

2.4. Biological Named Entity Recognition (NER)

NER (named entity recognition) describes the identification of entities in free text. Entities in the biomedical domain include genes, protein names and drugs. NER is the most common form of text analysis in the biomedical domain. Over 50 different information extraction and text mining tools have been developed in recent years for this specific task (e.g. AbGene[25], NLPProt [17] and GAPSCORE[1]). NER often forms the starting point of a text mining system, meaning that when the correct entities are identified, the search for patterns or relations between entities can begin. As Krallinger describes in his review, NER tools normally reach a level of accuracy which is about 80%, whereas similar tools for other domains, e.g. economy, reach a much higher accuracy. This points out that protein names are of a more complex nature than “normal” free text. The reasons for this lack of accuracy are explained below.

2.4.1 Problems in NER

As mentioned above, NER is often the starting point for text mining systems. Hence, its performance is critical for these text mining systems. However, there are three major problems in NER which form big difficulties in the process. These problems are very specific for the biomedical domain.

- Anaphora. Anaphora are by definition instances of an expression referring to another. This can best be explained by an example:

Sentence 1: *CasL/HEF1* belongs to the p130(Cas) family.

Sentence 2: *It* is tyrosine-phosphorylated following beta(1) integrin and/or T cell receptor stimulation and is thus considered to be important for immunological reactions.

The “It” in Sentence 2 refers to “CasL/HEF1” in Sentence 1. This structure is often seen in biomedical abstracts, especially when a new protein is characterized. This problem is not only a problem for NER methods, but also subsequently for Information Extraction (IE) methods, where relationships (e.g. protein-protein Interactions) between protein names are extracted.

Few systems attempt to resolve anaphoric relationships, so most systems are therefore unable to extract relationships that span multiple sentences.

This is not as big a limitation as it might seem, because most relationships are normally mentioned within a single sentence.

- **Ambiguous Protein Names.** The ambiguity problem occurs when one name refers to different entities, meaning that one protein symbol (e.g. VIP) refers to multiple gene products (e.g. vasoactive intestinal peptide and alpha-2 macroglobulin family protein VIP).

Liu et. al [15] report that ambiguity often occurs between species but also in one species. In an experiment, they show that the intra-species ambiguity is only 0.02%, but inter-species ambiguity can be as high as 14.2%. Another interesting statistic is that only 17.7% of protein names used in abstracts are the official protein names, 7.6% were the full names and 74.7% were gene synonyms.

Another problem in ambiguity is that gene/protein names often resemble “normal” English words. For example, the English words “was” and “if” occur, of course, in almost every publication. However, they are also the names of mouse genes. The same is true for drosophila genes like “kruppel” or “dachshund” which are also “normal” English words.

Although some strategies to resolve this ambiguity have been proposed (see also [15]), it still remains part of the “world problem” described in the first section of this thesis. Inter-species ambiguity can, however, be resolved by mining not only for protein names but also for organism names, which is performed by systems like NLProt [17].

- **Partial Matches.** In text mining and especially in NER, we can distinguish between full matches and partial matches. This is best explained by an example. The protein “protein kinase C”, or short “PKC”, transduces the cellular signals that promote lipid hydrolysis. In text mining, protein names like that are very hard to understand and to extract. The reason is simple: “Protein kinase” as such is a protein name on its own, while “Protein kinase C” would be the correct protein name in this case. So the question is if “Protein Kinase” should be counted as a True Positive (TP) when evaluating a NER method or not. In recent publications, scientists delivered two different types of evaluation, one being the so-called “SLOPPY”-mode, where partial protein matches (e.g. “ Protein Kinase”) are considered to be TP. The other is called “STRICT”-mode, where only the whole correct protein (e.g. “Protein kinase C”) name is considered to be a TP. The partial protein names might irritate or mislead the user when querying the extracted data. This problem is not easily solved and poses a complication in the evaluation of methods.

2.5. Information Extraction (IE) and Text Mining

2.5.1 Systems

Information Extraction (IE) in the biomedical domain is the extraction of associations between biological entities in text. The most interesting information that can be extracted is either Protein-Protein Interactions (PPIs) or functional protein annotation. This field is very diverse, reaching from extraction of kinase pathways to extracting SwissProt keywords for functional annotation.

In IE two sub-fields can be distinguished: Co-occurrence and NLP. In Co-occurrence, relationships of entities are identified when they co-occur within the same abstract or sentence. With sophisticated frequency-based scoring schemes, these systems can rank extracted relationships. NLP methods combine the analysis of syntax and semantics. When applied in IE, they extract the noun phrases in a sentence and represent their interrelationship. NER methods are used subsequently to semantically label the relevant biological entities. Finally, a rule set is used to extract relationships on the basis of the syntax and the semantic labels. Normally, this is done via a test and a training-set. Co-occurrence methods (PreBIND[4], iHOP[7] and PubGene[11]) tend to give better recall, but worse precision than NLP methods (MedScan, MedLEE and GeneWays[3, 6, 22]). Precision and recall are extensively described in Chapter 9. A negative aspect of these methods is the large number of rules that have to be used to extract relationships. Moreover, parsers are usually written for “normal” English text and not for text in the biomedical domain. In this very recent field, efforts have been made to construct ontologies, dictionaries and functional keywords which define relevant biological aspects of proteins. NLP is most prominent in the organization of events like BioCreative and TREC, which are important for comparative analysis of evaluation results.

Although some experts consider the finding of “novel and new” information as the only real text mining (see Section 2.5.2), others group IE methods and text mining together in one group. For clarity, I introduce a definition of Text Mining in the next section.

2.5.2 Text Mining

Text mining, in one definition, is the in-silico discovery of new, previously unknown information, by automatically extracting information from one or more written resources. Text mining is a new and exciting research area that tries to solve the information overload problem by using techniques from data mining, machine learning, Natural Language Processing and Information Retrieval.

Text mining is a variation on a field called data mining, that tries to find interesting patterns and relationships in large databases. A typical example in

data mining is using consumer purchasing patterns to predict which products to place close together on shelves. For example, if you buy diapers, you are likely to buy beer along with it.

The difference between regular data mining and text mining is that in text mining the patterns are extracted from natural language text rather than from structured databases of facts. It has to be said that the boundaries between text mining and data mining are fuzzy.

Text mining occurs, of course, not only in the biomedical domain which is the focus of this thesis, but in other domains as well. In the upcoming TREC (trec.nist.gov) conference, a conference concerned with evaluating all different kinds of text mining tools, several fields are distinguished (see Section 10.2.8.2 for details). The definition of text mining in the biomedical domain is quite vague. Some people define text mining as searching the literature for overlooked connections and interpreting the results to obtain “novel” facts that cannot be derived by just reading the text. In this definition, literature mining is defined as the general term for Information Extraction (IE) from text.

In some publications, literature mining and text mining are equivalent terms, meaning that text mining is the science of extracting specific associations, such as protein-protein interactions and protein functions from text. Following this definition, text mining is equivalent to IE.

In this thesis, literature mining and text mining will be both used in the same way, following the latter definition. The text mining from the first definition, meaning that “novel” facts (that cannot be derived from just reading the text) should occur in the results, is called Knowledge Discovery (KD) in this thesis. This is done because text mining is related to data mining, and data mining is a sub-field of Knowledge Discovery in Databases (KDD) [5]. The Knowledge discovery process includes producing the raw results by data mining and accurately transforming them into useful and “novel” information. The same is true for text mining and KD. Therefore also text mining should be distinguished from Knowledge Discovery (KD), as it is a sub-field of KD.

2.6. Microarray Analysis

Lately, methods have been developed which link results of microarray experiments to biomedical information found in text databases. Either single genes or groups of genes can be annotated, the text can be mined for functional terms which are associated with the gene or gene groups. Although these systems (microGENIE [13] and a yet unnamed system[20]) can link functional information to the entities of interest, they cannot provide automated summaries of biologically relevant information yet.

2.7. Knowledge Discovery

This type of system focuses on the construction of networks and interactions to discover new relationships. These relationships, which most of the times include either diseases or chemical substances, are found by discovering indirect relationships between entities that are not directly connected in text. First attempts tried to do that via MeSH terms([2, 16]). Some scientists still consider only this set of methods as real “text mining”. One application which can be grouped in the KD process will be presented in Section 11.4.

2.8. Conclusions

So, we can come to the conclusion that the field of text analysis consists of many different sub-fields, that all require different approaches. Nevertheless, there are some problems in text and literature analysis systems that are universal to all approaches. These problems are described in-detail in the following chapter.

Chapter 3

PROBLEMS WITH EXISTING SYSTEMS

In this section, I describe the most common problems of modern text mining systems. Some of these problems are specific for the biomedical domain, while others are general problems which appear in every text mining application. Some of the problems are solved by CONAN, the system we created and which I introduce later in this thesis (see Chapter 4).

- 1 Small Datasets. This was particularly a problem in the early stages of this (relatively) young field. Researchers either did not have the resources to handle lots of data (limited CPU power and memory) or wanted to concentrate their research on a very limited set of molecules. Also, the number of resources the researchers could rely upon was very small in the early days (e.g. pre-genomics era). A good example is the Kinase Pathway DB [12], which is a high quality database, but solely focuses on protein kinases.
- 2 Restricted List of Organisms. An additional problem is that some systems exclusively handle certain organisms and do not generate data which is valid for several species. A very important feature of data derived from text, whether PPIs or protein annotation, is cross-referencing and cross-validation over multiple species. Information about homologous proteins can help experimentalists in their research and can provide more insight into the way proteins work. One example for such a restricted list of organisms is the “Textpresso”-system [18], which focuses only on *C.elegans* data.
- 3 Incomplete Sets of MEDLINE. Systems that were developed over the last years often limit their data extraction to the newest articles or to a specific subset of articles. Experimentalists, however, want to get their information as complete as possible. When limiting a search to a subset of articles, im-

portant facts might get overlooked. When inferring relationships between proteins, “older” information is as useful as “new” information. PPIs have been established and published from the middle of the 1990’s, so the information wanted might be in an article that most systems do not process. This problem is common in all text mining systems. Some systems like iHOP [8] are trying to be complete, other systems on the other hand only use a very small subset of MEDLINE and stop there.

- 4 Achieving High Precision and High Recall Rates. It is a main goal of all IE systems to obtain high recall and precision rates (for definition see Chapter 9). Scientists and especially experimentalists do not want to work with data that is neither complete nor precise enough. Often text mining systems have the problem that one measure is much higher than the other (depending on the system), which leads to data that either is not representative for the text processed or has a very high amount of either false positives (FP) or false negatives (FN). The main goal of a text mining system is of course to keep the numbers of FP and FN as small as possible. Common solutions are extra filtering steps or by manual post-processing of the data. Nowadays, the F-measure (also see Section 9.2) is used. This was introduced because systems often have either a high recall or a high precision. The F-measure, which is the harmonic mean of those two, gives a nice balance of precision and recall in a system so that other scientists can see rather quickly whether a system provides good results or not. One good example is a group that took part in the LLL challenge (also see Section 10.2.5). It is reported ([19]) that one (unnamed) group achieved a recall of 98.1%, which is really high. However, their precision was only 10.6%, resulting in an F-measure of only 19.1%. It is crucial for text mining systems to obtain both high recall and high precision.

Chapter 4

RESULTS OF THIS THESIS

In Chapter 3, the most emergent problems in biomedical text mining systems have been identified. We can now specify the research problem and raise the following technical challenges:

- 1 Does the combination of text-mining classifiers and algorithm increase the performance?*
- 2 Can such a system still reach high precision and recall rates, resulting in a good F-measure?*
- 3 Is it possible to include the information extracted by such a system in other systems?*
- 4 Can the extracted information be presented in such a way that the information is helpful for experimentalists?*

After the problems with state-of-the-art text mining Systems have been identified and the technical problems have been identified as well, a solution is provided: **CONAN**, a text mining system that was developed by the author and which forms the main part of this thesis. CONAN is developed as a full-scale approach that will ultimately cover all of PubMed/MEDLINE. An in-depth description of CONAN can be found in Chapter 8 of this thesis.

The novelty that CONAN brings to this field and its added value can be summarized in two main points:

- **Combining Data Sources**

While all systems mentioned above are relying on a single data source, CONAN is designed to integrate as many data sources as possible. This is done to guarantee the user a large amount of data that is also cross-linked

between several data resources and databases. This ensures that CONAN users are provided complete overviews of the information included in text. Moreover, CONAN is designed in such a way that the addition of new data sources is very easy and quick and the integration of CONAN data into other systems is easy in equal measure. This directly solves the third challenge posed above.

■ Combining Techniques

The main goal of most researchers in the text mining area is to produce new techniques or new algorithms. In contrast to these methods, CONAN combines already published and well-known techniques and classifiers. So it is ensured that the best techniques and their outcome are combined to achieve the best possible scores and results. As I will show later on in this thesis, especially in Section 10.2, the combination of classifiers definitely improves the performance of the system, directly addressing challenge one. Similar to the previous point about combining data sources, the addition of new techniques and algorithms to the system is very easy.

When looking at the two points above, it becomes clear that by combining many data sources and techniques, the user is presented with lots of information. This information is presented in an easy and understandable way in CONAN, solving challenge four of the technical challenges above. The means how this data is presented to the user can be seen in Chapter 11. Moreover, as will be presented in Part IV of this thesis, the information is of very high quality, solving challenges one and two.

However, there are some aims that CONAN, while being a complete system, cannot achieve. When a user is searching for specific information, the amount of results that is returned might still be very big and the user has to filter out the significant information manually. Moreover, CONAN is still under construction, meaning that not all of MEDLINE has been processed so far.

4.1. Overview of this thesis

In this section, I give an outlook over the whole thesis. The thesis is constructed of six parts. The first part is the Introduction, which you are currently reading.

The second part describes the several components that were used in building CONAN and is named “Components”. It consists of four distinct chapters: Firstly, the biological databases used in constructing CONAN are introduced. Secondly, the program components of CONAN are explained. Finally, I describe the technical components used in the construction of CONAN.

The third part is about CONAN itself. Here it is explained how CONAN works and how the different parts described in the “Components” part work together to produce output.

The fourth part describes the Results obtained by CONAN. The first chapter of this part describes the most frequently used measures in text mining, namely precision, recall and the F-Measure. Moreover, the Inter-annotator agreement scores, which are important when constructing a corpus, are introduced. The second chapter of this part describes the experiments that were performed with CONAN. This chapter includes all evaluations performed on different corpora (YAPEX, BioCreative, LLL and LDD). Not only CONAN, the system, itself is evaluated, but also the Boosting classifier which was constructed as a post-processing step (see Section 7.4.2.1). Moreover, the chapter includes how such a corpus is constructed with two examples, the Prodisen corpus and the LDD corpus. The final section of this chapter is concerned with interesting results that were produced by CONAN. There we give an overview on why CONAN is so useful for biologists. The third chapter of this part is about the Applications that were implemented using CONAN. Three main applications can be found: the CONAN command line interface, the CONAN webserver and the application of CONAN in constructing gene interaction networks.

The fifth part contains the Discussion and an outlook on future work. I also give an overview in which applications CONAN could be a part of in the future. The Appendix, which forms the sixth part, gives more information on specific concepts explained in the thesis. In the main text, there is always a reference to the Appendix where applicable. Moreover, the Appendix includes a Glossary of terms that are frequently used in this thesis.

At this point, I want to point out a very important aspect of this thesis. This thesis is written for biologists as well as for computer scientists. Some concepts introduced might seem trivial for one group, but will be appreciated by the other group for the sake of clarity. Please note that, as gene and protein names cannot automatically be distinguished from each other in text, the term Gene Name / Protein Name is interchangeable in this thesis.

References

- [1] J.T. Chang, H. Schutze, and R.B. Altman. GAPSCORE: finding gene and protein names one word at a time. *Bioinformatics.*, 20(2):216–225, 2004.
- [2] D. Chaussabel and A. Sher. Mining microarray expression data by literature profiling. *Genome Biol*, 3(10):RESEARCH0055, 2002.
- [3] N. Daraselia, A. Yuryev, S. Egorov, S. Novichkova, A. Nikitin, and I. Mazo. Extracting human protein interactions from MEDLINE using a full-sentence parser. *Bioinformatics.*, 20(5):604–611, 2004.
- [4] I. Donaldson, J. Martin, B. de Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G.D. Bader, K. Michalickova, T. Pawson, and C.W. Hogue. PreBIND and Textomy—mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics.*, 4:11, 2003.
- [5] U.M. Fayyad, G. Piatetsky-Shapiro, and P.Smyth. *From Data Mining To Knowledge Discovery: An Overview*. AAAI Press/The MIT Press, 1996.
- [6] C. Friedman. A broad-coverage natural language processing system. *Proc AMIA Symp*, pages 270–274, 2000.
- [7] R. Hoffmann and A. Valencia. Protein interaction: same network, different hubs. *Trends Genet.*, 19(12):681–683, 2003.
- [8] R. Hoffmann and A. Valencia. Implementing the iHOP concept for navigation of biomedical literature. *Bioinformatics*, 21(Suppl2):ii252–ii258, 2005.
- [9] K. Hokamp and K. H. Wolfe. Pubcrawler: keeping up comfortably with pubmed and genbank. *Nucleic Acids Res.*, 32(Web Server issue):W16–9., Jul 2004.
- [10] L. J. Jensen, J. Saric, and P. Bork. Literature mining for the biologist: from information retrieval to biological discovery. *Nat Rev Genet.*, 7(2):119–129, 2006.
- [11] T. K. Jenssen, A. Laegreid, J. Komorowski, and E. Hovig. A literature network of human genes for high-throughput analysis of gene expression. *Nat Genet.*, 28(1):21–8., May 2001.

- [12] A. Koike, Y. Kobayashi, and T. Takagi. Kinase pathway database: an integrated protein-kinase and-based protein-interaction resource. *Genome Res.*, 13(6A):1231–1243., 2003.
- [13] M. Korotkiy, R. Middelburg, H. Dekker, van F. Harmelen, and J. Lankelma. A tool for gene expression based pubmed search through combining data sources. *Bioinformatics.*, 20(12):1980–2. Epub 2004 Mar 25., Aug 12 2004.
- [14] M. Krallinger and A. Valencia. Text-mining and information-retrieval services for molecular biology. *Genome Biol.*, 6(7):224, 2005.
- [15] H. Liu, S.B. Johnson, and C. Friedman. Automatic resolution of ambiguous terms based on machine learning and conceptual relations in the UMLS. *J Am Med Inform Assoc.*, 9(6):621–636, 2002.
- [16] D. R. Masys, J. B. Welsh, Lynn J. Fink, M. Gribskov, I. Klacansky, and J. Corbeil. Use of keyword hierarchies to interpret gene expression patterns. *Bioinformatics.*, 17(4):319–326, Apr 2001.
- [17] S. Mika and B. Rost. Protein names precisely peeled off free text. *Bioinformatics.*, 20(Suppl 1):I241–I247, 2004.
- [18] H.M. Muller, E.E. Kenny, and P.W. Sternberg. Textpresso: an ontology-based information retrieval and extraction system for biological literature. *PLoS Biol.*, 2(11):e309, 2004.
- [19] C. Nédellec. Learning language in logic - genic interaction extraction challenge. In *Learning Language in Logic Workshop (LLL'05) at ICML 2005*, 2005.
- [20] J.C. Oliveros, C. Blaschke, J. Herrero, J. Dopazo, and A. Valencia. Expression profiles and biological function. *Genome Inform Ser Workshop Genome Inform.*, 11:106–117, 2000.
- [21] D. Rebholz-Schuhmann, H. Kirsch, and F. Couto. Facts from text—is text mining ready to deliver? *PLoS Biol.*, 3(2):e65, 2005.
- [22] A. Rzhetsky, I. Iossifov, T. Koike, M. Krauthammer, P. Kra, M. Morris, H. Yu, P. A. Duboue, W. Weng, W. J. Wilbur, V. Hatzivassiloglou, and C. Friedman. Geneways: a system for extracting, analyzing, visualizing, and integrating molecular pathway data. *J Biomed Inform.*, 37(1):43–53., Feb 2004.
- [23] G.D. Schuler, J.A. Epstein, H. Ohkawa, and J.A. Kans. Entrez: molecular biology database and retrieval system. *Methods Enzymol.*, 266:141–162, 1996.
- [24] M. Shultz and De S. L. Groote. Medline sdi services: how do they compare? *J Med Libr Assoc.*, 91(4):460–7., Oct 2003.
- [25] L. Tanabe and W.J. Wilbur. Tagging gene and protein names in biomedical text. *Bioinformatics.*, 18(8):1124–1132, 2002.
- [26] D.L. Wheeler, D.M. Church, A.E. Lash, D.D. Leipe, T.L. Madden, J.U. Pontius, G.D. Schuler, L.M. Schriml, T.A. Tatusova, L. Wagner, and B.A. Rapp. Database resources of the National Center for Biotechnology Information: 2002 update. *Nucleic Acids Res.*, 30:13–16, 2002.

- [27] R. Wieringa. *Requirements Engineering: Frameworks for Understanding*. John Wiley & Sons., 1996.

PART II

COMPONENTS

In the course of the development of CONAN, which this thesis is based on, several methods for protein tagging and interaction extraction have been implemented. Moreover, several biological databases are used in CONAN. The following chapters are dedicated to an in-depth description of those parts of the system. In most cases, please refer to the original publication for a complete analysis, although I will explain how the parts work as complete as possible.

Firstly, I give an overview of the biological databases used in CONAN, explaining which data is contained in these databases. Secondly, I explain how the different parts of CONAN work, and in which text mining categories (e.g. IR, IE, NER) these components can be categorized. Finally, I give an overview of which technical bits and pieces were used in the construction of CONAN.

Chapter 5

BIOLOGICAL DATABASES

5.1. PubMed/MEDLINE

The most important database for CONAN is MEDLINE, as it contains the bibliographical information needed to perform text mining research. MEDLINE (Medical Literature Analysis and Retrieval System Online) is the U.S. National Library of Medicine's (NLM) premier bibliographic database that contains approximately 13 million references to journal articles in life sciences with a concentration on biomedicine.

5.1.1 PubMed

MEDLINE is the largest component of PubMed, the U.S. National Library of Medicine's (NLM) database of biomedical citations and abstracts that is freely searchable on the Web (<http://pubmed.gov>). MEDLINE covers over 4,800 journals published in the United States and more than 70 other countries primarily from 1966 to the present. MEDLINE includes references to articles indexed with terms from NLM's controlled vocabulary, MeSH. Citations in MEDLINE are from journals selected for inclusion in the database.

In addition to MEDLINE citations, PubMed also contains

- OLDMEDLINE for pre-1966 citations
- Citations to articles that are out-of-scope (e.g., covering plate tectonics or astrophysics) from certain MEDLINE journals, primarily general science and general chemistry journals, for which the life sciences articles are indexed for MEDLINE
- In-process citations which provide a record for an article before it is indexed with MeSH and added to MEDLINE or converted to out-of-scope status. Citations that precede the date that a journal was selected for MEDLINE indexing (when supplied electronically by the publisher).

- Some life science journals that submit full text to PubMedCentral and may not have been recommended for inclusion in MEDLINE although they have undergone a review by NLM, and some physics journals that were part of a prototype PubMed in the early to mid-1990's.

MEDLINE is the main source of input for CONAN. It is used in CONAN as the starting point, as it contains the abstracts of the articles that we want to analyze using CONAN. One of the features inside the MEDLINE service which is heavily used are the MeSH terms.

5.1.2 MeSH

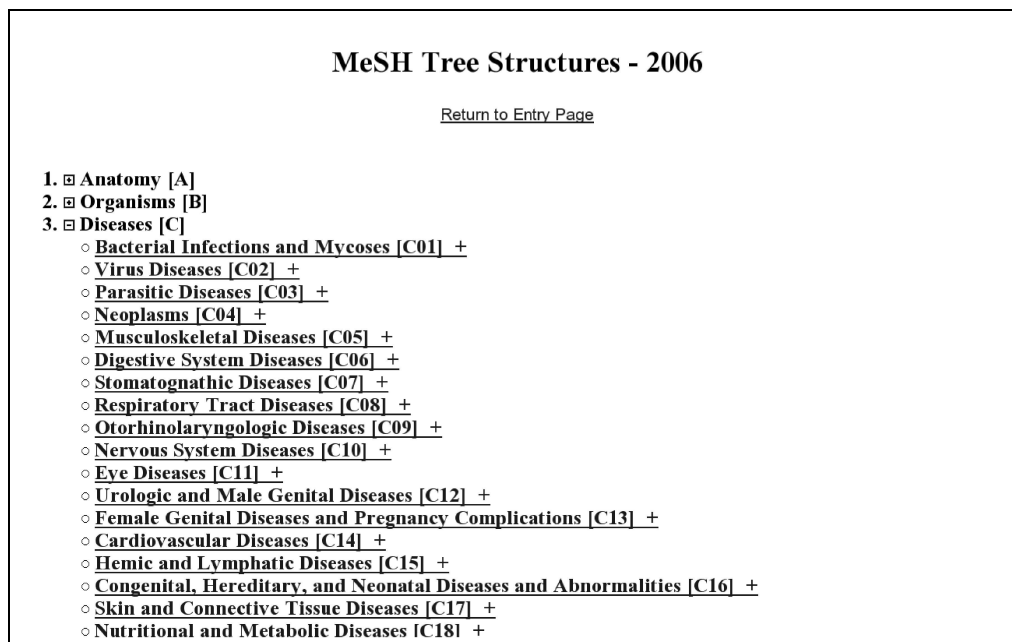


Figure 5.1. Screenshot of the MeSH browser

MeSH (short for Medical Subject Headings), is a service provided by the National Library of Medicine (NLM). It is a controlled vocabulary, consisting of a set of terms (=descriptors) in a hierarchical structure. Searching is possible in all level of the hierarchy. At the upper levels of the hierarchy are very broad terms like "Anatomy" or "Mental Disorders". Specific headings are found at deeper levels of the eleven-level hierarchy, such as "Ankle" and "Conduct Disorder". Moreover, MeSH descriptors are organized in 16 categories: category A for anatomic terms, category B for organisms, C for diseases, D for drugs and chemicals, etc. One example of such a broad descriptor would be "Neoplasms"(MeSH code: [C04]). As we go further down the tree, another

MeSH term is “Neoplasms by Site” ([C04.588]), which also has sub-headings, for instance “Abdominal Neoplasms” ([C04.588.033]).

Each abstract is assigned a set of MeSH terms that describe the content of the item. There are 22,997 descriptors in MeSH. In addition to these headings, there are more than 150,000 headings called Supplementary Concept Records within a separate thesaurus. These do not belong to the controlled vocabulary as such, instead they enlarge the thesaurus and provide the user with links to the best fitting descriptor to be used in a MEDLINE search. There are also cross-references between the terms. As in the biomedical field concepts and trends are changing all the time, descriptors also are updated and changed all the time.

The MeSH terms are extracted for each PubMed ID (PMID) by CONAN. These extracted MeSH terms are used in the keyword search by CONAN as well as in the Data Integration Step (see Section 8.3.1), when the MeSH terms are queried to verify mutations in proteins.

MeSH terms are frequently used in IR systems, but only since recently they are also used for Information Extraction (IE) and Knowledge Discovery (KD) purposes. A screenshot of MeSH can be seen in Figure 5.1. In this picture, the tree-like structure of MeSH becomes visible. The category “C” (Diseases) is expanded and the different descriptors can be seen.

5.2. Ensembl

Ensembl [21] is a joint project between EMBL - European Bioinformatics Institute (EBI) and the Wellcome Trust Sanger Institute (WTSI) to develop a software system that produces and maintains automatic annotation on selected eukaryotic genomes. The Ensembl project aims to provide:

- Accurate, automatic analysis of genome data
- Analysis and annotation maintained on the current data
- Public presentation of the analysis via the Web
- Distribution of the analysis to other bioinformatics laboratories

Ensembl concentrates on vertebrate genomes, but other groups have adapted the system for use with plant and fungal genomes. The genomes that are collected at Ensembl at this point in time are:

- Human (*Homo sapiens*)
- Chimpanzee (*Pan troglodytes*)
- Rhesus monkey (*Macaca mulatta*)
- Mouse (*Mus musculus*)

Ensembl Human GeneView Search e/ Human: e.g. ENSG00000139618, ENSG00000128573

Ensembl v38 - Apr 2006

ENSG00000087088 **Ensembl Gene Report for ENSG00000087088**

- Gene information
- Gene splice site image
- Genomic sequence
- Genomic sequence alignment
- Gene variation info.
- Transcript information
- Exon information
- Protein information
- Export gene data

Gene	BAX (HGNC Symbol ID) . To view all Ensembl genes linked to the name click here . This gene is a member of the human CCDS set: CCDS12742 , CCDS12743 , CCDS12744 , CCDS12745 , CCDS12746																														
Ensembl Gene ID	ENSG00000087088																														
Genomic Location	This gene can be found on Chromosome 19 at location 54,149,929-54,156,864 . The start of this gene is located in Contig AC026803.7.1.224271 .																														
Description	Apoptosis regulator BAX, cytoplasmic isoform beta. Source: Uniprot/SWISSPROT Q07814																														
Prediction Method	Genes were annotated by the Ensembl automatic analysis pipeline using either a GeneWise/Exonerate model from a database protein or a set of aligned cDNAs followed by an ORF prediction. GeneWise/Exonerate models are further combined with available aligned cDNAs to annotate UTRs (For more information see V.Curwen et al. , Genome Res. 2004 14:942-50.)																														
Chromosome 19 54,149,929 - 54,156,864	<input type="checkbox"/> Transcripts																														
	<table border="0"> <tr> <td>ENST00000293288</td> <td>ENSP00000293288</td> <td>BAXB_HUMAN</td> <td>[Transcript info]</td> <td>[Exon info]</td> <td>[Peptide info]</td> </tr> <tr> <td>ENST00000308548</td> <td>ENSP00000309421</td> <td>BAXC_HUMAN</td> <td>[Transcript info]</td> <td>[Exon info]</td> <td>[Peptide info]</td> </tr> <tr> <td>ENST00000345358</td> <td>ENSP00000263262</td> <td>BAXA_HUMAN</td> <td>[Transcript info]</td> <td>[Exon info]</td> <td>[Peptide info]</td> </tr> <tr> <td>ENST00000354470</td> <td>ENSP00000346461</td> <td>BAX</td> <td>[Transcript info]</td> <td>[Exon info]</td> <td>[Peptide info]</td> </tr> <tr> <td>ENST00000356483</td> <td>ENSP00000348871</td> <td>NP_620119.1</td> <td>[Transcript info]</td> <td>[Exon info]</td> <td>[Peptide info]</td> </tr> </table>	ENST00000293288	ENSP00000293288	BAXB_HUMAN	[Transcript info]	[Exon info]	[Peptide info]	ENST00000308548	ENSP00000309421	BAXC_HUMAN	[Transcript info]	[Exon info]	[Peptide info]	ENST00000345358	ENSP00000263262	BAXA_HUMAN	[Transcript info]	[Exon info]	[Peptide info]	ENST00000354470	ENSP00000346461	BAX	[Transcript info]	[Exon info]	[Peptide info]	ENST00000356483	ENSP00000348871	NP_620119.1	[Transcript info]	[Exon info]	[Peptide info]
ENST00000293288	ENSP00000293288	BAXB_HUMAN	[Transcript info]	[Exon info]	[Peptide info]																										
ENST00000308548	ENSP00000309421	BAXC_HUMAN	[Transcript info]	[Exon info]	[Peptide info]																										
ENST00000345358	ENSP00000263262	BAXA_HUMAN	[Transcript info]	[Exon info]	[Peptide info]																										
ENST00000354470	ENSP00000346461	BAX	[Transcript info]	[Exon info]	[Peptide info]																										
ENST00000356483	ENSP00000348871	NP_620119.1	[Transcript info]	[Exon info]	[Peptide info]																										

Features ▼

Figure 5.2. Screenshot of the Ensembl website

- Rat (*Rattus norvegicus*)
- Domestic Dog (*Canis Familiaris*)
- Domestic Cow (*Bos Taurus*)
- African Elephant (*Loxodonta africana*)
- Gray Short-tailed Opossum (*Monodelphis domestica*)
- Chicken (*Gallus gallus*)
- Clawed Frog (*Xenoups tropicalis*)
- Zebrafish (*Denio Rerio*)

and some other non-eukaryote organisms like *Caenorhabditis elegans* (Nematode) or *Drosophila melanogaster* (Fruit fly).

The Ensembl project provides data sets resulting from an automated genome analysis and annotation process. In addition to Ensembl and EST gene sets, the project also provides other data sets on a genome-wide scale. The Ensembl project releases approximately ten updates a year. Ensembl also offers many software tools that can be used to further analyze or annotate the data.

The most important entry point to Ensembl data is the so called Ensembl Gene. For every gene name currently known, there is a unique Ensembl identifier (will be referred to as Ensembl code throughout this thesis). This code consists of the letters ENS followed by 1-4 letters, providing information about the organism the gene is derived from, and 11 numbers, identifying the gene.

As can be seen in the screenshot (Figure 5.2) for the Gene Bax (Bcl2-associated X protein), the human Ensembl code is ENSG00000087088. The mouse-homologue of this gene would be ENSMUSG00000003873 and the rat-homologue ENSRNOG00000020876. In the information about those genes, thus in the Ensembl files about the gene itself, there is much information about cross-linking the gene to other databases (RGD, UniProt/Swiss-Prot, EntrezGene, IPI, UniGene, etc.), gene ontology annotation (see Section 5.4), orthologue genes/proteins (from other organisms), protein families/motifs and all proteins encoded by this gene.

In other words, Ensembl is an important resource for biologists that gives lots of information about genes and proteins and is an ideal place to cross-link information together. This is also the reason why Ensembl identifiers are included in CONAN. Each protein name is assigned an Ensembl identifier and this identifier is used to link the information together. As experimentalists often use Ensembl, they are also used for disambiguation (see Section 2.2). While a gene or protein name is often not unique, the Ensembl identifier is.

5.3. UniProt

UniProt (Universal Protein Resource) [8] is a comprehensive catalog of information on proteins. It is a central repository of protein sequence and function created by joining the information contained in Swiss-Prot, TrEMBL, and PIR. This makes UniProt a good resource for protein information with a wide range of information offered. Through the possibility to cross-link to other databases via UniProt (see Section 5.8.2), it is a very important starting point for bioinformaticians. While Ensembl is produced automatically, UniProt is manually curated. This means that Ensembl provides information about more genes/proteins, but the annotation of UniProt is more precise.

The UniProt consortium aims to support biological research by maintaining a high quality database that serves as a stable, comprehensive, fully classified, richly and accurately annotated protein sequence knowledge base, with extensive cross-references and querying interfaces freely accessible to the scientific community.

The UniProt databases consist of three database layers:

- The UniProt Archive (UniParc) provides a non-redundant sequence collection by storing the complete body of publicly available protein sequence data.

- The UniProt Knowledgebase (UniProtKB) is the central database of protein sequences with sequence and functional annotation.
- The UniProt Reference Clusters (UniRef) databases provide non-redundant reference data collections based on UniProtKB in order to obtain complete coverage of sequence space at several resolutions.

Similar to the Ensembl identifiers, UniProt terms are used in CONAN to link information together and to assign unique terms to the Protein names.

5.4. Gene Ontology

The screenshot shows the AmiGO Gene Ontology interface. The search query is 'BAX'. The results are organized into three sections, each with a table of Gene Ontology terms, their ontologies, evidence, and references.

Term	Ontology	Evidence	Reference
ATBI-1, ARABIDOPSIS BAX INHIBITOR 1 gene from <i>Arabidopsis thaliana</i> , data from TAIR (gene:3356157)			
anti-apoptosis	P	IGI	PMID:11593047
endoplasmic reticulum	C	IDA	PMID:11852101
nuclear envelope	C	IDA	PMID:11852101
bax, bcl2-associated X protein gene from <i>Danio rerio</i> , data from ZFIN (ZDB-GENE-000511-6)			
response to methylmercury	P	IDA	PMID:15984772
Bax, Bcl2-associated X protein gene from <i>Mus musculus</i> , data from MGI (MGI:99702)			
apoptosis	P	IDA	PMID:8358790
apoptotic mitochondrial changes	P	IDA	PMID:9560217
caspase activation via cytochrome c	P	IDA	PMID:9560217
induction of apoptosis	P	IMP With MGI:1857429	PMID:11836241
induction of apoptosis by extracellular signals	P	IMP With MGI:1857429	PMID:8358790
induction of apoptosis by intracellular signals	P	IMP With MGI:1857429	PMID:12925707
negative regulation of fibroblast proliferation	P	IMP With MGI:1857429	PMID:14507967
nervous system development	P	IMP With MGI:2387665	PMID:12952892
nuclear fragmentation	P	IMP With MGI:1857429	PMID:12810599
protein insertion into mitochondrial membrane during induction of apoptosis	P	IDA	PMID:12925707
protein homooligomerization	P	IDA	PMID:12925707
protein insertion into mitochondrial membrane during induction of apoptosis	P	IMP With MGI:1857429	PMID:12925707

Figure 5.3. Screenshot of Gene Ontology

The Gene Ontology (GO) project [9] provides a controlled vocabulary to describe gene and gene product attributes in any organism. The three organizing principles of GO are:

- Molecular Function
- Biological Process
- Cellular Component

A gene product has one or more molecular functions and is used in one or more biological processes; it might be associated with one or more cellular

components. Molecular function describes activities, such as catalytic or binding activities, at the molecular level. GO molecular function terms represent activities that perform the actions. Information on where or when this activity takes place, is not given. Moreover, information about the entities (e.g. protein, gene) that perform the actions, is not given. Molecular functions generally correspond to activities that can be performed by individual genes/proteins, but some activities are performed by protein complexes. Examples of broad functional terms are catalytic activity, transporter activity, or binding; examples of narrower functional terms are adenylate cyclase activity or Toll receptor binding.

A biological process is a series of events accomplished by one or more ordered assemblies of molecular functions. It can be difficult to distinguish between a biological process and a molecular function, but it can be said that a process has to contain more than one distinct step. A biological process is not equivalent to a pathway. Examples of broad biological process terms are cellular physiological process or signal transduction. Examples of more specific terms are pyrimidine metabolism or alpha-glucoside transport.

A cellular component is a component of a cell but with the condition that it is part of some larger object, which may be an anatomical structure (e.g. rough endoplasmic reticulum or nucleus) or a gene product group (e.g. ribosome, proteasome or a protein dimer).

A screenshot of the GO consortium website can be seen in Figure 5.3. Here you can see the GO descriptions for different BAX proteins. The term is given as well as the principle (M,P,C) it is associated with. GO annotation is used in CONAN to further annotate a protein. For each protein name, the corresponding GO categories are extracted (when applicable) using the GOA files (see Section 5.8.1), providing searchable information for the user.

5.5. UMLS Services

The UMLS Metathesaurus [11] is a very large vocabulary that contains information about biomedical and health related concepts, their various names, and the relationships among them. The Metathesaurus is built from the electronic versions of many different thesauri, classifications, code sets, and lists of controlled terms used in different biomedical fields. These are referred to as the "source vocabularies" of the Metathesaurus. It contains information about more than one million biomedical concepts and five million concept names from more than 100 controlled source vocabularies and classifications. The UMLS Metathesaurus is enhanced by the UMLS Semantic Network. The Semantic Network consists of a set of broad subject categories, or Semantic Types, that provide a consistent categorization of all concepts represented in the UMLS Metathesaurus. Each concept has a Semantic Type assigned to it. For example, the concept "Acromegaly" contains a definition in the

UMLS Knowledge Source Server (UMLS)
 UMLS Version: 5.0 UMLS Release: 2002 2002AB 2002AC 2002AD 2003AA 2003AB 2003AC 2004AA 2004AB 2004AC 2005AA 2005AB 2005AC 2006AA 2006AB

Home Advanced Search Logout

Metathesaurus Semantic Network SPECIALIST Lexicon

Metathesaurus Search for: **acromegaly** in UMLS Release 2002AC

Concept
 Definition
 Synonyms
 Other Languages
 Suppressible

Synonyms
 Sources

Context
 Ancestors
 Parents
 Siblings
 Children

Relations
 Narrower
 Broader
 Similar
 Other
 Related and possibly synonymous
 Source asserted synonymy
 Allowable Subheadings
 Associated Expressions

Co-occurring Concepts
 Co-occurring

Concept: Acromegaly
 CUI: [C0001206](#)
 Semantic Type: [Disease or Syndrome](#)

Definition:
 WHAT: Acromegaly: Hypersecretion of growth hormone from a pituitary tumor resulting in insidious increase in the skeleton, soft tissues and organs. WHY: Acromegaly can result in an arthropathy with initial cartilage hypertrophy resulting in widening of the cartilage space on radiographs. The cartilage then undergoes premature osteoarthritis change with productive bony change characterized by broad distal tufts in the phalanges. ([A1/RHEUM](#))

A disorder caused by excessive secretion of SOMATOTROPIN, characterized by bony enlargement of the face (especially prognathism), hands, feet, head, and thorax. Impaired glucose tolerance; HYPERTENSION; ARTHRITIS; diffuse hyperplasia of soft tissues; CARPAL TUNNEL SYNDROME; visceromegaly; and MUSCULAR WEAKNESS are frequently associated with this condition. The most common etiology is a somatotropin secreting pituitary ADENOMA (see also PITUITARY NEOPLASMS). (From Joynt, Clinical Neurology, 1992, Ch36, pp79-80) ([MeSH](#))

Synonyms:
[Acromegaly](#)
[Acromegalia](#)
[Anterior pituitary adenoma syndrome](#)
[Eosinophilic adenoma syndrome](#)

Figure 5.4. Screenshot of UMLS Metathesaurus

Metathesaurus. The Metathesaurus also holds information about synonyms of this concept (Acromegalia, Anterior pituitary adenoma syndrome, etc.). It also shows that the Semantic Network assigns the Semantic Type “Disease or Syndrome” to the concept. This information can also be seen in Figure 5.4. The tree structure of both the UMLS Metathesaurus and the Gene Ontology reveal that both services are biological ontologies. The main difference between GO and the UMLS Metathesaurus is that GO only annotates genes and proteins, while the UMLS Metathesaurus includes more information about diseases, tissues, organisms and other concepts.

The UMLS Metathesaurus and the Semantic Network are used in CONAN for the databases used in the BLAST search (see Section 6.1), where they are used to extract keywords and protein names from text. The databases are derived from the semantic types stored in the semantic network.

5.6. iProLink

iProLink (short for integrated Protein Literature, Information and Knowledge)[20] is a resource for text mining in the area of literature-based database curation, named entity recognition, and protein ontology development. They provide users with a collection of data sources that can be used to explore text information on proteins and their features or properties. For CONAN, the dictionary of protein name abbreviations (pir.georgetown.edu/pirwww/

iprolink/Dictionary.gz) was used to train the Boosting classifier (see Section 7.4.2.1). This dictionary is derived from the iProClass integrated protein knowledgebase. The iProClass database provides information for UniProt proteins, with links to over 90 biological databases, including databases for protein families, functions and pathways, interactions, structures and structural classifications, genes and genomes, ontologies, literature, and taxonomy.

5.7. SwissProt

```

ID BAXA_HUMAN STANDARD; PRT; 192 AA.
AC Q07812;
DT 01-FEB-1995, integrated into UniProtKB/Swiss-Prot.
DT 01-FEB-1995, sequence version 1.
DT 16-MAY-2006, entry version 67.
DE Apoptosis regulator BAX, membrane isoform alpha.
GN Name=BAX;
OS Homo sapiens (Human).
OC Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
OC Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
OC Catarrhini; Hominidae; Homo.
OX NCBI_TaxID=9606;
RN [1]
RP NUCLEOTIDE SEQUENCE [MRNA].
RC TISSUE=B-cell;
RX MEDLINE=93364978; PubMed=8358790; DOI=10.1016/0092-8674(93)90509-O;
RA Oltvai Z.N., Milliman C.L., Korsmeyer S.J.;
RT "Bcl-2 heterodimerizes in vivo with a conserved homolog, Bax, that
RT accelerates programmed cell death.";
RL Cell 74:609-619(1993).
RN [2]
RP NUCLEOTIDE SEQUENCE [LARGE SCALE MRNA].
RC TISSUE=Skin;
RX MEDLINE=22388257; PubMed=12477932; DOI=10.1073/pnas.242603899;
RA Strausberg R.L., Feingold E.A., Grouse L.H., Derge J.G.,
RA Klausner R.D., Collins F.S., Wagner L., Shenmen C.M., Schuler G.D.,
RA Altschul S.F., Zeeberg B., Buetow K.H., Schaefer C.F., Bhat N.K.,
RA Hopkins R.F., Jordan H., Moore T., Max S.I., Wang J., Hsieh F.,
RA Diatchenko L., Marusina K., Farmer A.A., Rubin G.M., Hong L.,
RA Stapleton M., Soares M.B., Bonaldo M.F., Casavant T.L., Scheetz T.E.,

```

Figure 5.5. Screenshot of SwissProt

SwissProt[12] is a manually annotated protein knowledgebase. TrEMBL, its counterpart, is computer-annotated. Together they form the UniProt Knowledgebase (see Section 5.3). The main point of SwissProt, and the reason why SwissProt is so popular, is the manual annotation. Each SwissProt entry contains core data (sequence data; bibliographical references and taxonomic data (description of the biological source of the protein)) and annotation, which consists of the description of the following items:

- Function(s) of the protein
- Post-translational modification(s).
- Domains and sites.

- Secondary structure
- Quaternary structure.
- Similarities to other proteins
- Disease(s) associated with deficiency(ies) in the protein
- Sequence conflicts, variants, etc.

Other advantages of SwissProt with regard to other protein sequence databases are the minimum of redundancy in the data and the integration with other databases. In order to have minimal redundancy and to improve sequence reliability, all protein sequences encoded by one gene are merged into a single SwissProt entry. It has to be said, however, that SwissProt annotation is not perfect. Even with manual annotation, there are misannotations found in SwissProt. These misannotations might influence the quality of the results of every text mining method.

SwissProt is used in CONAN via the UniProt terms (see Section 5.3) and as a part of a dictionary used in NLProt and the Boosting classifier (see Sections 6.3 and 7.4.2.1). For this dictionary, especially the Description (DE) lines of the SwissProt output are important. In Figure 5.5, you can see that the DE line holds the full name of the protein, in this case “Apoptosis regulator BAX, membrane isoform alpha”.

5.8. Other Services

In this section, I describe other database services which are related to databases mentioned before. These database services are used to link information together and to enrich the data by adding more information to the already extracted data.

5.8.1 GOA

The goal of the Gene Ontology Consortium is to produce a dynamic controlled vocabulary that can be applied to all organisms. In the GOA project [15], this vocabulary is applied to a non-redundant set of proteins described in the UniProt Resource (UniProtKB/Swiss-Prot, UniProtKB/TrEMBL and PIR-PSD) and Ensembl databases that collectively provide complete proteomes for *Homo sapiens* and other organisms.

In the first stage of this project, GO assignments have been applied to the human proteome by electronic mapping and manual curation. Subsequently, GO assignments for all complete and incomplete proteomes that exist in UniProt have been provided. In practice, the GOA-distribution consists of two main files.

The first file is a general mapping of GO identifiers to GO terms and GO processes. For example, the GO number *0000001* would be associated with *mitochondrion inheritance*, which is a *biological process*, thus assigned the letter P. Special numbers are *0000004*, *0005554* and *0008372*, which stand for *biological process unknown*, *molecular function unknown* and *cellular component unknown*, respectively.

The second file needed is the association of Uniprot terms to GO-identifiers. This file contains all GO assignments for the UniProt database, as well as other information:

- Database from which annotated entry has been taken.
- The GO identifier for the term attributed to the unique identifier in the DB.
- Reference cited to support the attribution.
- Identifier for the species being annotated.
- The UniProt identifier with which the GO id is associates

In CONAN, the information from the GOA files is used to link a protein name together with the GO categories. The UniProt identifier is the linking point that links the information together.

5.8.2 IPI

IPI [22] (International Protein Index) provides a top level guide to the main databases that describe the proteomes of higher eukaryotic organisms. IPI:

- effectively maintains a database of cross references between the primary data sources
- provides minimally redundant yet maximally complete sets of proteins for featured species (one sequence per transcript)
- maintains stable identifiers (with incremental versioning) to allow the tracking of sequences in IPI between IPI releases.

IPI protein sets are made for a limited number of higher eukaryotic species whose genomic sequence has been completely determined but where there are a large number of predicted protein sequences that are not yet in UniProt. IPI takes data from UniProt and also from sources of such predictions, and combines them non-redundantly into a comprehensive proteome set for each species.

The species currently included are: Human, Mouse, Rat, Arabidopsis, Zebrafish and Chicken. As these organisms are the most interesting and important ones for biologists, IPI is the perfect cross-reference-reference-point for

The screenshot displays the EMBL-EBI IPI service interface. At the top, the EMBL-EBI logo and navigation menu are visible. The main content area shows the entry details for IPI00071059.3, including general information, description, and coordinates. The 'Database cross-references' section is highlighted, showing the protein's accession number and related identifiers.

General information	
Entry name	IPI00071059.3
Accession number	IPI00071059 , IPI00027089, IPI00040935
Created	IPI Human rel. 2.04, 01-FEB-2002
Sequence update	IPI Human rel. 3.01, 01-DEC-2004
Description and origin of the Protein	
Description	BCL2-ASSOCIATED X PROTEIN ISOFORM EPSILON.
Organism source	Homo sapiens (Human).
Taxonomy	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
NCBI TaxID	9606
Comments	
CHROMOSOME	19.
START CO-ORDINATE	54149929.
END CO-ORDINATE	54156864.
STRAND	1.
Database cross-references	
DEFSEQ	REVIEWED MD_620110.ctd0621067.M

Figure 5.6. Screenshot of the IPI service

our system. A screenshot can be seen in Figure 5.6. Here, the information for the “BCL2-ASSOCIATED X PROTEIN ISOFORM EPSILON” is displayed. The most important information is the “Database cross-references” which are also used in CONAN to assign an Ensembl identifier to each UniProt identifier and therefore to each protein name found in text.

Chapter 6

CONAN PROGRAM COMPONENTS

In this chapter, I describe the components that are used in building CONAN. These components are already published methods and classifiers, as mentioned in the Introduction. In some cases, the original method has been used without changing the code, in other cases the code has been changed significantly by us. An in depth description of how the methods work together in CONAN can be seen in Chapter 8.

6.1. BLAST

The BLAST programs (short for Basic Local Alignment Search Tools) are a set of sequence comparison algorithms introduced in 1990 [7] that are used to search sequence databases for optimal local alignments for a query. BLAST is a word based heuristic similar to that of FASTA to approximate a simplification of the Smith-Waterman algorithm [24] known as the maximal segment pairs algorithm. BLAST can be used to perform a similarity search between two sequences (DNA or protein) or do a database search with a given DNA or protein sequence (query sequence). In the following description, I will focus on the database-scan function of the algorithm, because this function is used in CONAN. Firstly, the query sequence is broken into words with a fixed word size W , the size usually being between two and five letters. Then, the database is scanned for occurrences of these small words. When two non-overlapping words (within a certain distance in the query sequence) are matched against a database entry, and when no gap is opened inside the words, this pair is called a segment pair. This pair is extended to the left and right until the score drops below a certain cutoff score E . For each database entry, the same procedure is done, until there are no more resulting pairs (=hits) that satisfy the cutoff. All significant database hits are then shown in the results.

Table 6.1. Sample of Databases used in keyword search

Database	No. of Terms included
Gene	737,801
Protein	41,733
Organic Chemical	38,258
Disease or Syndrome	36,999
Therapeutic or Preventive Procedure	8,328
Neoplastic Process	7,791
Species	14,121
Body Part, Organ or Organ Component	6,555
Cell Component	818
Cell Function	456

In CONAN, the first method implemented is a BLAST-searching method first published by Krauthammer et al. [23]. It uses the BLAST-algorithm to discover relevant biological information in text. In the original method, only gene and protein names were used. In our approach, the original method was improved to extract even more information from an abstract.

We used the UMLS Metathesaurus [11] and the Semantic Network to retrieve lists of biological relevant terms. For a description of the UMLS Metathesaurus and the Semantic Network, please refer to Section 5.5.

For example, the term “Cell Migration” is assigned to the semantic type “Cell Function”, but the term “Cell Migration Inhibition” is placed in the semantic type “Laboratory Procedure”. It is important to know that the terms assigned to the semantic types are not overlapping.

The semantic types were chosen as names for the databases in which the concepts are stored. We chose the 91 semantic types that best suited the needs of the system, selecting the ones which are most practical for biomedical abstracts. Some of the semantic types chosen, together with the number of terms included, can be seen in Table 6.1. A complete overview of the databases used can be found in the Appendix. Other ones were deleted from the list, because the information included would be misleading or would yield a lot of false positives. Also, semantic types not related to the biomedical field were omitted. With this information, 91 databases ready for BLAST indexing were constructed.

For a normal BLAST search (`blastn`), it is obligatory to translate the query as well as the databases into a nucleic acids alphabet. This was done using the translation table also used in the original publication (see Figure 6.2). For each character, a four-letter code was used, each code starting with an A. Four letters have a major advantage over three letters in this application, because frame-shifts are averted. Moreover, each special character has the same four-letter-code, meaning that all special characters (like “-”, “(” or “)”) are treated in the same way.

Table 6.2. Translational Table used in the BLAST search

<i>Character</i>	<i>Code</i>	<i>Character</i>	<i>Code</i>
<i>A</i>	<i>AAAC</i>	<i>S</i>	<i>ACGC</i>
<i>B</i>	<i>AAAG</i>	<i>T</i>	<i>ACGG</i>
<i>C</i>	<i>AAAT</i>	<i>U</i>	<i>ACGT</i>
<i>D</i>	<i>AACC</i>	<i>V</i>	<i>ACTC</i>
<i>E</i>	<i>AACG</i>	<i>W</i>	<i>ACTG</i>
<i>F</i>	<i>AACT</i>	<i>X</i>	<i>ACTT</i>
<i>G</i>	<i>AAGC</i>	<i>Y</i>	<i>AGAG</i>
<i>H</i>	<i>AAGG</i>	<i>Z</i>	<i>AGAT</i>
<i>I</i>	<i>AAGT</i>	<i>0</i>	<i>AGCC</i>
<i>J</i>	<i>AATC</i>	<i>1</i>	<i>AGCG</i>
<i>K</i>	<i>AATG</i>	<i>2</i>	<i>AGCT</i>
<i>L</i>	<i>AATT</i>	<i>3</i>	<i>AGGC</i>
<i>M</i>	<i>ACAC</i>	<i>4</i>	<i>AGGG</i>
<i>N</i>	<i>ACAG</i>	<i>5</i>	<i>AGGT</i>
<i>O</i>	<i>ACAT</i>	<i>6</i>	<i>AGTC</i>
<i>P</i>	<i>ACCC</i>	<i>7</i>	<i>AGTG</i>
<i>Q</i>	<i>ACCG</i>	<i>8</i>	<i>AGTT</i>
<i>R</i>	<i>ACCT</i>	<i>9</i>	<i>ATAT</i>
		<i>Special Characters</i>	<i>ATCC</i>

Moreover, the terms belonging to one semantic type were split up according to size. This is because BLAST handles different sizes of queries differently and the BLAST parameters have to be tuned accordingly. The sizes used are 3 characters, 4-5 character, 6-10 characters, 11-20 characters and 20+ characters. An overview of the BLAST parameters used can be seen in Table 6.3. Please note that the most important differences are in the word size. A strict e-value cutoff was chosen to guarantee a high precision. These parameters were derived from the original publication, but altered slightly to ensure high precision. The settings were determined by running several experiments with different parameters. The best parameter set was selected.

So for each BLAST run, the abstract is converted into the query which is BLASTed against the databases constructed from the UMLS Metathesaurus.

The hits delivered by the BLAST algorithm are found back and parsed using the “BlastParser” library from Biopython (see Section 7.3), which is included in the “NCBIStandalone” module. Each hit is then read back to original text and the hit is expanded until the next special character or space.

Moreover, a stop-word list was compiled to filter out commonly appearing words in text that do not add any value in the biomedical domain. These list of stop-words contains the 500 most common words in the English language, downloaded from www.world-english.org/english500.htm.

Table 6.3. Parameters used in BLAST-search

Term length (characters)	e-value	Word Size	mismatch penalty
3	$1e^{-15}$	12	6
4-5	$1e^{-15}$	16	6
6-10	$1e^{-20}$	20	3
11-20	$1e^{-25}$	40	3
20 or more	$1e^{-25}$	80	3

An evaluation of the protein-tagging part of this method can be found in Section 10.2.6. The BLAST method can be categorized in the NER category, but also may be useful for IR. This means that the BLAST method can be used to find Protein/Gene Names in text as well as other important information.

6.2. AbGene

The second method implemented is a Gene/Protein-tagging method called AbGene first published by Tanabe [31]. This program is used without any modifications to the original program, so I only report here how this method was implemented originally. AbGene can be placed in the NER category of Literature Mining. It extracts Protein and Gene Names from text.

It uses a combination of statistical and knowledge-based strategies to extract gene and protein names from unstructured text. Firstly, the BRILL Pos Tagger [14] is used to assign each word of the text its most likely tag. Tags include Nouns, Adjectives, Adverbs and Verbs. A set of 7000 sentences was used to train the Brill tagger to recognize gene and protein names. For this reason, the new tag “GENE” was used. Moreover, new entries from the UMLS Specialist Lexicon were introduced to the training lexicon. New rules were extracted from this lexicon to ensure high quality of extracted gene and protein names. These rules are applied to text, resulting in a newly Brill-tagged output file.

This Brill output file is subsequently filtered, eliminating as much false positives and false negatives as possible. The false positive filtering is done through a list of 1505 precompiled terms, including terms that are not part of Protein names. The list includes amino acid names, restriction enzymes, cell line and organism names as well as general biological terms. Additionally, non-biological terms are filtered out through a list on terms compiled from the Wall Street Journal.

False negative names are tried to recover via a compilation of Gene/Protein names from LocusLink [28] and the Gene Ontology (see also Section 5.4)[9], which also forms the golden standard for this method.

For protein and gene names which might be overlooked by the Brill Tagger, the method uses tri-gram matching to identify terms that occur at least three times in MEDLINE. If a term contains one of 20172 low frequency trigrams

(i.e. tri-grams that do not occur often in the list of trigrams), it becomes a candidate for a “NEWGENE”-tag. In the final output, the “NEWGENE”-tag is converted to a “GENE”-tag.

False negatives are also corrected via a lexical look-up and a subsequent scoring mechanism. As this scoring mechanism is quite complex, please refer to the original publication [31] for details. Additional to the “NEWGENE”-tag, a “MULTIGENE”-tag is introduced for false-negative compound names. This tag is given to multi-word-gene/protein names. These multi-word entries are discovered by manually generated rules that are extracted from the gold standard of this method. The false negatives recovered with non-specific contextual rules, are tagged with the “CONTEXTGENE”-tag throughout the document.

For an extensive description of the filtering methods, please refer to the original publication [31].

This leaves a final output file where gene and protein names are tagged with the “GENE”- or the “MULTIGENE”- or the “CONTEXTGENE”-tag. These gene and protein names can then be extracted easily from the output file.

A typical output file of AbGene looks like this:

We/PRP conclude/VBP that/IN a/DT series/NN of/IN sites/NNS (NF-kappaB/GENE, IRF/GENE, GRE/CONTEXTGENE and/CC the/DT E/MULTIGENE box/MULTIGENE) are/VBP not/RP required/VBN for/IN efficient/JJ viral/JJ spread/NN in/IN the/DT sheep/NN model/NN, although/IN mutation/NN of/IN some/DT of/IN these/DT motifs/NNS might/MD induce/VB a/DT minor/JJ phenotype/NN during/IN transient/JJ transfection/NN assays/NNS in/IN vitro/FW.

Here we see that NF-kappaB and IRF receive the “GENE”-tag, whereas GRE has the “CONTEXTGENE”-tag, meaning that it was recovered from the false negatives. The protein “E box” is tagged as a “MULTIGENE”, because it represents a protein name that is split up in different words (=tokens) not connected to each other. The other tags are results of the Brill-tagger, representing different tags like proper noun (NNP), adjective (JJ) or verb/gerund (VBG). An evaluation of this method can be found in Section 10.2.6.

6.3. NLProt

Another method to find protein names and information about these proteins in text is the so called NLProt method [26, 25]. In this method, mostly Support Vector Machines (SVMs) are used to extract protein names from free text. For a description of SVMs, please refer to Section 7.4.1. This method belongs to the NER part of literature mining, although through the extraction of species

and tissue, it might also be placed in the Information Extraction (IE) category. In CONAN, NLProt is used for both purposes. Protein names are extracted (NER) and through the extraction of species and tissue, the UniProt code is determined (IE).

For NLProt, Protein and Gene name lists were compiled from different sources (SwissProt, TrEMBL) to derive a lexicon used as input for the SVMs. For this reason, the Description (DE) lines of the SwissProt entries have been parsed (see Section 5.7). Also a common dictionary was compiled as a negative list.

The input text is broken up into so-called tokens, every word in the text being a single token. Since protein names are never longer than five tokens, the token length was chosen accordingly, producing sample phrases from five tokens. Different sample phrases are compiled to serve as input for the SVMs. This input consists of two environment phrases and one center, the center being the phrase which is currently processed and the environment being the phrase before and the phrase after the center.

Four different SVMs are used to get the final output. Each of the first three SVMs concentrates on a specific input (center, environment, overlap between those two). The fourth SVM combines the output of the other three with a score derived from the protein-name dictionary, resulting in a final output and a score.

The major advantage of NLProt is its possibility to extract not only the protein name, but also the species, the tissue and the corresponding UniProt term from the text. NLProt scans each name found for surrounding words indicating a certain organism and gives only the protein identifier that fits both the protein and the species.

This is done via a list compiled from SwissProt terms (us.expasy.org/cgi-bin/specclist). If a protein name is surrounded (read: if a species name is part of the environment phrase, see above) by a term from the species list, it is assumed that the protein is part of that species and the species name and the UniProt term are changed accordingly. An evaluation of this method can be found in Section 10.2.6 and Section 10.2.4.

6.4. MuText

The fourth method implemented in CONAN is based on the MuText application [19]. This method can be categorized in the Information Extraction (IE) category of the Literature Mining process. It takes several regular expressions to detect mutations mentioned in an abstract. The pattern usually starts with an amino-acid in one- or three-letter-code, followed by a number (the residue where the mutation takes place) and another amino-acid. This code reveals that a certain amino-acid substitution occurs at a specific place

in a protein, also giving information about which amino-acid is substituted. The regular expressions used are:

```
[ARDNCEQGHILKMFPSTWYV][1-9][0-9]+
[ARDNCEQGHILKMFPSTWYV][1-9][0-9]*[ARDNCEQGHILKMFPSTWYV]
[ARDNCEQGHILKMFPSTWYV][ardnceqghilkmfpstwyv][ardnceqghilkmfpstwyv][1-9][0-9]*
[ARDNCEQGHILKMFPSTWYV][ardnceqghilkmfpstwyv][ardnceqghilkmfpstwyv][1-9][0-9]*[ARDNCEQGHILKMFPSTWYV][ardnceqghilkmfpstwyv][ardnceqghilkmfpstwyv]
```

An Example for a true-positive mutation would be:

We identified the NOS activated by the peptide as the neuronal isoform: the expression of the C415A neuronal NOS mutant inhibited both CCK-induced stimulation of NOS activity and cell proliferation. These two effects were also inhibited after expression of the C459S tyrosine phosphatase SHP-2 mutant and the betaARK1 (495-689) sequestrant peptide, indicating the requirement of activated SHP-2 and G-betagamma subunit.

In this example (PMID 10544187), both the mutation C415A and C459S are correctly extracted as mutations.

One very important thing to mention is that when applying these regular expressions, the false positive rate is very high. Very often cell-lines or other biologically interesting objects like receptors share a similar structure, also having one or more letters followed by a number and another letter. An example for text that fits the pattern but is not a true mutation is:

On the other hand, suppression of CBP/p300 function by the adenoviral protein E1A abolishes TORU promoter activation by p68.

Here (PMID 12527917), part of a protein name (adenoviral protein E1A) fits the regular expression. E1A is of course no mutation in the text, so it has to be filtered out.

This is why I invented special filtering methods that can distinguish between such a true positive (TP) and a false positive (FP). This is done via mapping of so-called “mutational terms” in the abstract by the Data Integrator. A further description can be found in Section 8.3.1.

6.5. PreBIND

The last method implemented in CONAN is used to extract interaction data from text. The basis of this method are again regular expressions as used in the PreBIND and BIND system [17, 10]. Some regular expressions

have been deleted by us from the set due to redundant results. PreBIND is one of the most prominent examples for Information Extraction (IE) in Literature Mining. There are several categories of possible interactions:

- positive interactions
- negative interactions (inhibitions)
- positive/negative complex building / subunit / association
- positive binding
- negative binding
- activation
- (de-)phosphorylation
- (co)precipitation
- conjugation
- mutation

It is very important to notice that not only positive interactions are found by this method. As inhibitions and complex-binding are also very important regulatory processes, they are also included in our results.

Basis of the extraction of PPIs are the Protein names extracted by the aforementioned NER methods. As interactions can only appear between two proteins, the protein names themselves are the key to extracting those interactions. So, for every abstract, a list of extracted protein names is passed on to the set of regular expressions, via which interactions between those proteins are found. A typical example of such a general regular expression would be:

”(A.*interacting.*B)” ^a

^aA list of all regular expressions used to find protein-protein interactions can be found in the Appendix

So, A and B are the proteins that should fit the regular expression. In between those two proteins, the word “interacting” should appear. An example sentence would be: *The Bcl-interacting protein Bax.....* This regular expression is quite general, when comparing it to one of the more specific regular expressions:

”(A(\S*\s+){0,3}and(\S*\s+){0,6}\S*B(\S*\s+){0,6}\S*interact(?!ion))”

Again, A and B represent the two protein names. Here we see that not only the protein names and the word that hints at the interaction are mentioned, but also the space which should be between A, and, B and interact(?!ion). At most 6 characters are allowed between those terms, limiting the search results significantly. An example sentence would be: *Bcl and Bax are interacting.....*

We see two different examples of regular expressions that are equally important for the system. More general regular expressions give the system a higher recall, whilst specific regular expressions are more designed to improve the precision of the system. To find a good balance between those two kinds of regular expressions is important.

An evaluation of this method can be found in Section 10.2.6 and Section 10.2.5.

Chapter 7

TECHNICAL COMPONENTS

7.1. XML

Extensible Markup Language (XML) [1] is a simple, very flexible text format derived from SGML (ISO 8879). As SGML is originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

The reason why XML is so popular at the moment is that the language is very simple and platform independent. Also, there is a large number of tools available. XML can be used to encode lots of different kinds of data, ranging from the Bible to technical manuals.

XML documents have a hierarchical and ordered tree structure. The structure is defined by elements. Elements are encoded in documents with tags, or words that are included in '<' and '>' brackets.

XML also supports schemas. The most important schema types are DTD and XML Schema. A DTD (Document Type Definition) is an external document which has to be declared in the XML file. It specifies the allowed values (see below) for each value of an element, e.g. character, decimal, string and the allowed length for each element. XML Schema is quite similar to DTD but it offers more functionalities, like the validation resulting in a collection of information adhering to specific datatypes.

A typical XML document looks like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PubmedArticle>
<MedlineCitation Owner="NLM" Status="MEDLINE">
<PMID>15231757</PMID>
<DateCreated>
<Year>2004</Year>
<Month>07</Month>
<Day>02</Day>
</DateCreated>
<DateCompleted>
<Year>2004</Year>
<Month>09</Month>
<Day>03</Day>
</DateCompleted>
<DateRevised>
<Year>2004</Year>
<Month>11</Month>
<Day>17</Day>
</DateRevised>
<Article PubModel="Print">
<Journal>
<ISSN IssnType="Print">1088-9051</ISSN>
<JournalIssue CitedMedium="Print">
<Volume>14</Volume>
<Issue>7</Issue>
<PubDate>
<Year>2004</Year>
<Month>Jul</Month>
</PubDate>
</JournalIssue>
<Title>Genome research. </Title>
<ISOAbbreviation>Genome Res.</ISOAbbreviation>
</Journal>
<ArticleTitle>Single nucleotide polymorphisms associated with rat
expressed sequences.</ArticleTitle>
<Pagination>
<MedlinePgn>1438-43</MedlinePgn>
</Pagination>

</Article>
</MedlineCitation>
</PubmedArticle>
```

The physical composition of an XML file consists of several building blocks:

- **Entities.** The main entity is the XML document as such. Other entities can be included as entity references (references to other XML documents).
- **Declarations.** An XML declaration explaining the version of XML used, the character coding and the entity definition is optional. In the example, the line “<?xml version=“1.0” encoding=“ISO-8859-1”?>” is the declaration of the XML-version used.
- **Elements.** As mentioned before, elements are encoded with tags or words. An example of an element would be “<PMID>15231757</PMID>”, with “<PMID>” being the opening tag, “15231757” being the element content and “</PMID>” being the closing tag.
- **Attributes.** Can be added for additional information about an element. Represented by an Attribute-Value-Pair. An example for an attribute is “<MedlineCitation Owner=“NLM” Status=“MEDLINE”>”. “MedlineCitation” is the element, “Owner=“NLM” and Status=“MEDLINE” are the attributes which belong to the element.
- **Comments.** Comments for the XML file.

For an XML document to be correct, it must be well-formed and valid. Well-formed means that it conforms to all XML syntax rules. An element that has an opening tag, but no closing tag, does not conform with these rules. Valid means that the document conforms to the user-defined rules (XML Schema). This means that if a certain element or attribute only can contain text (by user definition) and it contains numbers, it is not valid.

7.2. XPath

XPath (XML Path Language) [2] is a W3C recommendation for selecting and addressing nodes in an XML document. Initially, XPath started as a subset of the eXtensible Stylesheet Language (XSL) but became an independent language in an early stage of development of XSL. XSL consists of three parts:

- **XSL Transformations (XSLT):** a language for transforming XML.
- **XML Path Language (XPath):** an expression language used by XSLT to access or refer to parts of an XML document.
- **XSL Formatting Objects (XSL-FO):** an XML vocabulary for specifying formatting semantics.

XPath is used in XSLT, Xlink and Xpointer. In addition, several query languages used the Xpath syntax for the selection of XML nodes, e.g. XQL, Quilt, XQuery and SQL/XML.

The current version is XPath2.0 which includes important changes with regard to version 1.0., e.g. the support for XML Schema Data Types and the possibility to use variables. The most common kind of XPath expression is a path expression. A path is a sequence of commands to traverse through the nodes, starting with the root node. The steps are separated by “*slash*” commands. The simplest kind of XPath expression would be “/A/B/C”. This expressions selects the nodes C which are children of nodes B, which are children of node(s) A. This is called an “Axis specifier”, because it queries specific points over the axis of the XML document. The axes available are: child, attribute, descendant, descendant-or-self, parent, ancestor, ancestor-or-self, following, preceding, following-sibling, preceding-sibling, self, and namespace.

An XPath statement can have more components: A Node Test can be carried out, which means that a certain Node Name is identified within an axis and is true if (and only if) the type of the node is the type specified in the XPath Query. A Predicate query can be done, searching for specific values in all elements. Also a combination of those three types is possible. XPath also offers many different operators and standard functions. All those components can be best described with an example. Going back to our XML example, some interesting XPath queries would be:

```
/PubmedArticle/MedlineCitation/PMID/.
```

This query would give back the PubMed ID (PMID) of this article.

```
/PubmedArticle/MedlineCitation/PMID[contains(.,15231757)]/DateCreated/Year/.
```

This query says that it should get the value of the element year, for every PubMed Article with the PubMed ID 15231757. The result would be “2004”.

```
/PubmedArticle/MedlineCitation/PMID/DateCreated/Year[contains(.,"2004")]/../../*
```

This query looks for every article published in the year 2004 and displays the PMIDs associated with those articles.

So we see that when using XPath queries in a normal XML document, those queries are quite intuitive. While the query syntax is quite simple, it offers a lot of functionality.

7.3. Script Languages

Script Languages are computer programming languages initially designed for “scripting” the operations of a computer. Though in early times, scripting languages were only used for simple tasks, like automating computer commands, nowadays they are used in nearly all fields of computer programming.

Scripting normally means only “connecting pre-existing components” to deliver a certain task, but these days they are also used to program components of a systems. Programs written in such a Scripting Language are often not compiled (like in C or C++), but interpreted. The biggest advantages of Scripting Languages are that they are faster to program. For some script-languages, the code is more easily human-readable and the files are much smaller. On the downside, Scripts sometimes perform slower and use more memory than a comparable C or C++ program.

Here I introduce two important Scripting languages in the bioinformatics world, namely Python and Perl. Python [3] is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, Scheme or Java.

Python combines power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC, wxWidgets). New built-in modules are easily written in C or C++. Python is also usable as an extension language for applications that need a programmable interface.

One feature that makes Python one of the most important programming languages for biologists and bioinformaticians is the fact that Biopython [4], a project by an international association of developers, provides free-to-use Python tools for molecular biologists.

Biopython includes modules for handling and transforming sequences, connecting to all biological important databases, dealing with alignments and putting biologically interesting information in local databases. Moreover, Python is very good in processing strings, which is one of the reasons why Python is often used in the text mining community. Perl [5], short for Practical Extraction and Report Language, is an interpreted procedural programming language designed by Larry Wall. Perl has a unique set of features partly borrowed from C, shell scripting (sh), awk, sed, and (to a lesser extent) many other programming languages (even Lisp). Perl is commonly used in the bioinformatics world, maybe even more than Python. Similar to Python, Perl provides additional programs and tools for bioinformaticians. BioPerl [6] is provided by the Open Bioinformatics Foundation and the scripts and tools provided are freely downloadable and usable.

The main differences between BioPerl and BioPython are the packages they provide to the user. While both languages provide basic packages (e.g. Sequence Alignment, Database Retrieval, Basic Statistics), BioPython also provides many packages that provide code for storing and retrieving biological objects from relational database systems (e.g. the BioSQL packages). BioPerl, on the other hand, provides a large package for the analysis Microarray experiments (Microarray package) and for the construction of graphical user interfaces (GUI package). Since the latest release, however, BioPerl also offers

a database package. As often, the choice between BioPerl and BioPython is as much a choice of convenience as a matter of taste.

7.4. Classification Techniques

In text mining, classification techniques are heavily used. Classification questions asked range from IR (Does this document belong to the biomedical domain ?) to NER (Is this name really a Protein name ?). Classification is defined as a statistical procedure in which individual items are assigned to categories based on quantitative information on one or more characteristics inherent in the items (referred to as variables) and based on a training set of previously labeled items. The labelled cases are the training or learning set, in most cases the parameters for the whole classifier are estimated using this training set. Two important terms here are Predictor Variable and Target Variable. A predictor variable is a variable whose categories identify class or group membership, which is used to predict responses on one or more dependent variables. A target variable is a variable for which the right class has to be found using the classifier. This class is derived from the predictor variables. The problems that occur very often when using Classification techniques are: finding the optimal parameters and overfitting. Overfitting is fitting a statistical model that has too many parameters. Overfitting occurs in cases where learning was performed too long or where training examples (predictor variables) are rare. To avoid overfitting, cross-validation is used to evaluate the fitting provided by each parameter value set tried during the grid or pattern search process. In the next sections I present two of the most modern and frequently used Classification Techniques, Support Vector Machines (SVMs) and Boosting.

7.4.1 Support Vector Machines

Support Vector Machines (SVMs) [13] are supervised learning methods used for classification. An overview can be seen in Figure 7.1.

In a linear classification case, a maximum-margin hyperplane is constructed by the SVM to split the data into two distinct classes. Maximum margin means that the distance of the closest example(s) to the hyperplane is maximized. The parameters of the maximum-margin hyperplane are derived by solving a quadratic programming (QP) optimization problem with linear constraints. There are several possibilities to solve the QP problem, one of the most prominent being the Sequential Minimal Optimization (SMO) algorithm [27]. A predictor variable is called an attribute. A transformed attribute that is used to define the hyperplane is called a feature. The task of choosing the most suitable representation is known as feature selection. A set of features that describes one case (i.e., a row of predictor values) is called a vector. So the goal of SVM modeling is to find the optimal hyperplane that separates clusters

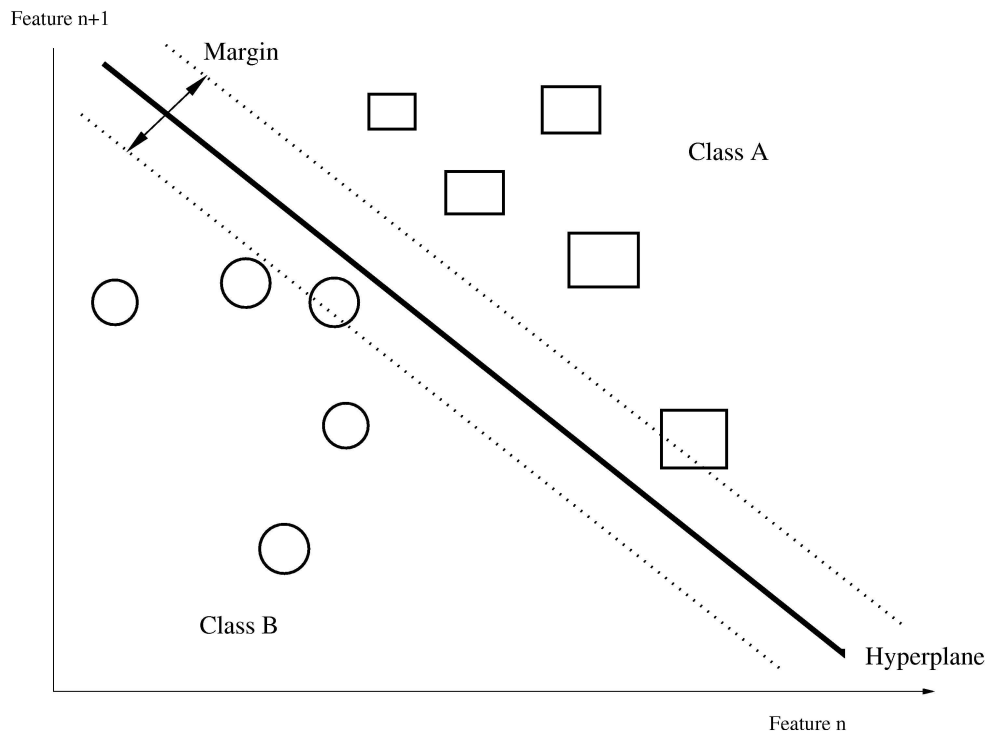


Figure 7.1. Support Vector Machine

of vectors in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other side of the plane. The vectors near the hyperplane are the support vectors.

In a non-linear classification case, the classifiers are constructed using a kernel-function to map the data into a different space (N-dimensional) where a hyperplane can be used to do the separation. The default and recommended kernel function is the Radial Basis Function (RBF).

There is a modified maximum margin idea that allows for mislabeled examples [16], called “Soft Margins”. If there exists no hyperplane that can split the “yes” and “no” examples, the Soft Margin method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. In CONAN, SVMs are used by NLProt to find protein names in text.

7.4.2 Boosting

Boosting (or boosted regression) [29] is a machine learning meta-algorithm for performing supervised learning. Boosting is based on probably approximately correct learning (PAC learning [32]), which is a branch of computa-

tional learning theory. The underlying idea of Boosting is to combine simple rules to form an ensemble such that the performance of the single ensemble member is improved, i.e. “boosted”.

$$f(x) = \sum_{t=1}^T a_t h_t(x)$$

In this equation, a_t denotes the weight of the ensemble member h_t . Both a_t and the learner h_t are learned within the Boosting procedure. $f(x)$ is the composite ensemble hypothesis. At every stage of the learning process, a “weak learner” (h_t) is trained with the data. The job of the “weak learner” is to find a hypothesis appropriate for the data. The quality of such a weak learner is measured by the error with respect to the distribution of the training data. The output of such a learner is added to the output function, examples which are classified wrongly, get a boost (=weight is increased, i.e. a_t is increased) for the next round of learning, giving another “weak learner” a chance to fix the error. So the weight is fixed in a way that the “weak learners” concentrate on hard examples that are difficult to classify. Subsequently, examples identified correctly are weighted less. After some rounds of boosting, the algorithm delivers a final hypothesis ($f(x)$) over the training set. The number of boosting rounds is normally a parameter. The final function can be used to classify other examples than the training examples. Problems that can occur in Boosting are: insufficient data, overly complex weak hypotheses and weak hypotheses that are “too weak”. Important Boosting algorithms include AdaBoost [18] and C5.0 (<http://www.rulequest.com>). AdaBoost is quite sensitive to noisy data and outliers, as is reported in the original publication. An algorithm that focuses on text categorization and is built using the AdaBoost algorithm, is BoosTexter [30].

An example how boosting (and especially BoosTexter) was implemented in our system is described in the next section.

7.4.2.1 BoosTexter

A Boosting implementation called BoosTexter [30] was used that is designed to work with text. BoosTexter is an extension of the AdaBoost Algorithm [29]. It uses so-called n-grams to classify a given document. n-Grams are sub-sequences of n items from a given sequence. In the NLPProt method, such n-grams are called tokens. Each word is considered a “gram”. An n-gram of size 1 is a “unigram”; size 2 is a “bigram”. So for a bigram, every sentence would be split up in two-word-snippets. This can of course also be applied not to full texts but also to words like protein names, because protein names often contain more than one word. In this approach, we used a specific kind of n-grams, called s-grams (for sparse n-gram). An example of a sparse ngram is the pattern “the ? boy” which matches any three word sequence beginning with the word “the” and ending with the word “boy”. An example from the biomedical world would be “protein kinase?”, where every protein kinase

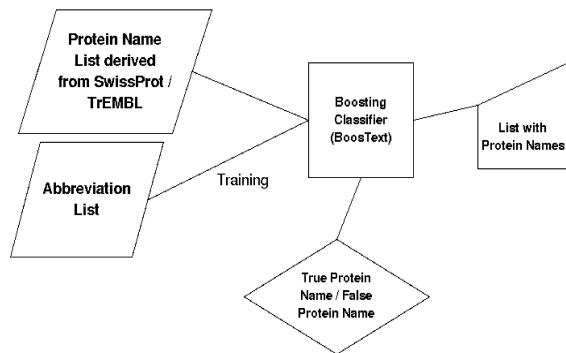


Figure 7.2. Overview of the Boosting classifier

like “protein kinase C” or “protein kinase A” would match the s-gram. The weak learners in the BoosTexter implementation are the “s-grams” in the text. Subsequently, the training data is weighted by checking how often a certain “s-gram” is mentioned in the dictionary.

In our case, we trained the classifier on a large dictionary containing protein names derived from SwissProt/TrEMBL (see Section 5.7) [12]. and a dictionary consisting of protein name abbreviations which is published by iProLINK (see Section 5.6) [20] as positive cases. This approach can be seen in Figure 7.2. So every positive case for a protein name will be classified as a positive and every example that does not contain a protein name, will be unclassified. When using Boosting, normally a list of negative examples is given. It is unclear what a “negative example” for protein detection should look like, given that protein names such as the Drosophila “dachshund” gene, which is also a normal English word, exist. Moreover, a negative list in this case would have to consist of all English words that are not part of any protein name. This seems an impossible task. Therefore we only used positive examples and a minimal threshold on the output weights. This threshold was determined by looking at different test sets and is set at +0.15. This threshold ensures that the Boosting Classifier produces very high precision results.

One important step is the pre-processing of the dictionary. First of all, all special characters like parentheses or commas were converted into spaces, and everything was converted to lower case. After that, digits were separated from characters. This means that for instance the protein name “transcriptional regulator atf3” was split to “transcriptional regulator atf 3”. This was done

because of the s-gram matching. The principle behind the matching is that there is more than one “transcriptional regulator atf” and not all of them might be part of the training set.

So the BoosTexter algorithm will make an s-gram saying:

“transcriptional#regulator#atf# ?”

,meaning that the classifier will recognize all the transcriptional regulator atf proteins, no matter what digit is behind it and will classify them as a true protein name. For the BoosTexter training phase we used 50 rounds of boosting. We used sgrams of size five, because no protein name is longer than five snippets or tokens. As a result, the BoosTexter algorithm came up with 74467 strong hypotheses (SHYPS). These hypotheses include a lot of unigrams (e.g. protein) and a lot of bigrams (e.g. tumor#suppressor).

Longer SHYPS that were found include “low#density#lipoprotein# ?”.

For the resulting final hypothesis, we see that a term like “protein”, which appears very often in protein names, gets a large weight of +3.41. An interesting result is that the s-gram “cytochrome#b5” gets a weight of +0.45 whereas the single word “cytochrome” gets a weight of +1.48. This shows the strength of the s-grams. “cytochrome” as such is a perfect protein name, “b5” is not, so the weight is decreased. It is still above the threshold, so “cytochrome b5” is classified as “correct”.

For a given test set, BoosTexter compares those SHYPS with the input of the test set and gives back weights whether the input belongs to the class “protein” or not. Although BoosTexter can also be used for multi-class problems, we focussed on the specific classification “protein” or “no-protein”. The outcome is that, when a name is classified as a protein, it gets a positive weight above 0.15, when it is classified as being no protein, it gets the weight 0 to 0.15. Those weights could be converted to probabilities, but for a binary classification problem, this is not really necessary.

A problem that occurs when building this classifier is the plural form of protein names. If we look at an example “glutathione ? transferase”, every glutathione transferase would be classified correctly, but any term saying “glutathione S transferases” would be classified incorrectly, because the s-gram does not match. This can be easily overcome by checking if any words in the test-set have an “s” at the end. If there is any, both the form including the “s” and without the “s” are added to the test-set (e.g. glutathione S transferases and glutathione S transferase). If the version including the “s” is considered as a positive, the version without the “s” is thrown away. If it is considered a negative and the version without the “s” is a positive, the version with the “s” is thrown away. This is best described by the protein name “c-Fos”. This would be added as “c-Fos” and “c-Fo” in the test set. As “c-Fo” is not a correct protein name, but “c-Fos” is, the “c-Fo” is thrown away.

References

- [1] <http://www.w3.org/XML>.
- [2] <http://www.w3.org/TR/xpath>.
- [3] <http://www.python.org>.
- [4] <http://www.biopython.org>.
- [5] <http://www.perl.org>.
- [6] <http://bioperl.org>.
- [7] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J Mol Biol.*, 215(3):403–410, 1990.
- [8] R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, R. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L. S. Yeh. UniProt: the Universal Protein knowledgebase. *Nucleic Acids Res*, 32:D115–D119, 2004.
- [9] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet*, 25(1):25–29, 2000.
- [10] G.D. Bader, D. Betel, and C.W. Hogue. BIND: the Biomolecular Interaction Network Database. *Nucleic Acids Res.*, 31(1):248–250, 2003.
- [11] O. Bodenreider. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res.*, 32:267–270, 2004.
- [12] B. Boeckmann, A. Bairoch, R. Apweiler, M.C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, PhanI., S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res.*, 31(1):365–370, 2003.

- [13] B.E. Boser, I. M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual ACM Workshop on COLT*, pages 144–152, 1992.
- [14] E. Brill. Some Advances in Rule-Based Part of Speech Tagging. *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 722–727, 1994.
- [15] E. Camon, M. Magrane, D. Barrell, D. Binns, W. Fleischmann, P. Kersey, N. Mulder, T. Oinn, J. Maslen, A. Cox, and R. Apweiler. The Gene Ontology Annotation (GOA) project: implementation of GO in SWISS-PROT, TrEMBL, and InterPro. *Genome Res.*, 13(4):662–672, 2003.
- [16] C. Cortes and V.N. Vapnik. Support-vector networks. *Machine Learning*, (3):273, 1995.
- [17] I. Donaldson, J. Martin, B. de Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G.D. Bader, K. Michalickova, T. Pawson, and C.W. Hogue. PreBIND and Textomy—mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics.*, 4:11, 2003.
- [18] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [19] F. Horn, A.L. Lau, and F.E. Cohen. Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors. *Bioinformatics.*, 20(4):557–568, 2004.
- [20] Z.Z. Hu, Mani I., H. Liu, and C.H. Wu. iProLINK: an integrated protein resource for literature mining. *Comput Biol Chem*, 28(5-6):409–416, 2004.
- [21] T. Hubbard, D. Andrews, M. Caccamo, G. Cameron, Y. Chen, M. Clamp, L. Clarke, G. Coates, T. Cox, F. Cunningham, V. Curwen, T. Cutts, T. Down, R. Durbin, X. M. Fernandez-Suarez, J. Gilbert, M. Hammond, J. Herrero, H. Hotz, K. Howe, V. Iyer, K. Jekosch, A. Kahari, A. Kasprzyk, D. Keefe, S. Keenan, F. Kokocinski, D. London, I. Longden, G. McVicker, C. Melsopp, P. Meidl, S. Potter, G. Proctor, M. Rae, D. Rios, M. Schuster, S. Searle, J. Severin, G. Slater, D. Smedley, J. Smith, W. Spooner, A. Stabenau, J. Stalker, R. Storey, S. Trevanion, A. Ureta-Vidal, J. Vogel, S. White, C. Woodwark, and E. Birney. Ensembl 2005. *Nucleic Acids Res*, 33(Database Issue):D447–D453, 2005.
- [22] P. J. Kersey, J. Duarte, A. William, Y. Karavidopoulou, E. Birney, and R. Apweiler. The International Protein Index: An integrated database for proteomics experiments. *Proteomics*, 4(7):1985–1988, 2004.
- [23] M. Krauthammer, A. Rzhetsky, P. Morozov, and C. Friedman. Using BLAST for identifying gene and protein names in journal articles. *Gene*, 259(1), 2000.
- [24] D.J. Lipman, W.J. Wilbur, T.F. Smith, and M.S. Waterman. On the statistical significance of nucleic acid similarities. *Nucleic Acids Res.*, 11:215–226, 1984.
- [25] S. Mika and B. Rost. NLProt: extracting protein names and sequences from papers. *Nucleic Acids Res.*, 32, 2004.

- [26] S. Mika and B. Rost. Protein names precisely peeled off free text. *Bioinformatics.*, 20(Suppl 1):I241–I247, 2004.
- [27] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.
- [28] K.D. Pruitt and D.R. Maglott. RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Res*, 29(1):137–140, 2001.
- [29] R.E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.
- [30] R.E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2):153–168, 2000.
- [31] L. Tanabe and W.J. Wilbur. Tagging gene and protein names in biomedical text. *Bioinformatics*, 18(8):1124–1132, 2002.
- [32] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

PART III

CONAN

Chapter 8

CONAN

In this chapter, I introduce CONAN [1, 2], a system that was developed for systematic text mining. CONAN is the core part of this thesis. It was developed by me with all components described in Part II of this thesis. All the Results in Part IV are computed with CONAN and all Applications presented in Chapter 11 include CONAN in one way or the other. In this Chapter, I talk about the workflow of this system, giving information about input, pre-processing, processing the data, filtering the results and finally the output. The flow-diagram of how CONAN works can be seen in Figure 8.1. The CONAN Process begins with a MEDLINE XML file (see Section 5.1), which also includes the MeSH terms (see Section 5.1.2). After pre-processing, the several methods of information extraction are started, namely Mutation Finding (see Section 6.4), Protein Name Finding (see Sections 6.1, 6.2 and 6.3), Interaction Finding (see Section 6.5) and the Biological Concept Finding (BLAST search, see Section 6.1). All of this information is collected in the Data Integrator (see Section 8.3.1), the Boosting Classifier (see Section 7.4.2.1) is applied and the output is stored in an XML file (see Section 8.4). This XML file can subsequently be queried via a web-server (see Section 11.2) or via a command line tool (see Section 11.1).

At this point of time, the CONAN webserver is available for testing purposes at <http://h094.niob.knaw.nl/conan>. A set of example queries which can be executed can be found in the Appendix.

8.1. Input

In this section, I describe all files used as input for CONAN. The databases which these input files are derived from can be seen in Chapter 5.

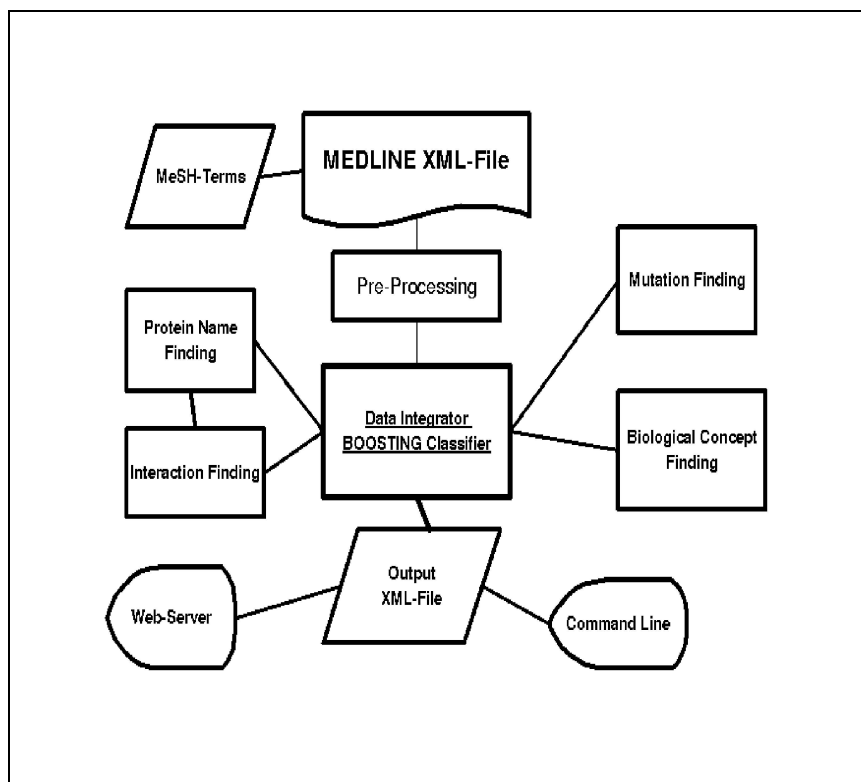


Figure 8.1. Flow Diagram of CONAN

8.1.1 MEDLINE XML Files

The main input of the whole system consists of MEDLINE files which are freely downloadable for academic purposes when a license agreement is signed. The MEDLINE files serve as the only source of input of text data.

These MEDLINE input files are provided in the MEDLINE XML style. The main structure of such a MEDLINE file can be seen in the Appendix. The fields needed for analysis are the following:

```
<MedlineCitation Owner="NLM" Status="Completed" >
```

This entry of the MEDLINE file signals that this specific abstract is Completed. In later versions (from 2005), “Completed” was changed to “MEDLINE”. This field is assigned if the abstract is complete and is going to be in the online version of MEDLINE. Other versions of this field include “In-Process”, meaning that this version of the abstract is not the final version and “In-Data-Review”, meaning that important data parts of this abstract

are still under review. For the analysis with CONAN, only abstracts that are “Completed” or “MEDLINE” are taken into account. This minimizes the risk of processing an entry more than once. It also means that all important information is stored in the abstract. When the abstract is not “Completed”, information might still change in time. Hence, the chance of errors is much smaller when using only “Completed” abstracts.

```
<PMID>12411390</PMID>
```

This field gives the PMID of the abstract. The PMID (PubMed ID) is an unique identifier assigned to every abstract in PubMed/MEDLINE. This information is needed in CONAN to identify the article when querying the data, for instance, when querying the interactions, it is important to know to what article the extracted interactions belong.

```
<AbstractText>The focal pattern of atherosclerotic lesions in arterial vessels suggests that local blood flow patterns are important factors in atherosclerosis. Although disturbed flows in the branches and curved regions are proatherogenic, laminar flows in the straight parts are atheroprotective. Results from in vitro studies on cultured vascular endothelial cells with the use of flow channels suggest that integrins and the associated RhoA small GTPase play important roles in the mechanotransduction mechanism by which shear stress is converted to cascades of molecular signaling to modulate gene expression. By interacting dynamically with extracellular matrix proteins, the mechanosensitive integrins activate RhoA and many signaling molecules in the focal adhesions and cytoplasm. Through such mechanotransduction mechanisms, laminar shear stress upregulates genes involved in antiapoptosis, cell cycle arrest, morphological remodeling, and NO production, thus contributing to the atheroprotective effects. This review summarizes some of the recent findings relevant to these mechanotransduction mechanisms. These studies show that integrins play an important role in mechanosensing in addition to their involvement in cell attachment and migration.</AbstractText>
```

This field gives the text of the abstract. As the abstract is the main point of processing PubMed/MEDLINE entries, this is the most important field for our analysis.

```
<MeshHeading>
<DescriptorName MajorTopicYN="N">Animal</DescriptorName>
</MeshHeading>
```

MeSH terms (see Section 5.1.2) are used to describe the content of an abstract and annotate the abstract according to pre-defined fields. MeSH terms are frequently used to categorize articles by IE and IR systems. As the MeSH terms form an important part of MEDLINE entries, the values of the MeSH headings are extracted. As these terms are not further processed, but stored directly in the CONAN output file, the MeSH terms serve as an input for the system.

8.1.2 GOA

The description of the GOA service can be found in Section 5.8.1. For the assignment of Gene Ontology Terms and Identifiers to each protein mentioned in the abstract, two files have been downloaded from the GOA website (www.ebi.ac.uk/GOA/). They are converted to an easy queryable format. The first file (“GO.terms_and_ids”) provides a mapping between the GO identifiers (e.g. 0000023), the GO description (e.g. maltose metabolism) and the class it belongs to (e.g. Process). The version used in all experiments is GOA version 35.0.

The second file (“gene_association.goa_uniprot”) is needed for the mapping of UniProt identifiers derived from NLPProt (e.g. MYO1F_HUMAN) to the GO identifiers (e.g. for MYO1F_HUMAN, these are 0000004, 0000166, 0003774, 0003779, 0005516, 0005524, 0016459, 0016461). It is important to note that this is a “many-to-many” relationship, meaning that one UniProt term can be mapped to multiple GO identifiers and one GO identifier can map to multiple UniProt terms.

As this second file is quite big (683,609,334 bytes or 7,029,958 entities), it was necessary to index this file for fast processing. This was done via the freely available software tool Glimpse (webglimpse.net/).

8.1.3 IPI

As described before (see Section 5.8.2), the IPI (International Protein Index) files provide a mapping from (almost) all available protein identifiers to the protein names and their synonyms. In this case, we use the mapping from UniProt terms to Ensembl identifiers. As much of the other information is not needed by CONAN, we processed the IPI files for several organisms (Human, Rat, Mouse, Zebrafish, Arabidopsis, Cow) and extracted the UniProt terms together with the corresponding Ensembl identifiers to form a lookup-table. This ensures that finding the wanted information is very quick. The IPI file with version number 3.11 was used throughout all experiments, this file is based on the NCBI 35 assembly of the human genome, the NCBI m34 mouse assembly and the 3.4 version of the rat genome.

8.2. Pre-Processing

Before starting to produce the results with CONAN, the input data has to undergo several pre-processing steps. This pre-processing is part of the CONAN workflow, although there is no actual information extracted in these steps.

- Extraction of Abstracts and PMIDs
- Storing the information in a temporary file
- Translating the Abstracts into DNA Alphabet (see Section 6.1)
- Converting PMIDs/Abstracts to the AbGene input format and producing the AbGene input file
- Converting PMIDs/Abstracts to NLProt input format and producing the NLProt input files

These steps are necessary to ensure that all files are in place before the processing begins and that the processing works as quickly and as smoothly as possible. When these steps are finished, the real CONAN processing and extraction steps are started.

8.3. Processing

In CONAN, each extraction method (see Chapter 6) is started one at a time. Firstly, the keywords and entities are extracted by the BLAST method, after that the Mutations in the text are extracted, then the NER takes place with NLProt and AbGene and finally, the PPIs are extracted by CONAN. This ensures data integrity. If the processing would take place simultaneously, wrong information might be entered in the output file or XML entries might overlap, i.e. if one method is finished before the other, the closing tags might not be set correctly. Firstly, the BLAST method is run and useful biological concepts (protein names, gene names, diseases, etc) are extracted and stored. Secondly, AbGene is run to find protein names in the text. Thirdly, MuText is run to find mutations in the text. Fourthly, NLProt is run to extract protein names, the corresponding organism and the UniProt identifier in the text. Subsequently, via the GOA and IPI files, the Ensembl identifier for a given protein name and the Gene Ontology classification are extracted and stored. Following this extraction, the lists of protein names found by the different methods are passed on to the PreBIND method which extracts protein-protein interactions. Finally, the MeSH terms are extracted from the original XML files and are also stored in the output file. At every given stage (see Section 8.3.1), Data Integration occurs.

For each method, the data is extracted in the way that is described in Chapter 6. The output format is described in Section 8.4.

8.3.1 Data Integration

The most important aspect of CONAN is the Data Integration. There are three major stages in which CONAN combines data. Although they are not all happening at the time the XML file is produced, I describe all three steps at this time, for the sake of clarity.

- Firstly, combination of data is done when storing it in the Output-XML-file. In this step, the data of the NER methods is combined together with the regular expressions of the interaction finding methods. So the Data Integration takes place in the interaction finding method. The protein names found by the two protein-tagging methods and the BLAST method (given that the term belongs to the "Gene" or "Protein" database) are used as input for the interaction finding method. The list of protein names found by the NER methods is passed on to the regular expressions used to find interactions. This ensures a high precision of interaction finding, because only protein names which are found before are taken into account. This also means that protein names which are not found by the NER methods cannot participate in an interaction. Moreover, when the protein name is classified as "false" by the Boosting Classifier in the post-processing, the interactions that contain the specific protein are marked as "false" as well.
- Secondly, a case of data integration is the validation of protein names found by NLProt. In this step, results from all three NER methods are combined. The lists of all three NER methods are comparable, because they also contain the position of the protein name in the abstract. NLProt offers a reliability score for every protein name found in the text. This score (normally between 0 and 3) is normalized and divided by three so that it is in the range between 0 and 1. This makes comparison to other scores (e.g. PreBIND regular expression measure, see below) easier. By comparing those lists of protein names with the lists produced by the other methods, namely AbGene and the BLAST-searching, we make sure that high quality protein names also receive a high score in the interaction finding process (see below). This is ensured by the fact that the score is increased by 0.5 if the protein name is found by the other methods as well or decreased by 0.5 if the protein name is not found by either of the other methods. This means that proteins that are found by all methods receive a high score. The ones that are found by one or two methods receive a lower score, but stay in the list, and are passed on to the interaction finding method.
- Thirdly, data is combined at the time when the Output-XML-file is queried. One example is the querying for Mutations, the method of extracting these mutations can be found in Section 6.4. Mutations are found in text by simple regular expressions, the pattern usually starting with one amino-acid in one- or three-letter-code, followed by a number and another amino-

acid abbreviation. This method alone would result in a large number of false positives, because many other biological concepts like Cell Lines have these kinds of abbreviations. To solve this problem, the BLAST-searching method is queried for “mutation-related” terms in text.

Terms as “Mutation”, “Mutational Analysis” or “Amino Acid Substitution” are examples for those kind of keywords. If one of those keywords is found by the BLAST-searching method or the MeSH terms describing this abstract contain one of these terms, then the chance of the mutation being a true positive is high. The BLAST-searching method also gives the possibility to compare the positions of the mutation and the mutational term in the abstract. How closer the two are found together, the higher is the chance of the mutation being a true positive. In fact, we compute the distance of the Mutation and the mutational term. We assume, that an average sentence in an abstract is about 10-20 words long. If the distance is not longer than 30 words, it is considered a positive. This is due to the fact that those two facts should ideally be inside one sentence, but also could be in the next sentence. The other example of this kind of data integration is the score for the interaction data. Each regular expression has a score assigned by PreBIND, yielding a measure how good this specific regular expression performs in the extraction of interaction data. This measure ranges from 0 to 1. The score for protein 1 included in the interaction (see above), the score for protein 2 included in the interaction and the score for the regular expression are combined to one score by taking the mean of all three scores, giving the user the possibility to see directly whether this interaction is predicted to be of high quality or not. This is also the reason why the scores of NLProt are normalized to a range from 0 to 1. In this example, data from two different sources (NER methods and PreBIND data) is integrated to form one single score which provides information on the quality of the extracted interaction.

8.4. Output

CONAN stores its output as an XML file, as already mentioned in the Introduction. Since the different methods format their output slightly differently, I briefly describe the typical output for each method and then the output that is unique for every method. The DTD for the CONAN output file can be found in the Appendix.

8.4.1 General

The general output of CONAN looks like:

```
<abstract>
<abstract_header>
<abstract_id> </abstract_id>
<abstract_file> </abstract_file>
<abstract_file_position> </abstract_file_position>
<PMID> </PMID>
</abstract_header>
<method_overview>
<method> </method>
<parameters> </parameters>
</method_overview>
<abstract_body>
. . .
</abstract_body>
</abstract>
```

For each abstract that is processed, an XML Tag “<abstract>” is opened. The abstract consist of three different sub-tags, namely the “abstract_header”, the “method_overview” and the “abstract_body”.

In the “abstract_header”, the running number of the abstract in the output-file (“<abstract_id>”), the MEDLINE-file from which this abstract is taken (“<abstract_file>”) and the position of the abstract in the MEDLINE file (“<abstract_file_position>”) are given. This ensures that every abstract can be identified via those three identifiers later. As they were included for testing and debugging, these numbers are not used when querying CONAN.

In the “method_overview”, the method used to extract the information (“<method>”, can be either BLAST, AbGene etc.) and what the parameters were used in this round of extraction (“<parameters>”). In the files processed so far, the parameters are all set to “standard”. Different parameters can be used in the BLAST search.

The “abstract_body”, finally, contains all the information extracted. As this differs from method to method, I describe the layout for every method separately.

8.4.2 BLAST

A typical BLAST-entry looks like this:


```

<term_id>0</term_id>
<term_text>receptor tyrosine kinase family
<term_pos_start>54</term_pos_start>
<term_pos_end>85</term_pos_end>
<term_tag>Protein</term_tag>
</term_text>

```

A pattern is repeated for each term that was found by the BLAST method. It is assigned a “term_id”, which is a running identifier in each abstract. The tag “term_text” consists of several sub-tags. Firstly, the extracted term is stored in “term_text” (e.g. receptor tyrosine kinase family). Secondly and thirdly, the position in the abstract where the term begins and where it ends are given in “<term_pos_start>” and “<term_pos_end>” (e.g. 54 and 85). Finally, the database from which this term was extracted is given in “<term_tag>” (e.g. protein).

8.4.3 AbGene

```

<term_id>0</term_id>
<term_text>ErbB-4</term_text>

```

Similar to the BLAST entry, a unique identifier is placed in “<term_id>”. For every protein name found in the text, this name is stored in the XML file as “<term_text>”. A typical AbGene-entry looks like this:

8.4.4 MuText

```

<mutation>T47D
<pos>490</pos>
</mutation>

```

For every mutation found in text, the mutation itself is stored in the “<mutation>”-tag, and the position in the abstract where this mutation occurs is stored in the sub-tag “<pos>”. This enables verification by the Data Integrator (see Section 8.3.1). A typical MuText-entry looks like this:

8.4.5 NLProt

For NLProt, the structure of the XML document is quite complex. A typical entry would look like:

```

<term_text>ErbB-4
<term_pos>1</term_pos>
<term_organism>human</term_organism>
<term_score>0.950</term_score>
<term_synonyms>ERB4_HUMAN</term_synonyms>
<synonym_score>100</synonym_score>
<ensembl_name>ENSG00000178568</ensembl_name>
<goa_id>0016740</goa_id>
<goa_function>transferase activity</goa_function>
<goa_id>0008283</goa_id>
<goa_process>cell proliferation</goa_process>
<goa_id>0005887</goa_id>
<goa_component>integral to plasma membrane</goa_component>
<goa_id>0005524</goa_id>
<goa_function>ATP binding</goa_function>
<goa_id>0006468</goa_id>
<goa_process>protein amino acid phosphorylation</goa_process>
<goa_id>0007169</goa_id>
<goa_process>transmembrane receptor protein tyrosine kinase signal-
ing pathway</goa_process>
<goa_id>0004714</goa_id>
<goa_function>transmembrane receptor protein tyrosine kinase
activity</goa_function>
<goa_id>0004872</goa_id>
<goa_function>receptor activity</goa_function>
<goa_id>0005006</goa_id>
<goa_function>epidermal growth factor receptor
activity</goa_function>

```

For every term (protein name) found, an extra “<term_text>”-tag is opened in the “abstract_body”. There, the protein name found is reported. There are several sub-tags of the “<term_text>”-tag.

- “<term_pos>” describes the position of the term in the abstract.
- “<term_organism>” describes the organism of which the protein is thought to belong to. This is estimated by NLProt.
- “<term_score>” is the score assigned by NLProt to the protein term.
- “<term_synonyms>” is the UniProt synonym of the protein name found. This is estimated by NLProt.

- “<synonym_score>” is the score assigned to the UniProt synonym, ranging from 0 to 100.
- “<ensembl_name>” is the Ensembl identifier assigned to the term via the UniProt synonym, using the IPI resource.

For the GO identifiers, which are derived by the UniProt synonym and the GOA resource, the number of the GO entry is stored under “<goa_id>”. The GO description, depending on which category it belongs to, is described in “<goa_function>”, “<goa_process>” or “<goa_component>”

8.4.6 PreBIND

For the PreBIND entries, the structure is quite simple:

```
<interaction>
<Protein1>Cyclin B</Protein1>
<Protein2>p42 MAPK</Protein2>
<Interaction_Type>24</Interaction_Type>
</interaction>
```

For each interaction, an “<interaction>”-tag is introduced. Inside this tag, the names of Protein 1 and Protein 2 are reported (“<Protein1>” and “<Protein2>”). As additional information, the number of the regular expression that fitted the pattern is recorded (“<Interaction_Type>”). Using this information, the Interaction Type can later be displayed.

8.4.7 MeSH

The simplest structure inside the XML document is for the MeSH terms:

```
<term_text> Animals </term_text>
```

In each “<term_text>”-entry, one MeSH term extracted from the MEDLINE input file is saved.

8.5. Boosting

After the XML file is produced, the protein names are checked and filtered by the Boosting Classifier. This is done at this specific point of time because of time reasons. The Boosting Classifier takes the same amount of time for all sizes of lists of protein names, no matter if the list is only 5 names long or 5,000 names long. Therefore, the classifier is more or less a “post-processing” filtering step.

All protein names found by all three NER methods (BLAST, AbGene, NL-Prot) are extracted and stored in a separate file. This file serves as an input for the classifier. The output of the classifier is exactly the same list, but then with “correct” and “false” attached to the protein name. From this output list, the “false” names are changed in the original CONAN-XML-file. The original text is subsequently changed (e.g. “chaperonins” becomes “chaperonins - false”). The same is done for interactions, where this protein name appears, the interaction is tagged as “false” as well.

This is done because the results should still be visible for the user when querying for results, but the user also should get the information that this might be a false positive. In the web-visualization of the results, this is also done by adding other colors to the output.

After all these steps have been finished, the Output XML file is done and ready for querying and for user output. This is described in Chapter 11 of this thesis.

References

- [1] R. Malik, L. Franke, and A. Siebes. Combination of text-mining algorithms increases the performance. *Bioinformatics*, 2006. In Press.
- [2] R. Malik and A. Siebes. Conan: An integrative system for biomedical literature mining. In C. Bento, A. Cardoso, and G. Dias, editors, *LNAI 3808, EPIA05*, pages 248–259, 2005.

PART IV

EXPERIMENTS

Chapter 9

MEASURES IN TEXT MINING

In this chapter, I explain the measures which are important when evaluating a text mining system. These measures will be used throughout the evaluation.

9.1. Precision and Recall

Precision and recall are the basic measures used in evaluating search strategies. These measures assume:

- That there is a set of records in the database that is relevant to the search topic
- That records are assumed to be either relevant or irrelevant (these measures do not allow for degrees of relevancy).
- That the actual retrieval set may not perfectly match the set of relevant records.

In Figure 9.1, the notion of recall and precision becomes clear. There are four different sets: The records which were retrieved, the records which were not retrieved, the relevant records and the irrelevant records (as annotated in the test set). The intersections of these sets (A,B,C,D) represent the following: A is the number of irrelevant records not retrieved (true negatives), B is the number of irrelevant records retrieved (false positives), C is the number of relevant records not retrieved (false negatives) and D is the number of relevant records retrieved (true positives).

- Recall is defined as:

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

When using the Figure, this equation would translate to: $Recall = \frac{D}{D+C}$.

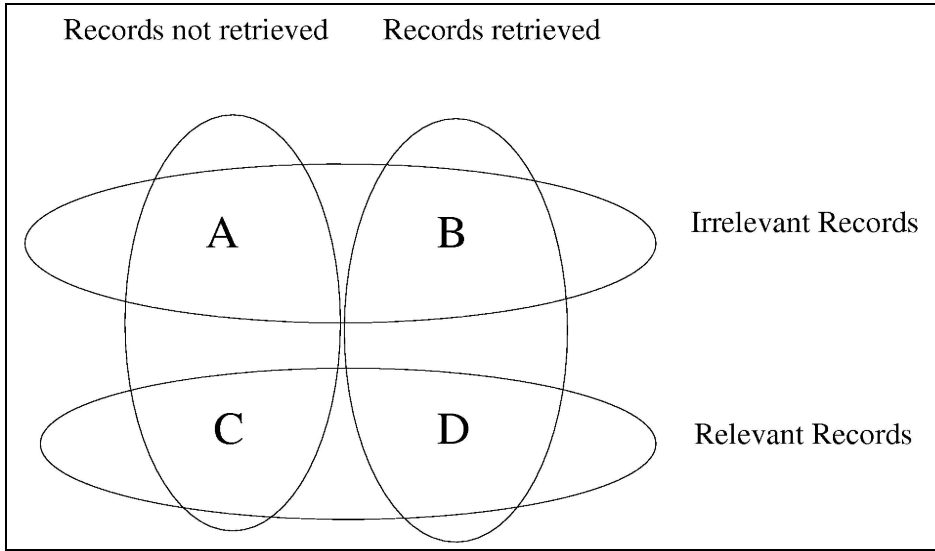


Figure 9.1. Definition of Recall and Precision

- Precision is defined as:

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

When using the Figure, this equation would translate to: $Precision = \frac{D}{D+B}$

Recall and precision are often inversely related, meaning that, in an actual text mining system, the precision often goes down when the recall goes up and vice versa. This relation is displayed in Figure 9.2.

As noted earlier, records must be considered either relevant or irrelevant when calculating precision and recall. Obviously, records can exist which are marginally relevant or somewhat irrelevant. Others may be very relevant and others completely irrelevant. This problem is complicated by individual perception: what is relevant to one person may not be relevant to another. This is also important when considering “partial matches” (also see Chapter 3).

Measuring recall is difficult because it is often difficult to know how many relevant records exist in a database. As measuring recall is often so difficult, annotated corpora have been constructed for text mining purposes, making the estimation of recall easier. Recall is then computed by looking at the annotated pool of relevant records and then determining what proportion of the pool the search retrieved.

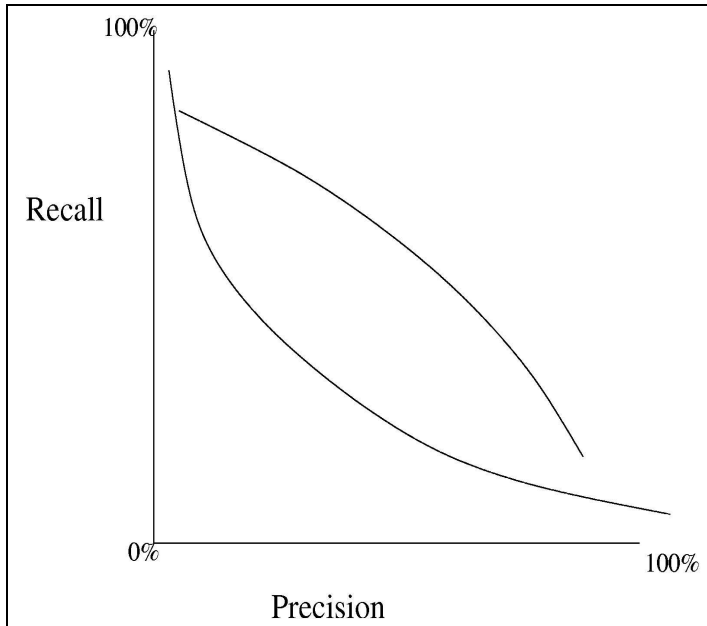


Figure 9.2. Recall-Precision Graph

9.2. F-measure

An often-used measure in information retrieval and NLP is the so-called F-measure. This measure was first introduced by Rijsbergen [16] and it combines Recall (r) and Precision (p) with an equal weight in the following form:

$$F(r, p) = \frac{2rp}{r + p} \quad (9.1)$$

In fact, the F-measure is the harmonic mean between precision and recall. The Harmonic Mean H of numbers x_1, \dots, x_n is given by

$$\frac{1}{H} = \frac{1}{N} \sum_{i=1}^n \frac{1}{x_i} \quad (9.2)$$

When applying this formula to precision and recall, we get:

$$H = \frac{1}{\frac{1}{2}(\frac{1}{r} + \frac{1}{p})} = \frac{2}{\frac{r}{rp} + \frac{p}{rp}} = \frac{2rp}{r + p} \quad (9.3)$$

So we see that the F-measure is a harmonic mean. If we multiply Equation 9.2 by H on both sides, we get:

$$\frac{1}{N} \sum_{i=1}^n \frac{H}{x_i} = 1 \quad (9.4)$$

In other words, the average of ratios between the Harmonic mean and the data points is unity. This means that every data point contributes to the harmonic mean equally. As the two data points are recall and precision, those two measures contribute equally to the F-measure, no matter how large those numbers are.

The F-measure is becoming more and more important in the field of text mining. As already mentioned in Chapter 3, some text mining systems achieve a very high recall, others a very high precision. When reporting a high F-measure in publications about text mining, one can be sure that the system achieves a high recall as well as a high precision.

9.3. Inter Annotator Agreement

The Inter-Annotator agreement scores are measures that are used when constructing a corpus. Normally, more than one annotator annotates the abstracts or sentences when building the corpus. This also leads to the fact that computing Precision and Recall for a given corpus is not straight-forward, as the annotators might disagree on a certain term (e.g. a certain protein name). For this reason, the Inter-Annotator Agreement Scores help to see if a given corpus is constructed well.

These agreement scores were first used in interpreting medical images (e.g. X-ray photographs) [11]. In constructing a corpus, such scores are used to show how good the different annotators agree with each other. There are some different measures:

- p_{pos} , the positive agreement.
- p_o , the overall proportion of agreement.
- p_{neg} , the negative agreement.
- p_e , the agreement expected by chance.
- $kappa$, the measure of agreement corrected by chance.

These measures are best explained by an example: Suppose the annotators want to find disease names in abstracts. The total number of abstracts is 150.

	Annotator 2		
Annotator 1	Positives	Negatives	Total
Positives	7	10	17
Negatives	12	121	133
Total	19	131	150

Table 9.1. Example for Inter-Annotator Agreement

- p_{pos} is the number of positives that both annotators agree on, divided by the number of all positives for both annotators. In our example, the calculation is: $p_{pos} = \frac{7+7}{(10+7)+(12+7)} = 0.39$
- p_{neg} is calculated in a similar way. $p_{neg} = \frac{121+121}{(10+121)+(12+121)} = 0.92$
- p_o is calculated by the number of positives both readers agree on plus the number of negatives both readers agree on, divided by the total number of abstracts. $p_o = \frac{7+121}{150} = 0.85$
- The joint agreement expected by chance (p_e) is calculated for each combination. $p_e = (\frac{17}{150} * \frac{19}{150}) + (\frac{133}{150} * \frac{131}{150}) = 0.79$
- $kappa$ is calculated by subtracting the portion of the annotations which are expected to agree by chance from the overall agreement, and dividing the remainder by the number of cases on which agreement is not expected to occur by chance.

$$kappa = \frac{p_o - p_e}{1 - p_e} \quad (9.5)$$

In this example, $kappa = \frac{0.85-0.79}{1-0.79} = 0.31$

Normally, only the $kappa$ score is provided. $kappa$ has the big advantage that it is corrected by chance. In the paper by Landis [12], a guideline for the strength of the $kappa$ -values is given.

$kappa$ value	Strength of Agreement
<0	Poor
0-0.20	Slight
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Substantial
0.81-1.00	Almost Perfect

Table 9.2. Guideline for the Strength of the $kappa$ -value

Dingare et. al describe in their publication [7] that inter-annotator agreement in the biomedical domain is normally in the range of 87% and 89%. This number corresponds to p_o . This number normally is higher for corpora that do not belong to the biomedical domain. This is due to the fact that the biomedical domain requires a lot of expert knowledge and the language is quite different from “normal” text. In my point of view, the $kappa$ value gives a much stronger insight on how well the corpus is constructed, as it removes the agreement by chance.

Chapter 10

RESULTS

In this chapter, I present the results obtained by CONAN. I will firstly present a short overview of how fast the system is performing, then I show how such a system is evaluated and how CONAN performs in such evaluations. Finally I present some biologically interesting results.

10.1. Experimental Setup

For a complete system like CONAN, it is important to know how long processing the PubMed/MEDLINE files takes. As we ultimately want to cover the whole MEDLINE, we need to know how long this is going to take. All experiments that are shown in this thesis, were conducted on an Intel Pentium 4, 2 GHz, 512MB RAM, running SuSE Linux 8.3. In a first test of the reliability, stability and speed of the system, 100,000 articles published on PubMed/MEDLINE were processed. These files were medline04n0576.xml - medline04n0594.xml (from 2004), including articles from every field. The whole collection of articles is approximately 745 Megabytes; this includes not only the abstracts, but all other information given in the PubMed/MEDLINE files. The computation of the results of all those articles took about two weeks. MEDLINE holds about 13 Million citations at the moment. With this standard PC, the calculation of the whole MEDLINE would take five years. With a faster CPU and more memory, however, this number can be set to less than 1 year for processing the whole MEDLINE. On a cluster, this computation can be done even faster. The size of a CONAN output file is in the range of 15 - 45 MB, depending on the amount of information.

In a second round of testing, we extracted only the protein names and interactions from 160 MEDLINE-input-files (medline05n0565.xml-medline05n0749.xml, from 2005), consisting of 250,695 abstracts from different fields. This resulted in a list of 112,546 interactions. Further experiments show that from those

112,546 interactions, approximately 80,000 interactions are unique. The extraction of this information took five weeks. It has to be said that only interactions were taken into account where both proteins are clearly assigned an Ensembl code. When adding these interactions with the interactions discovered in Experiment one, we get a total number of approximately 160,000. These interactions are of high-quality as we will show in Sections 10.2.5 and 10.2.6.

A simple query (e.g. searching for protein p21) on one CONAN output file takes about 20 seconds on the standard PC. On the webserver we are currently using, the same query takes only 3 seconds. As the query time for more CONAN output files is almost linear, querying the whole MEDLINE would take a very long time. Solutions have to be found in the future to solve this scalability problem.

10.2. Evaluation

In all scientific fields, a method has to be evaluated. Evaluation means that a certain procedure or process is tested against certain standards (in this case: corpora) and the outcome of this testing procedure is later translated into numeric values (Precision, Recall, F-Measure). This translation is done by the statistical measures that were already introduced.

As stated in the Introduction, partial matches are a big problem when evaluating a text mining method. For the following evaluations, we both evaluated the system in “SLOPPY”-mode (partial matches count as positives) and “STRICT”-mode (partial matches count as negatives).

The evaluation strategy used in all evaluations is the following:

- Pre-Processing: Download abstracts that are part of the corpus. When needed, extract specific sentences of those abstracts. Some corpora use single sentences instead of whole abstracts.
- Processing: Let CONAN process all abstracts/sentences.
- Post-Processing: Convert the output in the right format. For some evaluation corpora, automated scripts exist that automatically compute the evaluation results.
- Evaluation: Manually evaluate the results of CONAN by looking through the abstracts/sentences and comparing the results to the “true” results. Sometimes this also can be done automatically. The outcome of this step will be the number of TP, FP, FN and TN.
- Computation of Measures: Compute numbers for Precision, Recall and F-measure, for both SLOPPY and STRICT modes.

10.2.1 Corpora

A corpus is by definition a collection of documents. In linguistics, a corpus is a large and structured set of texts, now usually electronically stored and processed.

In the text mining community, a great number of corpora have been developed over the last years, mainly for testing and evaluation of systems. Most of the corpora, however, completely focus on the NER element of the text mining process. At this time, the development of corpora for other steps in the process (e.g. IE), is ongoing, but there is only one very small corpus for the evaluation of PPI extraction.

In the next sections I describe firstly how such a corpus is constructed, using the construction of the Prodisen Corpus as an example. Furthermore, I introduce other corpora that are interesting in the field of biomedical text mining. For this thesis, I also constructed a new corpus that covers both the NER aspect and the IE (i.e. PPI extraction) aspect of text mining that is called "LDD Corpus" (see Section 10.2.6). For all those corpora, I present the evaluation of CONAN.

10.2.2 Prodisen

To complement existing resources and to help in the efforts to bridge the gap between IR approaches and text mining, we developed the Protein descriptions in sentences (Prodisen) corpus [10].

10.2.2.1 Construction

The starting point for construction of Prodisen was the selection of the articles to be included in the corpus. As only a fraction of articles in MEDLINE have molecular biology or gene descriptions as their topic, those abstracts have to be found. One possibility would be to take a subset of MEDLINE and use a NER method to get all articles that have at least one gene/protein name in them.

This approach proved not to be successful, because many important articles are missed, as NER methods are not 100% reliable. Moreover, this approach takes quite a long time to identify enough abstracts to build a corpus. Another problem in this approach are the so-called "anaphora". For a description of Anaphora, refer to Section 2.4.1. Anaphora are a problem in the construction of this corpus, because the gene descriptions are associated to a certain gene/protein. When this gene is not mentioned in the sentence, but an anaphoric expression is, the NER method puts this sentence into the wrong category, because it does not find any protein names in the text.

We used another approach to construct this corpus. Firstly, selected MEDLINE abstracts (see below) were split into sentences. Then, two domain experts were presented with the different sentences together with the correspond-

ing abstracts. They had to classify each abstract into three categories: (Y) the sentence is useful as gene description, independent of whether the gene name is mentioned or a referential expression is used (based on the abstract context); (N) the sentence does not correspond to a gene description or (D) uncertain ambiguous cases, where the expert was in doubt.

The sentences containing gene description information were additionally classified in groups containing information on relevant aspects of a gene, gene product, gene group, protein family or protein domain based on the analysis of the contextual information. Those cases include descriptions where the gene names appear as well as cases where referring expressions were used or it could be inferred that the sentence contains a relevant gene description (based on the context). The classes considered include:

- Descriptions related to molecular functions.
- Descriptions related to biological processes.
- Descriptions which refer to descriptions of cellular location.
- Descriptions related to associations to diseases, symptoms or treatments.
- Descriptions referring to interactions, e.g. protein interactions and dimerization or protein-compound interactions.
- Information related to the gene expression (e.g. in which tissues a given gene is expressed).
- Descriptions related to homology information.
- Descriptions of sequence and structural features (including mutations, protein family, isoforms, post-translational modifications, SNP, chromosome mapping).
- Other useful gene descriptions such as information related to phenotypes, experimental usage (markers) and enzyme kinetics.

All these basic types of gene descriptions refer to different relevant aspects that characterize genes, proteins and protein families. They represent the diversity of annotation information stored in different biological annotation databases. In practice single gene description sentences are often not limited to a single description aspect, but provide several characterization types. The annotators had to categorize the sentences in every category that is applicable, so categorization of one sentence into multiple classes is possible.

As stated before, only a fraction of abstracts contained in MEDLINE are related to Molecular Biology and of those only some sentences are related to gene descriptions. Thus, to address the detection of gene description sentences from the whole PubMed collection and to estimate the total amount of

functional description sentences in PubMed, we constructed a set of randomly selected abstracts.

Table 10.1. Prodisen corpus in numbers

Corpus	No. of Sentences	No. of Abstracts	No. of Words	Positives	Negatives	Uncertain
Random Corpus	10,039	1,234	224,890	1,704	7,899	436
Enriched Corpus	11,125	1,232	244,549	7,693	2,949	483
All	21,164	2,466	469,439	9,397	10,848	919

We have classified each of those sentences into one of the three previously described categories. As shown in Table 10.1, the Prodisen random corpus contains over 10,000 sentences, of which around 15 percent have been classified as gene description sentences. Taking into account that there are over 7 million PubMed entries that contain abstracts in English, and that their average length is around 9 sentences, one would expect around 9,5 million sentences to be in PubMed containing gene description related information.

In order for a text corpus to be useful, for instance as a training/test set for text classifiers, it is important to have a balanced set of positive and negative training cases [5]. As the proportion of positive cases, i.e. sentences corresponding to gene descriptions, is relatively small in the random Prodisen corpus, we use a strategy to construct an enriched set of abstracts in terms of gene descriptions. This strategy is based on PubMed article citation overlap between different biological annotation databases, each of them with a different focus regarding the gene description type.

To have a collection of abstracts that covers all previously defined relevant gene description types we first extracted for each database the number of (non-redundant) PubMed articles used for their annotations. Then we identified the articles that were used as citations by several different databases. Table 10.2 illustrates the binary overlap between the citations extracted for each of the biological databases. The core set of the enriched Prodisen corpus consisted in the articles cited by the following biological databases: Uniprot, OMIM, GeneRif and GOA. Additional articles cited in GOA and PDB, Pfam or IntAct were included. The resulting Prodisen enriched corpus contained over 11 thousand sentences with over 7 thousand gene description sentences (see Table 10.1). The sentences of this enriched corpus were again classified and annotated by the domain experts.

Table 10.2. PubMed article usage overlap between different biological databases.

Database	GOA	GeneRif	UniProt	OMIM	PDB	IntAct
GOA	29,248	3,972	15,409	9,465	135	764
GeneRif	3,972	84,380	4,890	6,637	620	283
UniProt	15,409	4,890	112,476	19,859	5,061	764
OMIM	9,465	6,637	19,859	88,766	296	193
PDB	135	620	5,061	296	11,790	35
IntAct	764	283	764	193	35	1184

The construction of such a corpus is not a simple task. However, once such a corpus is constructed, its usability is large. We can foresee many potential uses of the Prodisen corpus in the training and testing of information extraction techniques dedicated to the identification of gene/protein descriptions in text. It can be a useful resource for both, bag of words based approaches focusing on word frequencies, as well as for the discovery of description patterns. Moreover, the corpus can be used to derive contextual word frequencies for protein mention discovery (disambiguation).

10.2.2.2 Evaluation

To estimate the difficulty of identifying gene description relevant sentences, the measurement of annotator agreement is useful. Many of the existing evaluations of text mining and information extraction strategies in biology were based upon data sets lacking such agreement measures. Recently carried out community-wide evaluation efforts, especially the BioCreAtIvE contest addressed the importance of measuring inter-annotator agreement [4].

This is especially true for domain specific literature, such as the molecular biology literature. Authors of biomedical articles make presuppositions on the background knowledge of readers from the targeted scientific community. This is reflected within their writing style. Also the context of a given sentence is crucial in order to make inference of its underlying semantics. The agreement scores for both Prodisen sets, regarding the three classification types are provided in Table 10.3. The description of the Agreement Scores can be found in Section 9.3.

Agreement Index	Description	Random	Enriched
P(o)	Overall	0.856	0.705
P(pos)	Positive	0.656	0.799
P(neg)	Negative	0.928	0.306
P(e)	Chance	0.662	0.490
Kappa	Corrected	0.574	0.421

Table 10.3. Agreement indices of the polytomous ratings of the two Prodisen data sets.

The interpretation of how good the kappa-value is, can be seen in Table 9.2. Here we see that the *kappa* scores of the two corpora are good, but not overwhelming. This is probably due to the fact the gene description sentences are highly subjective and even if the annotators have good biological domain knowledge, their interpretation of the sentence or abstract might be really different. Better annotations guidelines will be used in the further development of the Prodisen Corpus, which should lead to higher agreement scores. CONAN will be evaluated on the Prodisen Corpus once the final version of Prodisen is released.

Fortunately, there are already some well-used corpora in the text mining community that are freely available. Unfortunately though, most of them do not give scores on inter-annotator agreement. In the next few sections I describe these corpora and present the evaluation of CONAN on these corpora, reporting Precision, Recall and F-Measure in the process.

10.2.3 BioCreative Corpus

BioCreative (Critical Assessment of Information Extraction systems in Biology) [1] is a competition for text mining applications, first held in 2004. As in other fields in bioinformatics and data mining, like structure prediction, competitions are common (e.g. CASP [2]). The time was ripe for such a competition in the text mining field as well. As CONAN was not yet functional when the competition was held, CONAN was evaluated afterwards by the same standards.

BioCreative consists of two different tasks: Entity Extraction (NER) and functional annotation of proteins (IE). In the evaluation, I will focus on Task 1, meaning that the NER part of CONAN is evaluated. The task requires the identification of terms in a biomedical research article that are gene and/or protein names. The BioCreative corpus consists of training data (7500 sentences) and evaluation (test) data, which consists of 5000 sentences. It was explicitly stated beforehand that the training data contained approximately 9000 gene/protein name mentions and the test set contained roughly 6000.

Using the common evaluation strategy (see above), we came to the evaluation results listed in Table 10.4. The scores of the anonymous groups are taken from [18]. Please note that we consider the scores of the “open” evaluation. Here, the groups were allowed to use external resources. This seems appropriate when comparing the results to those of CONAN.

Table 10.4. Overview of Results on BioCreative Corpus

Group	Precision	Recall	F-measure
CONAN without Boosting	79.8%	80.1%	79.9%
CONAN with Boosting	82.2%	78.9%	80.5%
A	84.1%	81.4%	82.7%
B	75.1%	81.3%	78.1%
C	86.4%	78.7%	82.4%
D	80.1%	81.8%	80.9%
E	82.3%	74.1%	78.0%
G	73.8%	79.9%	76.7%
H	63.2%	70.5%	66.7%

To further compare the results to other groups, the organizers of BioCreative support the user with a quartile-system, as can be seen in Table 10.5. A quartile is by definition a segment of a sample representing a sequential quarter (25%) of the group.

Table 10.5. Quartile Segments in BioCreative

F-Measure	Quartile
0.83-1	Top
0.81-0.83	Q1
0.78-0.81	Median
0.57-0.78	Q3
0.25-0.57	Low

The results achieved by CONAN with Boosting can be placed in the first quartile (Q1), whereas CONAN without Boosting falls into the Median category. This means that CONAN, although three groups perform slightly better, still performs as well as the top-ranked contenders in this experiment. When comparing the results to those on other corpora (see below), the results are in the same range. The exact scores, of course, depend heavily on the data set.

It is also clear from the results in Table 10.4 that the Boosting Classifier improves the Precision of CONAN, but the Recall goes down. The reason is that some TPs, although not very many, are categorized wrongly by the Boosting classifier.

10.2.4 YAPEX Corpus

Table 10.6. Overview of Results on YAPEX-Corpus-SLOPPY

Group	Precision	Recall	F-measure
NLProt	-	-	85%
GAPSCORE	81.5%	83.3%	82.5%
YAPEX-method	83.3%	82.1%	82.9%
KeX	82.1%	83.5%	82.8%
CONAN without Boosting	88.92%	85.33%	87.08%
CONAN with Boosting	89.65%	85.26%	87.40%

Table 10.7. Overview of Results on YAPEX-Corpus-STRIC

Group	Precision	Recall	F-measure
NLProt	-	-	75%
GAPSCORE	56.7%	58.5%	57.6%
YAPEX-method	67.8%	66.4%	67.1%
KeX	40.4%	41.1%	40.7%
CONAN without Boosting	79.1%	77.2%	78.1%
CONAN with Boosting	79.6%	76.9%	78.2%

The YAPEX-corpus, published in the year 2002 [8], consists of 101 MEDLINE abstracts annotated by domain experts connected to the YAPEX project. The corpus is divided into two parts, the first part being the result of a PubMed query, consisting of 48 abstracts, the second part being a randomly chosen subset of 53 abstracts of the GENIA corpus [6]. The YAPEX corpus focuses on the extracting of protein names out of text. It contains 1,890 annotated protein names.

The evaluation results can be seen in Table 10.7 for STRICT evaluation and in Table 10.6 for SLOPPY evaluation. Here it becomes clear that CONAN performs very well in both the SLOPPY and the STRICT evaluation. In both, CONAN outperforms all other groups or methods that were evaluated on the YAPEX corpus. Two other points are interesting: firstly, the Boosting Classifier improves the performance of CONAN. Secondly, by combining different classifiers and integrating several data sources, CONAN performs better in this task than NLProt alone does.

10.2.5 LLL Challenge Corpus

The LLL-Challenge-Corpus, named after the LLL-challenge which took place in course of the Learning Language in Logic Workshop (LLL05) in 2005, focuses on the evaluation of interaction extraction from text. It consists of two

parts: The training set consisting of 57 sentences derived from MEDLINE-files and the test set consisting of 87 sentences. Both sets contain only *Bacillus subtilis* protein names and interactions, focusing specifically on transcription in the bacterium. *Bacillus subtilis* was chosen because it is a model bacterium and because transcription is both a central phenomenon in functional genomics involved in gene interaction and a popular IE problem. There are two types of sets, the so-called “basic data set”, including sentences, word segmentation and biological target information, and the “enriched data set” which also includes lemmas and syntactic dependencies. We focused on the “basic data set”.

The evaluation results can be seen in Table 10.8. This shows that CONAN outperforms all other systems evaluated on this corpus. However, it has to be said that the LLL corpus is very small, making the comparison of the results not significant. Nevertheless, the performance of CONAN on this set and the results are encouraging.

Table 10.8. Overview of Results on LLL-Challenge Corpus as reported in [14]

Group	Precision	Recall	F-measure
1	50%	53.8%	51.8%
2	10.6%	98.1%	19.1%
3	37.9%	55.5%	45.1%
4	25.0%	81.4%	38.2%
CONAN	53%	52.1%	52.49%

As the LLL corpus is only a very small corpus, the need for another corpus that includes PPis is big. So I decided to produce another corpus that includes NER evaluation as well as evaluation of PPis. I named this corpus the LDD corpus (for Large Distributed Databases, the group where CONAN was developed).

10.2.6 LDD Corpus

In order to further analyze interaction data and protein name finding, the LDD corpus of 1,768 abstracts has been created, all of these containing one or more interactions. A high percentage of MEDLINE abstracts do not contain any interactions and protein names at all. This is similar to the problem in the construction of the Prodisen Corpus in Section 10.2.2.1. In this case, we solved the problem by selecting these 1,768 abstracts, a combination of available lists of PMIDs from BIND [3] and DIP [17], which are both databases holding information about PPis. This gives us the guarantee that at least one interaction is in the abstract, which also implies that at least two different protein names are mentioned in the abstract. This solution is also similar

to the construction of Prodisen, but because we wanted to include as many interactions as possible, other data sets than in the Prodisen construction were selected.

Because this set is not or only partially annotated, there was a need to annotate these abstracts by hand. One hundred of those 1,768 abstracts have been selected completely at random by us to ensure that no organism or protein family is overrepresented and those interactions have been manually annotated. 100 abstracts might seem quite a small number, but manually annotating over 1,000 abstracts would have been a too-big workload for one person. Also, manually evaluating the results of our system on 1,768 abstracts would have taken quite a substantial amount of time.

10.2.6.1 Constructing the LDD Corpus

Two annotators with biological domain knowledge were presented with the data of the LDD corpus. Their task was to identify Protein/Gene-names mentioned in the not-annotated LDD Corpus file. The guidelines given to the annotators are the following:

- Protein and/or Gene Names are positives
- Protein Family names are positives
- Protein Domains or Motifs are negatives and remain untagged

The annotated Proteins had to be tagged with a “<Protein>”-tag in the empty LDD corpus file.

```
<Protein>HNF-1alpha</Protein>
```

After this first round of annotation, the second round of annotation included the annotation of protein-protein interactions.

The guidelines for this second round were the following:

- Direct (physical) interactions are positives
- Indirect interactions (e.g. Phosphorylation) are positives
- Anaphora are not to be taken into account
- Interactions of Proteins with DNA are negatives and remain untagged

For annotating the interactions, I constructed a file containing all binary relations in the corpus. An example can be seen below. It ensures that everybody using this corpus for test purposes can easily see what interactions can be extracted and can use their own format to evaluate.

>10934475
mPAR-6 Cdc42

10.2.6.2 Results

When calculating the agreement scores, I calculate them per token. This means that each token was checked separately, e.g. when one annotator tagged the phrase “KH domain protein MEX-3” as a protein name and the second annotator tagged only “MEX-3”, it is not counted as a full negative. As this phrase consists of four tokens, three tokens (KH domain protein) are counted as negatives and one (MEX-3) as a positive. By this calculation, we also tackle the problem of Partial Protein Matches.

For the evaluation of constructing the corpus, similar to constructing the Prodisen corpus (see Section 10.2.2.2), the measure of Inter-annotator agreement (see Section 9.3) is used. Moreover, I give the F-measure for the inter-annotator agreement as well. For the calculation of the F-measure, one annotator’s results is assigned the gold standard, meaning that the annotation of one annotator is labelled as completely true. Hachey et al. [9] report that the F-measure is symmetric in respect to the gold and test set. The F-measure is symmetric since $\text{recall}(A,B)=\text{precision}(B,A)$ and the F-measure is the harmonic mean between those two. The results of this evaluation can be seen in Table 10.9.

The LDD corpus consists of 17,810 tokens in total. From these, only 951 tokens were categorized differently between the annotators.

Table 10.9. Inter-Annotator Agreement Results

Score	Resulting Value of 2 Annotators
p_{pos}	0.8862
p_o	0.9466
p_{neg}	0.9651
p_e	0.6407
$kappa$	0.8460
Recall	0.9116
Precision	0.8622
F-Measure	0.8862

For the Interaction Tagging, the agreement was almost 100%. Out of the 427 interactions reported in the corpus, only 10 interactions were not tagged by both annotators, so both annotators agreed on 97.7% of all interactions. These 10 cases arise because of the different annotation of some protein names in text. It has to be said that annotating interactions is an easier task than

annotating protein names. Interactions are often visible immediately upon reading the text carefully and therefore the error rate is much smaller.

When looking at the scores, it can be said that the agreement between the annotators is high, though not perfect. One major source of different annotation is the protein Actin. As it is normally tagged as a protein name, the term “actin filaments” is not tagged, although actin filaments are just polymers of the protein actin. As the name “actin” appears 32 times in the corpus, this is one major source of different annotation. It is also responsible for different annotations in the Interaction Tagging.

The *kappa* score of 0.8460 shows that the annotator agreement is placed in the category “Almost perfect”. Moreover, the F-measure of 88.62% shows that the annotator agreement is in the range of the expected results. As the inter-annotator agreement for NER is normally in the range from 87% to 89%, our resulting F-measure can be placed in the top range.

In conclusion, the LDD corpus is well-annotated and will become a valuable resource to the community.

10.2.6.3 Protein Tagging

We used the LDD test-set of 100 manually annotated articles. When analyzing the three Protein-name-tagging methods (BLAST, AbGene, NLProt), we see that the Boosting Classifier boosts the performance of those methods. The results are shown in Table 10.10. We cannot show a comparison to other systems, because no system except CONAN was evaluated on the LDD corpus so far.

Table 10.10. Evaluation of CONAN on LDD Corpus

Method	Recall	Precision	F-Measure
CONAN	80.9%	85%	82.90%
CONAN with Boosting	79.5	87.2	83.17%

The LDD data set has a small bias towards yeast-related articles. As reported in [13], text mining methods usually perform better on yeast-related articles than on other organisms, because protein-naming is much simpler in yeast than it is in other organisms.

The good result of Protein Tagging by CONAN is also reflected in the PPI evaluation (see below), because interaction finding is highly dependent on good protein name finding.

It is also interesting to see that the F-Measure of 83.17% is only slightly smaller than the F-Measure of 88.62% achieved by the annotators. This means that CONAN is almost as good as a trained biologist, who has to finish a long study to achieve this accuracy.

10.2.6.4 Interactions

Finally, the protein-protein interactions were evaluated. It is important to say at this point that our method does discriminate between positive interaction and negative interactions (inhibitions). In this evaluation, we consider both positive and negative interactions as true positives.

In the 100 manually-annotated abstracts, a total of 427 interactions are documented. Those 427 interactions were manually annotated. CONAN found 477 interactions in total, compared to the 427 interactions that were annotated manually in the abstracts, this yields a number of 50 or more false positives. Analyzing the abstracts resulted in a precision of 81.55% (389/477) and a recall of 91.10% (389/427), resulting in an F-Measure of 86.06%. Here we see that, we get very good results by using our system, detecting almost all available interactions mentioned in the abstracts.

An important observation is that the missed interactions were missed because of the performance of the protein tagging methods. As stated before, the lists produced by these methods are passed on to the interaction-finding method. If these lists do not contain certain protein names, then of course interactions containing these proteins can not be found. Moreover, when false positive protein names are included in the list, false positive interactions are found. An example from the LDD corpus is the abstract with PMID 10966642. Here, the term “MODY3” is tagged as a protein name and is passed on to the interaction finding. Here it is found that it interacts with “DCoH” and “HNF-1alpha”. MODY3 looks like a protein name, but it is a disease, namely “Maturity-onset diabetes of the young type 3”. Mutations in HNF-1alpha and its coactivator DCoH lead to the phenotype. So it becomes clear that wrong protein names which are passed on to the Interaction Finding method, are a big source of false positives in the interaction finding.

10.2.7 Boosting Evaluation

The Boosting Classifier was tested separately because I wanted to know how well the classifier performs on data. We have already seen before that Boosting improves the performance of CONAN. We also want to know why Boosting improves the results significantly. We tested the Boosting classifier on a test set, containing 5113 protein names. This test set was a MEDLINE-input file that was processed by CONAN. Those 5113 protein names were sent to the classifier which labeled 4917 protein names correctly (96.17%) and 196 wrongly (3.83%). The classifier was able to filter out 120 false positives (e.g. motheaten, desmosterolosis).

It is important to know what percentage of protein names that are classified as “positives” and “negatives” by the Boosting classifier, are in the dictionary. This gives a rough estimate of how good the classifier performs on “unseen”

examples. There are two classes of “unseens”, namely “partial unseens” and “full unseens”. A protein name is “partial unseen” if parts of the Protein name, but not the full name, appear in the dictionary. A “full unseen” appears if no parts of the protein name are listed in the dictionary.

To get insight in this, a second round of evaluation was performed on the YAPEX Corpus. In the YAPEX corpus, a total of 353 protein names were “unseen”. From those 353 protein names, 44 were classified as “false” and 309 were classified as “correct”. From the 44 unseen examples that were classified as false, 36 names are true negatives (e.g. “chaperonins”) and 8 are false negatives (e.g. “SWUV39H1 HMTase”, “NOSIP”).

From the 309 protein names that were classified as “correct”, 244 were “partial unseens” and 65 were “full unseens”. From those 309 protein names, 25 protein names are false positives (e.g. “signal transducers and activators of transcription” or “endothelial adhesion molecules”). From those 25 false positives, 8 were “full unseens” and 17 were “partial unseens”.

From the 44 unseen examples classified as false, only two are “partial unseen”, the other 42 are “full unseen”.

So we show that when using Boosting for the tagging of protein names, the results improve significantly. For partial protein names, the score is better than for full protein names. The classifier not only works well on names that appear in the training set (“seen” names), but also on those that do not appear in the training dictionary (“unseen” names).

10.2.8 Other Corpora

In addition to the corpora I presented before, there are also other corpora I report on, but on which CONAN was not evaluated due to time constraints. CONAN will be evaluated on these corpora in the near future.

10.2.8.1 GENIA

The GENIA corpus [15] focuses on the NER part of text mining. They present a very big corpus that is already Part-of-speech-tagged. This means that they already assigned a (NLP) word category to the single terms. GENIA is a main resource of the NLP community, the current version of their corpus is GENIA version 3.02 corpus. The training corpus was formed from a controlled search on MEDLINE using the MeSH terms ‘human’, ‘blood cells’ and ‘transcription factors’.

From this search, 2,000 abstracts were selected and hand annotated according to a small taxonomy of 48 classes based on a chemical classification. Among the classes, 36 terminal classes were used to annotate the GENIA corpus. For the shared task, the 36 classes were simplified and only the classes protein, DNA, RNA, cell line and cell type were used. For the test set, 404 abstracts were used that were annotated for the same classes of entities.

Due to time reasons, CONAN will be evaluated when the new GENIA corpus 3.03 is released.

10.2.8.2 TREC

TREC (Text Retrieval Conference) is a competition similar to BioCreative first carried out in 1989. However, TREC originally did not focus on the biomedical domain. TREC activity is organized into tracks of common interest, such as question-answering, multi-lingual IR, Web searching, and interactive retrieval. A few years ago, they added a so-called "Genomics TREC" to their program, which includes evaluation of IR, NER and IE systems.

For the Genomics TREC,

- The task scenario will be that of a user seeking to acquire new knowledge in a sub-area of biology linked with genomics information.
- The databases will be publicly available.
- The focus of the task will be on text retrieval.

Evaluation in TREC is based on the "Cranfield paradigm" that measures system success based on quantities of relevant documents retrieved, in particular the familiar metrics of recall and precision. In most TREC tracks, the two are combined into a single measure of performance, mean average precision (MAP), which measures precision after each relevant document is retrieved for a given query.

In 2005, the Genomics TREC consisted of the "Categorization Task" and the "Ad Hoc Retrieval Task". For the "Ad Hoc Retrieval Task", the topics were:

- Find articles describing standard methods or protocols for doing some sort of experiment or procedure.
- Find articles describing the role of a gene involved in a given disease.
- Find articles describing the role of a gene in a specific biological process.
- Find articles describing interactions (e.g., promote, suppress, inhibit, etc.) between two or more genes in the function of an organ or in a disease.
- Find articles describing one or more mutations of a given gene and its biological impact.

For the "Categorization Task", the task was to retrieve containing the following:

- Tumor biology

- Embryologic gene expression
- Alleles of mutant phenotypes
- GO annotation

10.3. Interesting Results and Usefulness

In this section I present interesting results found in the data which gives the reader a hint of what difficulties exist in the field of text mining and especially in the biomedical domain. Moreover, I present some results showing how useful CONAN really is for the experimentalists. The results for “Mutations” and “Keywords” were found in the first experiment, while the “Protein Names” and “Interactions” were found in the second experiment (see Section 10.1).

10.3.1 Protein Names

The most common problems with gene/protein names are listed in Section 2.4.1. I now especially concentrate on the Partial Matches and the Ambiguity of protein names and show how CONAN performs on these problems. Firstly, short protein Names are very well extracted by CONAN. Protein names like p72, omtA, HSF1 or Rap1 are all true positives.

Secondly, protein names which are formed by more than one word are also extracted very well by CONAN. Good examples are: “Heat shock protein hsp70”, “heat shock cognate protein hsc70” and “protein phosphatase 2A”. It is seen in evaluations of text mining systems that partial matches (e.g. “hsp70” instead of “Heat shock protein hsp70” or “protein phosphatase” instead of “protein phosphatase 2A”) appear very often, although they are not really correct. CONAN can extract full matches very well, which is a big asset. One specific example appears in the abstract with PMID 10956549. The protein names mentioned in this abstract are: “Y-box protein 1” and “Y-box protein 3”. A method that would extract partial matches, would give only “Y-box protein” or “protein 1” and “protein 3” as a result, which would be false positives. CONAN gives the true result “Y-box protein 1”.

CONAN is also able to extract protein names that are difficult to extract, because their notation is different from abstract to abstract. One example is the protein “TCR.CD3” which denotes the complex of TCR and CD3. Normally, a complex is denoted by the “/”-sign, but CONAN is able to detect this complex as well. A partial match in this example would be if only “TCR” would be extracted. This shows that CONAN is able to extract full protein names with very high quality.

The second problem that appears very frequently in NER is ambiguity, which means that one protein symbol refers to multiple gene products (see also Section 2.4.1). This problem is solved by NLProt. As it gives information about the organism this protein belongs to and an UniProt identifier, CONAN

can assign one single Ensembl identifier to the protein name using IPI (see Section 5.8.2). With this Ensembl identifier, the protein can be identified correctly.

10.3.2 Mutations

Some interesting results appear when the Mutation Data was analyzed. As already mentioned in Section 8.3.1, cell lines or biological molecules often have names similar to point mutations. In the analysis, we found some false positives appearing very often: R6K, I3K and H2A.

R6K is an E.coli plasmid vector very often mentioned in literature. I3K comes from the abbreviation “PI3K” which is short for phosphoinositide-3-kinase. H2A is the name of a histone protein which also is mentioned quite often in text. From the data extracted so far it also is clear that most of the mutations extracted by CONAN are true positives. Especially when applying the Data Integrator (see Section 8.3.1), almost all of the false positives get filtered out. This is a big achievement of CONAN, as I do not know any other system at the moment that can extract mutations over all MEDLINE abstracts. From the data that is present at the moment, it is visible that most mutations appear not in the beginning of a protein, as single-digit mutations are rare, while the false positive results that we see (e.g. R6K, I3K and H2A) often have a single number. This could be another filtering step in future versions of CONAN.

When analyzing the output files, I noticed a mutation that fits the regular expression, but seems strange at first: G20210A. The position 20210 normally would be the sign of a false positive, because a protein is normally not over 20,000 amino acids long. I found this mutation to appear quite often in text (PMIDs 14961156, 14977830, 15134466, 15199492, etc.). From these articles it becomes clear that G20210A is a mutation of the prothrombin protein, but not on the protein level, as one might think, but on the DNA level, hence the “G” and “A” that appear in the mutation. This specific mutation in prothrombin shows up in patients for whom it is seen as a weak risk factor for VTE (venous thromboembolism). Using this example, we see that CONAN is able to extract all mutations that appear in text, no matter if they are protein point mutations or DNA point mutations.

10.3.3 Interactions

When giving examples of extracted interaction data, it first has to be mentioned that the interaction data heavily relies on the extraction of protein names from text. Although the protein name extraction methods work very well (see Section 10.2.6.3 for Evaluation), still false positives are encountered. Most of these false positives are drug names (e.g. ambroxol, gemcitabine) or “real” English words also used as Protein Names (e.g. per, which is short for

“period gene”, but also of course the normal English word “per”). Nevertheless, it has to be said that CONAN does very well on extracting PPIs from text.

Moreover, as also mentioned before, some regular expression used for Interaction Finding are very general, while some others are very specific. With the regular expression: “(A.*interacting.*B)”, which is a very general one, we nevertheless find very good results. In the abstract with PMID 15328027, CONAN finds the interaction between “JNK” and “JIP1”. This interaction is mentioned in text: “A neuroprotective protein, JNK-interacting protein 1 (JIP1).....”.

With the more specific regular expressions, CONAN is also able to find good interactions. As also mentioned in Section 6.5, one of these specific regular expressions is “(A(\S*\s+){0,3}and(\S*\s+){0,6}\S*B(\S*\s+){0,6}\S*interact(?!ion))”. In the abstract with PMID 12730328, the following sentence: “Moreover, steroidogenic factor-1 and nuclear factor Y are shown to physically interact with each other.” is found, which matches the regular expression perfectly.

It is interesting to know which regular expressions contribute most to the interactions found in the text. I extracted the regular expressions used in the second experiment (see Section 10.1) from the output XML files and counted them. The five most used regular expressions are the following:

(A(\S*\s+){0,6}\S*activat(?!ion)(\S*\s+){0,6}\S*B))
 (A.*B association)
 (A.*is(\S*\s+){0,3}for activation of(\S*\s+){0,3}\S*B)
 (A associate\w* with B),
 (complex of (\S*\s+){0,3}proteins(\S*\s+){0,3}\S*A(\S*\s+){0,4}\S*B)

From this list it is clear that there are two very general regular expressions (119 and 106) which contribute to the interactions, but also three quite specific ones. So we can conclude that both the general and the specific regular expressions are needed to extract high-quality PPIs. It also becomes clear that the words “activation” and “association” are fundamental to these regular expressions. Using SL the specific and the general regular expressions, CONAN is able to extract PPIs with very high quality.

Interesting false positives that I come across quite often are interactions with “glutathione-S-transferase”. This protein is, of course, a normal protein, but it is also very often used in the so called “glutathione-S-transferase pull-down assay”. This is a method to detect interactions experimentally. Hence, abstracts quoting this method also often have interactions mentioned. An example from the abstract with PMID 12388720 : “To better understand the mechanism of chromosomal tethering, we performed glutathione S-transferase (GST) affinity and yeast two-hybrid assays to identify LANA-interacting proteins with known chromosomal association.” Here, CONAN finds an interaction between LANA

and glutathione S-transferase, which is a false positive to the human reader, but completely understandable from the automated extraction viewpoint.

Some interactions were extracted that do not form part of the golden standard for the human gene interaction network (see Section 11.4), but are excluded as true negatives. These interactions appeared very often (30-90 times) in the extracted interactions. By analyzing these interactions, we saw that these interactions are indeed true positives and the golden standard and the true negative lists used in the construction of the gene interaction network should be updated. These interactions were: PSEN1-PSEN2, IL4-JAK1, IL2-JAK1, INSL3-IL2 and INSL3-IL4. This result shows that CONAN produces high-quality results that offer high reliability to the user.

10.3.4 Keywords

When looking at the results from the keyword extraction method (BLAST), CONAN displays some very interesting results.

The method is very good in extracting diseases and tissues, like “amyotrophic lateral sclerosis” or “skeletal muscle tissue”, which is very useful for experimentalists. Note that for the extraction of keywords, the extraction of multi-word terms is very important. This is done almost flawlessly by CONAN.

The extraction of Experimental Techniques is really valuable for the user. Terms like “Co-immunoprecipitation” or “mass spectrometry” are frequently found in text. This gives the biologists the chance to search for those specific terms. This is also one of the strong points of CONAN. Lots of different information can be displayed and the information is really helpful for experimentalists. In the keyword search, the results of the query can be narrowed down to the really important and interesting results, which is a big asset for the experimentalist.

Moreover, Pharmacologic Substances like “aluminum hydroxychloride” are extracted accurately by CONAN. Terms like “Anticoagulant Drugs” are found very often.

This information combined gives every experimentalist the chance to find the articles he is interested in. The combination of all these keywords gives the experimentalist a good overview of what is mentioned in the abstract and helps him a great deal in deciding which articles to read.

References

- [1] <http://www.pdg.cnb.uam.es/BioLINK/BioCreative.eval.html>.
- [2] <http://predictioncenter.org/>.
- [3] G.D. Bader, D. Betel, and C.W. Hogue. BIND: the Biomolecular Interaction Network Database. *Nucleic Acids Res.*, 31(1):248–250, 2003.
- [4] E.B. Camon, D.G. Barrell, E.C. Dimmer, V. Lee, M. Magrane, J. Maslen, D. Binns, and R. Apweiler. An evaluation of GO annotation retrieval for BioCreAtIvE and GOA. *BMC Bioinformatics.*, 6:S17, 2005.
- [5] K.B. Cohen, L. Fox, P.V. Ogren, and L. Hunter. Corpus Design for Biomedical Natural Language Processing. *Proc of ACL-ISMB 2005 Workshop*, pages 38–45, 2005.
- [6] N. Collier, H.S. Park, N. Ogata, Y. Tateishi, C. Nobata, T. Ohta, T. Sekimizu, and H. Imai. The genia project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proceedings of the ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 271–272, 1999.
- [7] S. Dingare, M. Nissim, J. Finkel, C. Manning, and C. Groer. A System for Identifying Named Entities in Biomedical Text: How Results From Two Evaluations Reflect on Both the System and Evaluations. *Comp Funct Genom*, 6(1), 2005.
- [8] K. Franzen, G. Eriksson, F. Olsson, L. Asker, P. Liden, and J. Coster. Protein names and how to find them. *Int J Med Inf*, 67, 2002.
- [9] B. Hachey, B. Alex, and M. Becker. Investigating the Effects of Selective Sampling on the Annotation Task. In *Proceedings of the 9th Conference on Computational Natural Language Learning.*, 2005.
- [10] M. Krallinger, R. Malik, and A. Valencia. The prodisen corpus: exploring the construction and applications of a protein description corpus. 2006. Submitted to BioNLP06.
- [11] H.L. Kundel and M. Polansky. Measurement of observer agreement. *Radiology*, 228(2):303–308, 2003.
- [12] J.R. Landis and G.G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

- [13] S. Mika and B. Rost. Protein names precisely peeled off free text. *Bioinformatics.*, 20(Suppl 1):I241–I247, 2004.
- [14] C. Nedellec. Learning language in logic - genic interaction extraction challenge. In *Learning Language in Logic Workshop (LLL'05) at ICML 2005*, 2005.
- [15] T. Ohta, Y. Tateisi, M. Hideki, and J. Tsujii. GENIA Corpus: an Annotated Research Abstract Corpus in Molecular Biology Domain. *Proceedings of the Human Language Technology Conference.*, 2002.
- [16] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
- [17] I. Xenarios, L. Salwinski, X.J. Duan, P. Higney, S.M. Kim, and D. Eisenberg. DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res.*, 30(1):303–305, 2002.
- [18] A. Yeh, A. Morgan, M. Colosimo, and L. Hirschman. BioCreAtIvE Task 1A: gene mention finding evaluation. *BMC Bioinformatics*, 6, 2005.

Chapter 11

APPLICATIONS

In this chapter I introduce several applications based on CONAN. Firstly I introduce the CONAN querying systems, both the CONAN Webserver and the CONAN command line tool. Secondly, I present the human gene interaction networks approach by Lude Franke, where CONAN forms an integral part of the system.

11.1. Command Line

The main way to extract data from the Output XML file (see Chapter 8) is a command line tool programmed in Python. The script is called “findme.py”.

The structure of this script is:

Argument 1: search term, Argument 2: search database

if Search term is one word only **then**

 Leave the term as it is

else if Search term is a multi-word term **then**

 Split the term into single words

end if

Append all terms to WORDS_TO_FIND

Load Index of CONAN Output file

Search for WORDS_TO_FIND in index, use search database

if There is a result **then**

 Result contains XML nodes

 Load CONAN Output File for “real” text

 Extract “real” text from specified XML nodes

 Display Results

end if

For simplicity reasons, there are only two command line parameters for a CONAN command line search. Firstly, the search term has to be entered. When the search term is a multi-word term (e.g. “cytochrome P450”), the term has to be quoted. The second parameter is the category (e.g. Protein, Mutation) the user wants to search in. So for instance, if the user wants to search for information about “p53” as a protein, the parameters would be “p53 Protein”. If the user wants to search for p53 included in any PPI, the parameters would be “p53 Interaction”.

The script consists of several important parts. Firstly, the input values have to be determined and processed accordingly. So if the search term includes a multi-word term, this term has to be first split into the different single-word terms (e.g. “cytochrome P450” will be split into “cytochrome” and “P450”). Multi-word terms are not only split by spaces, but also by other non-alphanumerical characters like “_”, “-”, “(”, “)”, “/” and “+”.

This has to be done because of the search method that is included in the script and which forms the second important part of the command-line tool.

Because the CONAN output XML files are quite big (14-45 MB per file), there was a need to index the XML files with an XML indexer. Normally, indexing is done in a database to allow quick finding of specific rows, usually via a balanced tree. Searching in those index files is much quicker than in the original database, although the index files might be bigger than the original files. The same thing can be done for an XML database like the CONAN output structure. Indexing is done by the package “Gnosis XML Indexer” [1]. The Gnosis XML indexer stores the XML path of each occurring word and the number of occurrences of this word in a database. Firstly, a splitter has to split all the XML entries (i.e. plain text) into single words. It deletes common stopwords. Then it stores the words, XML paths to the words and the occurrences of this word in a database, one database for each begin-letter of the word. The Gnosis XML Indexer source code had to be changed slightly, because in the original version numbers and short words were not indexed. In the case of CONAN, numbers (e.g. PMIDs) and short words (e.g. protein symbols) are an integral part. After the indexing is done, all the indexes are put in one place and the original files are not modified any further to prevent incompatibilities. Note that this indexing is the reason why the input fields of the XML querying tool are split into single words. When going back to our example, “cytochrome P450” would be split by the indexing method into “cytochrome”, belonging to the index file for words beginning with “C” and “P450” would be stored in the file for the letter “P”.

The Gnosis Indexer also includes a method to query files with the words the user wants to find. After splitting the input fields into single words, these single words are stored into an array, called “WORDS_TO_FIND”. The Gnosis XML Indexer searches for instances of all words included in this array. It

automatically computes the overlap of the sets found for each single word and gives back this overlap as the result.

So, for every of those words, the indexes are searched and the XML paths where this specific word(s) are found, are returned. In our example, Gnosis tells us that the words we are looking for appear in the abstract with number 1090, and in the “term_text” tag (which is a child of the “abstract_body” tag) with number 15.

```
/abstract[1090]/abstract_body/term_text[15]
```

From these XML paths, the original XML entries from the original XML file can be retrieved. The retrieval of the original words cannot be done with the Gnosis XML indexer, as it only gives back the number of the XML node. The Python built-in method “libxml2” is used for these purposes. “libxml2” includes full XPath compatibility. Via XPath statements that include the XML Paths found by the Gnosis Indexer, we can look for the original content of this specific abstract. These XPath statements look like this:

```
/abstract[1090]/method_overview/method/.  
/abstract[1090]/abstract_header/PMID
```

Because we already know from the Gnosis XML Indexer results which abstract(s) contain the search term (in this case, the abstract with number 1090), we can easily query the native XML database by giving it precise XPath directions to the abstract(s) wanted. These XPath statements (see above) will return the method used in this abstract (e.g. BLAST, AbGene etc.) and the PMID of this abstract. The output of the command line tool for a simple protein name search is the following.

```
>./findme.py p53 Protein  
15994823 PreBIND 16  
15916963 PreBIND 15  
15994771 AbGene 14  
15994771 NLProt 13  
15640620 AbGene 13  
15884108 PreBIND 12  
15640620 NLProt 12  
15994774 NLProt 9
```

```
.  
.
```

Here we see the PMID where the search term is found, the method by which it was found and the number of occurrences. With this architecture, we can find the query words in all indexes in a fast and reliable way. The XPath queries for the CONAN webserver are explained in Section 11.3. The webserver, explained and shown in the next section, also uses this command line script.

11.2. Web Server

In order to deliver the data produced as completely and concisely as possible to experimentalists, a web-service was created that provides the user with lots of possibilities to query the data and get the results they want out of the data collection. This is also done to address the problem of visualizing the data in such a way that experimentalists can use the system without any problems. In Figures 11.1 and 11.2, the overall layout of this website can be seen.

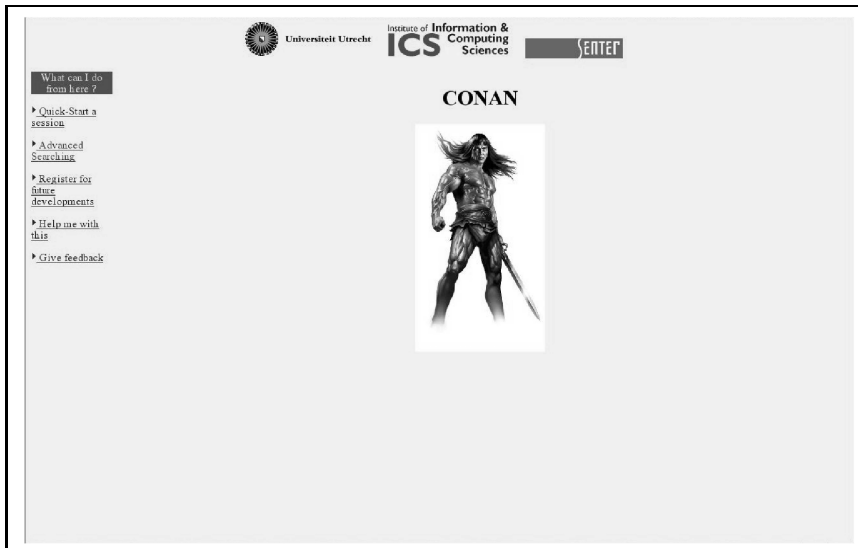





Figure 11.1. CONAN Website Layout

The main part of the website, of course, consists of HTML code. Furthermore, a Perl script was constructed that makes use of the Python command line tool. This relation can be seen in the Figure 11.3.

11.2.1 System Architecture

As the data presented is stored in XML (see also Chapter 8), the main concern of the web-server is to extract the data from XML and present it in HTML as quickly and simply as possible.

The main script of the web-server is a Perl Script programmed by me and is called "doitmore.pl". The main reason why I chose for a Perl script in place


 Universiteit Utrecht
 
 Institute of Information & Computing Sciences
 

Search: Protein Name

What can be entered ?

- a Protein/Gene name - all information regarding this protein will be retrieved
- a keyword - all information regarding this keyword will be retrieved
- a PMID - all information in this article will be retrieved
- an interaction partner - give a protein name and all interaction partners will be retrieved
- a mutation - give a protein name and all mutations for this specific protein will be retrieved
- Enter a GO-number - a list of proteins will be retrieved. note: only go-number is possible at the moment, not go-term
- Enter an ENSEMBL-ID - all articles containing a protein with this ID will be retrieved
- Enter an UniProt Synonym - all articles containing a protein with this synonym will be retrieved
- Enter a MESH-Term - all articles containing this MESH term will be retrieved

IMPORTANT: for search of keywords which are longer than 1 word -> please use quotation marks (") around the words, e.g. "carcinoembryonic antigen".

Figure 11.2. CONAN Input Form

of a Python script is that Perl has many built-in functionalities for web-server programming.

In principle, it takes the data filled in by the user at the CONAN HTML website. The search term and the goal-database are passed as parameters to the Python script `findme.py`. This script performs the search and the results are handed back to the Perl script.

The Perl script then processes this information and displays the results. As the display is different from (search) database to database, we had to produce many different methods for the display of all included extraction methods (e.g. BLAST, NLPProt, etc.).

11.3. Queries

For each query that is possible with the CONAN webserver or command-line tool, I give information how it is processed and also give an example on how the HTML output looks like. The queries that can be sent to the web-server or be made via the command-line interface, include the following functionalities.

11.3.1 PMID

The simplest search using the CONAN webserver is the search for a certain PMID. The input is simply the number of the MEDLINE abstract the user wants to see. This input is given to the Perl Script.

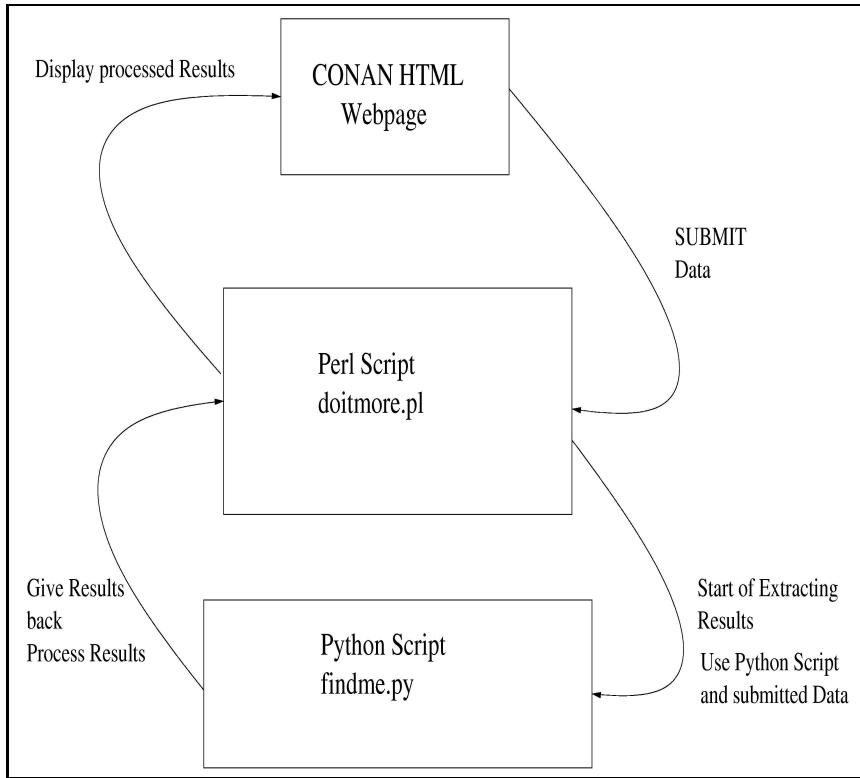


Figure 11.3. CONAN Webservice Communication

```
10532697
```

In the Perl Script, firstly the information is passed on to the Python Script.

```
findme.py 10532697 PMID
```

In the Python script, the information is processed in the following way:

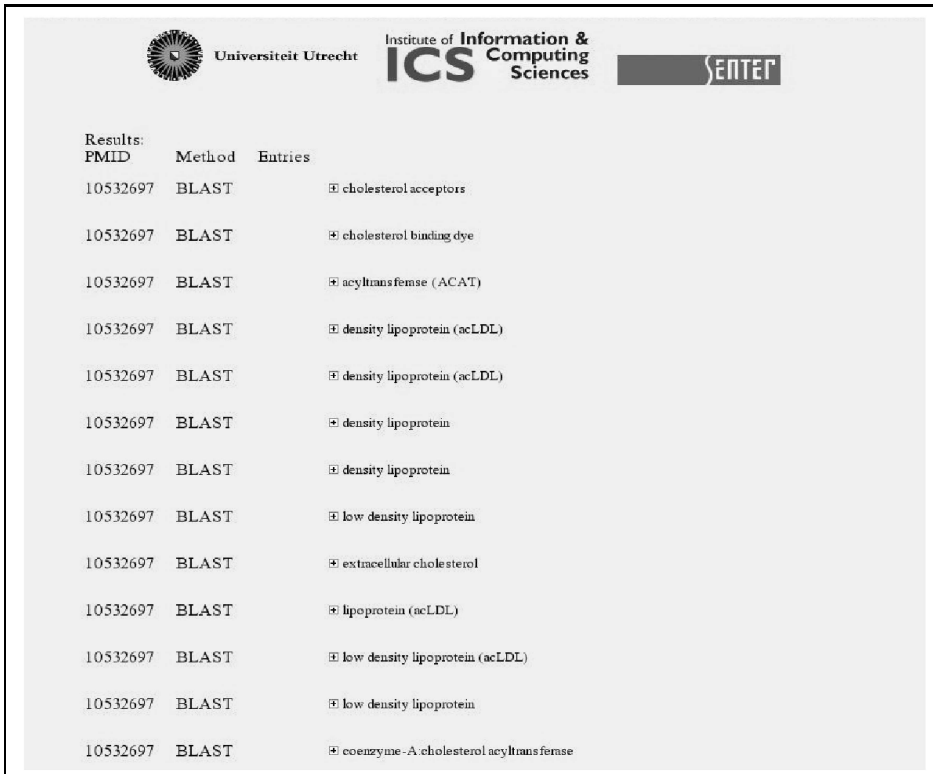
```
method="/collection/abstract[number]/method_overview/method/."
result="/collection/abstract[number]/abstract_body/."
method=ctxt.xpathEval(method)
res1 = ctxt.xpathEval(result)
```

This means that for each different entry with the same PMID (read: for every different extraction method that was used on one abstract), the name of the

method (saved in the variable `method`) and the corresponding entries (saved in the variable `res1`) are returned.

The information that is passed back from the Python script to the Perl script is processed. Because a search for a certain PMID will yield all information contained in the abstract with the corresponding PMID, the processing step first has to distinguish between the methods. This is done by the information stored in the “method” variable.

The information stored in the “results” variable for each method is then processed accordingly. The result that is given back to the user can be seen in Figure 11.4.



Results: PMID	Method	Entries
10532697	BLAST	+ cholesterol acceptors
10532697	BLAST	+ cholesterol binding dye
10532697	BLAST	+ acyltransferase (ACAT)
10532697	BLAST	+ density lipoprotein (acLDL)
10532697	BLAST	+ density lipoprotein (acLDL)
10532697	BLAST	+ density lipoprotein
10532697	BLAST	+ density lipoprotein
10532697	BLAST	+ low density lipoprotein
10532697	BLAST	+ extracellular cholesterol
10532697	BLAST	+ lipoprotein (acLDL)
10532697	BLAST	+ low density lipoprotein (acLDL)
10532697	BLAST	+ low density lipoprotein
10532697	BLAST	+ coenzyme-A cholesterol acyltransferase

Figure 11.4. CONAN Result: PMID Search

Please note that for the sake of presentation, the attributes have been collapsed using the java-script. By clicking on the “+” sign, all the information about a specific term will be displayed. This tree-like structure was chosen because, as you will see below, some terms have a lot of attributes, which need a lot of space. Displaying all the results at once would make the system unusable.

11.3.2 Keyword

Another search method implemented in the CONAN webserver is the search for keywords. The keywords search consists of both searching for keywords found by the BLAST method and searching for MeSH terms. The input for the Perl script looks like this:

```
plasma membrane
```

The Perl script passes this information on to the Python script:

```
findme.py "plasma membrane" Keyword
```

The Python script transforms this to:

```
method="/collection/abstract[number]/method_overview/method/."
result="/collection/abstract[number]/abstract_body/."
method=ctxt.xpathEval(method)
res1 = ctxt.xpathEval(result)
```

It becomes clear that for both methods mentioned so far, the XPath statements are the same. This is also one of the strong points of the script. For both the PMID and the Keyword method, only the method itself and the information stored in the "abstract_body" tag are needed.

In contrast to the PMID finding method, a list of PMIDs is compiled that contains the search-term. The list is made semi-non-redundant by a simple method in the Python Script. It is "semi-non-redundant" because the user can see how often a specific search term appears as found by each method. In Figure 11.5, the list of PMIDs is displayed as well of the number of occurrences of each query term in the results. For instance, if "plasma membrane" is found 4 times in an abstract by the BLAST method and 1 time in the same abstract by MeSH, both results are displayed. The user then can simply click on one PMID and gets the information about the PMID, as explained above.

In this method, no transformation of the data is needed. The Python script returns a list of PMIDs that contains the wanted search-term and the Perl script converts this list to HTML.

11.3.3 Protein

For a search for a protein name, the approach is similar to that of the keyword search.

```
p53
```

The Perl script passes this information on to the Python script:

```
findme.py p53 Protein
```

The Python script transforms this to:

```
method="/collection/abstract[number]/method_overview/method/."
result="/collection/abstract[number]/abstract_body/."
method=ctxt.xpathEval(method)
res1 = ctxt.xpathEval(result)
```

As one can see, the XPath statements are again the same as for the other two methods. Identical to the Keyword Search method, a list of PMIDs containing the wanted Protein is displayed. Again, the list is made semi-non-redundant and passed on to the Perl Script.

The Perl Script again transforms the list to HTML and displays the list. Clicking on one of the PMIDs will yield the information about the article. As the list of PMIDs is the result also for the Mutation and Keyword Search, I present a screenshot of such a clickable list in Figure 11.5.

11.3.4 Mutation

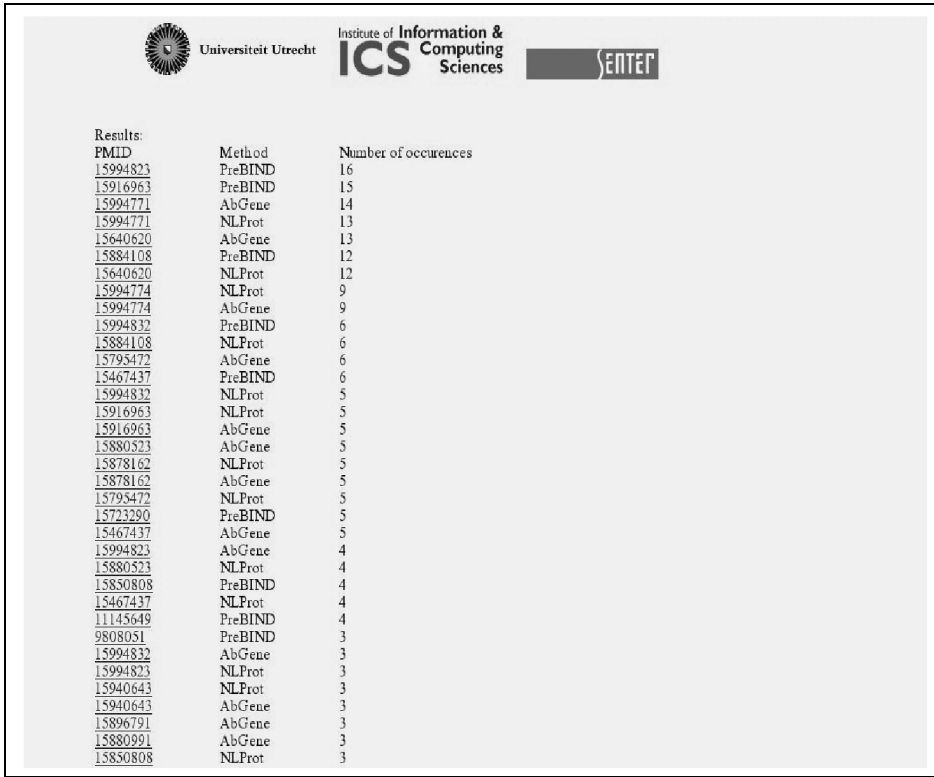
The search for a mutation in a specific protein differs quite a lot from the methods mentioned above. When searching for a mutation, the user has to give the name of the protein that should show a mutation. So the input looks like this:

```
nNOS
```

The Perl script passes this information on to the Python script:

```
findme.py nNOS Mutation
```

The script will first start to look for occurrences of those protein names in the abstracts, similar to the Protein Search method. When the protein name is found in certain abstracts, the method starts to look for mutation indicators in the same abstracts. This is done simply via the PMID.



PMID	Method	Number of occurrences
15994823	PreBIND	16
15916963	PreBIND	15
15994771	AbGene	14
15994771	NLProt	13
15640620	AbGene	13
15884108	PreBIND	12
15640620	NLProt	12
15994774	NLProt	9
15994774	AbGene	9
15994832	PreBIND	6
15884108	NLProt	6
15795472	AbGene	6
15467437	PreBIND	6
15994832	NLProt	5
15916963	NLProt	5
15916963	AbGene	5
15880523	AbGene	5
15878162	NLProt	5
15878162	AbGene	5
15795472	NLProt	5
15723290	PreBIND	5
15467437	AbGene	5
15994823	AbGene	4
15880523	NLProt	4
15850808	PreBIND	4
15467437	NLProt	4
11145649	PreBIND	4
9808051	PreBIND	3
15994832	AbGene	3
15994823	NLProt	3
15940643	NLProt	3
15940643	AbGene	3
15896791	AbGene	3
15880991	AbGene	3
15850808	NLProt	3

Figure 11.5. PMID List

If an abstract contains the protein name as well as a mutation, an additional “mutational term” (see Section 8.3.1) is searched for. This ensures that this is a real mutation for this specific protein.

As already explained in the Chapter 8, if a “mutational term” is found and this term is within a certain distance to the protein name and/or the mutation, it is considered as a true positive. We assume, that an average sentence in an abstract is about 10-20 words long. If the distance is not longer than 30 words, it is considered as a true positive.

The XPath statements look like this:

```
PMID="/collection/abstract[number]/abstract_header/PMID/."
PMID=ctxt.xpathEval(PMID)
result="/collection/abstract/abstract_header/PMID[contains(.,PMID)]
/../../method_overview/method[contains(.,'MuText')]/../../abstract_body/*"
result=ctxt.xpathEval(result)
result2="/collection/abstract/abstract_header/PMID[contains(.,PMID)]
/../../method_overview/method[contains(.,'BLAST')]/../../abstract_body/*"
```

In the XPath expression we can see that this query actually consists of three different queries, first for the protein name itself, then for the MuText mutation and then for the “mutational term” found by the BLAST method.

Again, the HTML output contains a list of PMID where the protein, a mutation and a mutational term is found.

11.3.5 Interaction

What might seem strange for a user of the CONAN webserver is that there are two different ways to query for PPIs. This was actually done due to a suggestion by the biologists who tested the system.

When searching for interactions, normally only very well-known and well-documented interactions appear as result because some interactions are simply overrepresented in MEDLINE. Therefore we decided to construct two different methods for interaction searching, namely “Interaction / all” and “Interaction / PMID”.

11.3.5.1 Interaction / all

The “Interaction / all” method is the conventional method. It takes, again, a protein name as input.

```
p53
```

The Perl Script starts the Python Script by:


```
./findme.py p53 Interactionall
```

In this conventional method, all articles are searched through and a list with the PPI and the number of occurrences of this PPI are displayed (see Figure 11.6).

```
method="/collection/abstract[number]/method_overview/method/."
method=ctxt.xpathEval(method)
PMID="/collection/abstract[number]/abstract_header/PMID/."
PMID= ctxt.xpathEval(PMID)
cmd="/collection/abstract/abstract_header/PMID[contains(.,PMID)]/../../method_overview/method[contains(.,'PreBIND')]/../../abstract_body/*"
```

The XPath statements tell us that, again, first the Method and the PMID that contains the wanted protein are looked for. After that, we look at the PMID where the protein is mentioned and then extract the information stored in the “abstract_body” of the PreBIND method.

The display of this information is also quite different. Instead of a list of PMIDs, the user is presented a list of all found PPIs. Furthermore, the number



Results:

Show the whole network

Protein 1	Protein 2	Interaction Type	No. of occurrences	More Interactions for TNFalpha	More interactions for p38	Add this interaction to network
TNFalpha	p38	Activation	5	More Interactions for TNFalpha	More interactions for p38	Add this interaction to network
TNFalpha	MAPK	Activation	3	More Interactions for TNFalpha	More interactions for MAPK	Add this interaction to network
TNFalpha	Cdc42	Activation	3	More Interactions for TNFalpha	More interactions for Cdc42	Add this interaction to network
p53	TNFalpha	Activation	2	More Interactions for p53	More interactions for TNFalpha	Add this interaction to network
TNFalpha	p53	Activation	2	More Interactions for TNFalpha	More interactions for p53	Add this interaction to network
p53	TNFalpha	Mutation	1	More Interactions for p53	More interactions for TNFalpha	Add this interaction to network
TNFalpha	mitogen-activated protein kinase p38	Activation	1	More Interactions for TNFalpha	More interactions for mitogen-activated protein kinase p38	Add this interaction to network

Figure 11.6. Interaction / All

of occurrences of this specific interactions is reported. Moreover, the sort of interaction (Activation, Inhibition, etc.) is reported. An additional function is the displaying of Interaction Graphs, which will be explained in detail in Section 11.3.5.3.

Although this sort of interaction finding can be handy for some biologists, we propose another strategy for looking for interactions that are not mentioned very often in text.

11.3.5.2 Interaction / PMID

For this second method, the Input is exactly the same, because the same information is extracted as in the previous set of queries. The calling of the Python script is only a little different:

```
./findme.py p53 InteractionPMID
```


Instead of displaying the number of occurrences, however, we display every single interaction for every single abstract. The main point of this method is to display the interactions that are found in the newest (by date) abstracts first, because they might be of bigger importance for the researcher. Older abstracts often contain interactions that are well-known to experimentalists.

The downside of this method is that many interactions could be displayed more than once (redundancy) and that the list of interactions is quite long as can be seen in Figure 11.7. Here, the interaction “TNFalpha” with “p53” appears very often. To compensate for this downside, we offer additional functionalities for this search method. For each interaction, we display the PPI itself and the sort of interaction as well as the PMID where this interaction was found. For each single interaction, also a score of this interaction is displayed (also see Chapter 8). This score is translated into a color code. Hence, the “greener” an interaction appears, the more clear the interaction is. In the black-and-white version of Figure 11.7, a lighter colour represents a “greener” interaction. The possibility to display the Interaction Graphs is also given in this method.

Protein 1	Protein 2	Interaction Type	Method	Score	PMID found			
TNFalpha	p38	Activation	PreBIND	0.84	15561768	More Interactions for TNFalpha	More interactions for p38	Add this interaction to network
TNFalpha	p38	Activation	PreBIND	0.74	15561768	More Interactions for TNFalpha	More interactions for p38	Add this interaction to network
p53	TNFalpha	Activation	PreBIND	0.91	15561768	More Interactions for p53	More interactions for TNFalpha	Add this interaction to network
p53	TNFalpha	Activation	PreBIND	0.84	15561768	More Interactions for p53	More interactions for TNFalpha	Add this interaction to network
p53	TNFalpha	Activation	PreBIND	0.81	15561768	More Interactions for p53	More interactions for TNFalpha	Add this interaction to network
TNFalpha	p53	Activation	PreBIND	0.91	15561768	More Interactions for TNFalpha	More interactions for p53	Add this interaction to network
TNFalpha	p53	Activation	PreBIND	0.81	15561768	More Interactions for TNFalpha	More interactions for p53	Add this interaction to network
						More		Add this

Figure 11.7. Interaction / PMID

11.3.5.3 Interaction Graphs

Displaying interaction graphs is important for biologists. As the two main search functions only display binary (one-to-one) relationships (e.g. PEBP2 - mMCP-6 in Figure 11.7), the experimentalists might want to display whole interactions paths. This is why the CONAN webserver gives this possibility.

As you can see in the screenshots (Figures 11.6 and 11.7), we offer the possibility to add a certain binary interaction to the network. When the user is done with adding, he can display the whole network at once. All the selections the user makes are stored for the whole session, meaning that the user can perform multiple searches without losing the information he added before.

This method was implemented using the “graphviz” package [2] and especially the “dotty” library. A dotty file looks like this:

```
digraph "" {
  "PEBP2" -> "mMCP-6"
  "MITF" -> "mMCP-6"
  "MAZR" -> "mMCP-6"
}
```

In this case, the PPIs “PEBP2 with mMCP-6”, “MITF with mMCP-6” and “MAZR with mMCP-6” were added to the network.

So each time the user adds information to the network, a new line is added to this “.dot” file. When the user wants to display the whole network, a new Python Script, called “getgraph.py” is started.

With the program “dot” (also contained in the graphviz package), such a “.dot” file can be easily converted into a “PNG” graphics file. This graphic is displayed in the user’s web-browser. This Figure (see Figure 11.8) can be saved or printed out for later purposes.

11.3.6 Ensembl, UniProt, Gene Ontology

As for these three methods, the way of querying in displaying the data is exactly the same, I just give an overview. Identical to the search for a protein name, the system is queried for an Ensembl number (e.g. ENSG00000149591), an UniProt term (e.g. EPO_HUMAN) or a GO number(e.g. 0005125). The XPath statements also are identical:

```
method="/collection/abstract[number]/method_overview/method/."
result="/collection/abstract[number]/abstract_body/."
method=ctxt.xpathEval(method)
res1 = ctxt.xpathEval(result)
```

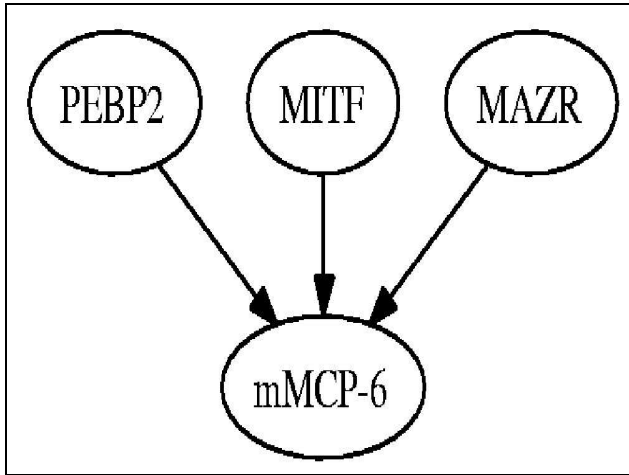


Figure 11.8. Gene Interaction Graph

Again, lists of PMIDs that contained the wanted search term, are displayed.

This proved to be the most important search for biologists. Biologists often annotate their data with Ensembl numbers or UniProt terms. Being able to search for articles that contain those terms or numbers is a very important step for biologists.

For instance, if a biologist is interested in a certain protein and this protein is assigned a GO number (e.g. 0005125 = cytokine activity), he can look for abstracts describing proteins which are assigned the same number, thus being involved in the same process or pathway. Doing this search without the help of a text mining system is almost impossible. The same is true for the UniProt and Ensembl identifiers. Searching with those identifiers with systems like Entrez would yield no results at all.

So we see that by using rather simple queries, CONAN offers lots of functionality. We also see that the visualizing of the data can be done so that every experimentalist can find the information he wants really easy. The pictures of interaction networks produced by the server can be easily included in presentations or publications.

11.4. Gene Interaction Networks

Another application where the data produced by CONAN is used, is the construction of Gene Interaction Networks (GINs).

Over the last years, the data about proteins and protein-protein interactions (PPIs) has improved significantly. Especially the information about PPIs are crucial for understanding fundamental processes taking place in a cell. Therefore, high-quality PPI networks provide new insights for biologists into the function of proteins and the function of a cell itself. More specifically, PPI

networks can be used to discover new candidate genes for diseases or new genes acting in signalling mechanisms.

An experimental technique for gathering those PPIs is the yeast two-hybrid screen (Y2H), but also other methods are used to experimentally collect data about PPIs, like Chromatography, Lethal Assays and Genome Context Methods. Large-scale interaction experiments with human proteins are still to be done. In Systems Biology and Bioinformatics, however, the collection of data from literature and other publicly available databases has provided highly accurate PPIs, but the number of interactions extracted by those methods is still quite small. With those two different approaches, we attempt to integrate experimental and theoretical data to construct very detailed PPI networks that are still comprehensive.

PPI networks have been constructed for different organisms, like Yeast [9], Drosophila [10] or Bacteria like Plasmodium falciparum [8]. First results of PPI networks in human [12] have also been presented, mostly as hypothesis-driven studies or from orthologous interactions.

11.4.1 Constructing the Network

The method where results from CONAN are included in a future release is extensively described in [4]. Firstly, a “gold standard” has to be derived from validated direct gene-gene and/or protein-protein relationships from several resources. “Gold-standard” means that only real true positives are allowed in this list, so only high quality interactions are part of the gold standard. PPIs were derived from BIND [3], HPRD [11], Reactome [6] and KEGG [7].

Also a negative set had to be constructed. This is not as straight-forward as it may seem, because it is impossible to be certain that two proteins do not interact. However, by using GO, it is possible to construct a list of gene-pairs that are highly unlikely to interact. The GO Component annotation was used to yield groups of gene pairs which have exclusive cellular component annotation. If two genes do not appear in the same cell component, they are unlikely to interact. In order to be able to process this data, the GO, microarray and PPI data has first to be pre-processed and binned into different bins. Binning is converting continuous data to discrete data by replacing a value from a continuous range with a bin identifier. Several measures were defined to categorize the data into different bins. Especially for the GO data, which is not continuous, measures of relatedness using the maximal hierarchical depth in the GO tree had to be defined. For compatibility reasons, each gene pair was assigned its Ensembl code.

A Bayesian classifier was constructed to integrate the various bins of data. A Bayesian classifier is a simple probabilistic classifier based on probability models. An introduction to Bayesian networks can be found in [5].

This classifier has a pre-defined network structure, which was derived from the gold standard set (training phase). Four different networks were constructed, generated on the basis of a Bayesian framework, with interactions from the aforementioned sources: the GO network, where the data is based on the GO data, the MA+PPI network, which includes the interactions derived from microarray data and predicted PPIs, the GO+MA+PPI network, which includes all of the above and which was complemented with all known true interactions to form the final network (GO+MA+PPI+TP network). The bayesian classifier was evaluated on these networks.

First results show, that the network achieves considerable accuracy (see Table 11.1). Please note that the AUC (Area under the Curve) is 50% when the classifier is uninformative (random). From this table, it becomes clear that the GO network performs well and provides good evidence for PPIs . When the microarray and the predicted PPI data are added, the performance is even better. The microarray and PPI network does not perform very well, probably because the microarray dataset does not discriminate between co-expression and co-regulation.

Table 11.1. Accuracy of the Gene Interaction Networks

Network	Area under the Curve (AUC)
GO	88%
MA+PPI	68%
GO+MA+PPI	89%

Moreover, when validating the network on a list of new found interactions (thus not in the data sources), it was found that the gene network is capable of inferring unknown interactions. The network can be used for selecting candidate genes of diseases. This application can be studied in the original publication [4]. The new found interactions show that these kind of gene interaction networks are true Knowledge Discovery (KD) tools.

11.4.1.1 www.genenetwork.nl

A web-server was constructed to let the user query the gene interaction network. This service also includes visualization of the results, as can be seen in Figure 11.9. In this figure, the interactions of BAX are displayed in a list and in an interaction graph. Please note that due to the colour scheme of the website, the black and white screenshot had to be resized and the brightness had to be reduced.

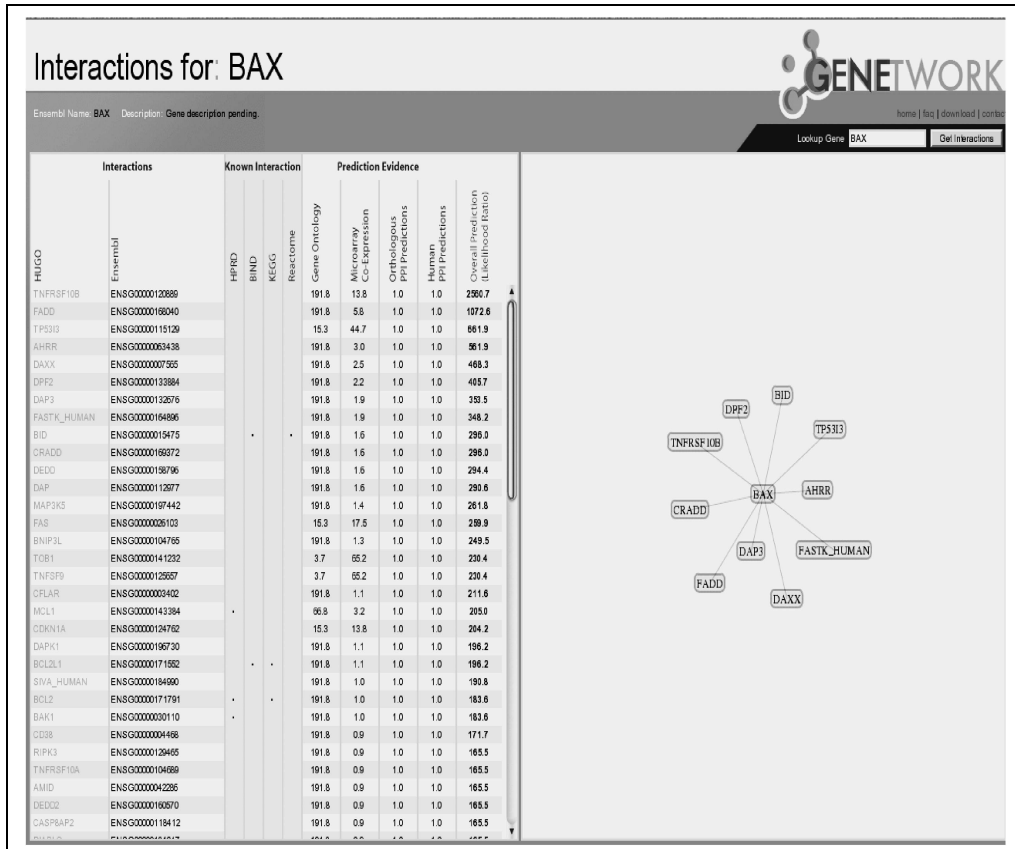


Figure 11.9. www.genenetwork.nl

11.4.2 How Literature Can Help

MEDLINE data, so “real” extracted interaction data, has been chosen to complement the data used already. It is another data source that improves the reliability and stability of the network. For this reason, PPIs extracted by CONAN are used to build an additional data source. Two different types of data are used in this experiment, cocitation Data and “real” extracted data.

11.4.2.1 Cocitation vs. Extraction

The two main ways to extract PPI data out of abstracts are: Cocitation data and “real” extracted interaction data.

Cocitation data solely relies on the fact that if two proteins occur together in one or (preferably) more abstracts, they probably share a common function or are integrated in the same process.

Real extracted PPI data has some disadvantages when compared to cocitation data, but the advantages are much higher here. The disadvantage lies in the fact that the recall is much higher in cocitation data, but the precision is much lower. This is due to the fact that when two proteins are mentioned together in an abstract, they might not interact with each other. This would be considered as a false positive, which causes the precision value to decrease.

When two proteins are simply mentioned together in one or more abstracts, they might be involved in the same process or perform the same action, but they might not have a direct interaction with each other.

```
ENSG00000141510,ENSG00000087088,ENSG00000153201,ENSG00000197424 10741712
```

In this example we see that proteins “p53” (ENSG00000141510) and “Bax” (ENSG00000087088) are co-cited together. In this article, however, there is no interaction reported of those two proteins.

“Real” extracted PPIs have very high precision, so the interactions extracted have a very high reliability. On the other hand, however, some interactions might get missed.

```
ENSG00000168394 ENSG00000080469 100 10586064
```

In this example, the interaction found between “TAP1” and “TAP2” is a real interaction, found in many more articles.

11.4.2.2 (Preliminary) Results

In this first preliminary analysis, we focused on the “real” extracted interaction data. We assessed a total number of 288,026 PPIs, from which 173,101 were interactions of two human gene products. 47,361 of those were unique interactions.

When integrating the interactions found by CONAN, we see an improvement of the Area under the Curve (AUC). The original AUC from GO+MA+PPI is 89.75%, when adding the PubMed data, this improves to 90.04%. This means that including extracted interactions from MEDLINE will definitely improve the performance of the network. This is also reflected by the Likelihood Ratios that are obtained for CONAN. The binning procedure works as follows: For every bin per cross validation there should be at least one TP and one TN. Because 10-fold cross validation is performed, each time the training consists of 9/10 of the data and the subsequent testing on the latter 1/10 of the data. We wanted to be sure that there were sufficient TPs and TNs per bin in the training set. When using the above strategy there are at least 10 TPs and 10 TNs per bin when training, resulting in an accurate likelihood ratio that does not have a big variance. This approach works out well in the

cross validation: the AUCs per cross validation are highly comparable, thus indicating there are no major overfitting issues. The resulting likelihood ratios for the different bins of MEDLINE data can be seen in Table 11.2.

Table 11.2. Likelihood Ratios for PubMed Bins

Bin	Likelihood Scores
0	0.968
1	17.607
2	27.813
3	40.379
4	70.615
5	120.524
6	119.221
7	155.723

The reason why the improvement is not that big is that the overlap between the extracted data by CONAN and the gold standard is quite small. This is due to the fact that only 4 years of MEDLINE (2003,2004,2005 and 2006) have been processed so far. Interactions that are present in the gold standard might have been published before these years. When extracting more interactions from previous years, the overlap and thus also the AUC will improve.

We see that when more CONAN interactions are added to the network, the AUC increases. This means that when more interactions are extracted in the near future, the performance of the network will improve significantly. The quality of the interactions is also very high. This is reflected in the fact that only 415 interactions extracted by CONAN were in the true negative list of the Gene Interaction Network. We also see that TP interaction pairs are significantly ($P < 0.05$) underrepresented in the set of TN interaction pairs. The results we obtained at this time are encouraging.

For a second experiment with the same data, we tried to correlate the PPIs to the journal impact factors. The impact factors were extracted from a list provided by sciencegateway (www.sciencegateway.org/impact). The impact factors are from the year 2003, but given that the impact factor does not change dramatically over time, the analysis is also valid for the most recent impact factors. The first results show, that the likelihood scores improve with the impact factor. The likelihood ratio for interactions extracted from journals with an impact factor below five is 85.74 (for the highest bin), while the likelihood ratio for interactions extracted from journals with an impact factor above five is 156.14. For interactions extracted from journals that have no impact factor assigned, the likelihood score is 81.47. So it can easily be seen that the impact factor of a journal definitely influences the quality of PPIs. This can have several reasons: Editors in high impact journals might demand from the authors to put down known gene/protein symbols and might demand

clearer writing from the authors. Another reason might be that authors in high-impact journals tend to write clearer articles and the interactions are therefore easier extractable. A completely different explanation is that the quality of the experiments is higher in high-impact journals, leading to better quality of the data. Another reason for this observed influence could be that most of the interactions of the gold standard set are also derived from high-impact journals and therefore the increase in likelihood ratios could be biased. In Table 11.3, the distribution of the extracted interactions with regard to the impact factors can be seen.

Table 11.3. Extracted Interactions

ImpactFactor	UniquePairs	TotalPairs
<5	25,602	77,261
>5	20,154	68,709
unknown	11,411	27,131

For future analysis, we also want to see how the “real” extracted interaction data compares to the cocitation data mentioned earlier. This will give good conclusions on what method is preferable and how these interactions can be further refined using the impact factor of the journals.

References

- [1] <http://gnosis.cx/download/gnosis>.
- [2] <http://www.research.att.com/sw/tools/graphviz>.
- [3] G.D. Bader, D. Betel, and C.W. Hogue. BIND: the Biomolecular Interaction Network Database. *Nucleic Acids Res.*, 31(1):248–250, 2003.
- [4] L. Franke, H. van Bakel, L. Fokkens, E. D. de Jong, M. Egmont-Petersen, and C. Wijmenga. Reconstruction of a functional human gene network with an application for prioritizing positional candidate genes. *Am J Hum Genet*, 78(6):77–85, 2005.
- [5] D. Heckerman. A tutorial on learning with bayesian networks. In Michael Jordan, editor, *Learning in Graphical Models*. MIT Press, 1997.
- [6] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D’Eustachio, E. Schmidt, B. de Bono, B. Jasal, G.R. Gopinath, G. R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Res.*, 33(Database Issue):D428–D432, 2005.
- [7] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.*, 28(1):27–30, 2000.
- [8] D.J. LaCount, M. Vignali, R. Chettier, A. Phansalkar, R. Bell, J. R. Hesselberth, L. W. Schoenfeld, I. Ota, S. Sahasrabudhe, C. Kurschner, S. Fields, and R.E. Hughes. A protein interaction network of the malaria parasite plasmodium falciparum. *Nature*, 438(7064):103–107, 2005.
- [9] I. Lee, S.V. Date, A.T. Adai, and E.M. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306(5701):1555–1558, 2004.
- [10] M. Middendorff, E. Ziv, and C.H. Wiggins. Inferring network mechanisms: the drosophila melanogaster protein interaction network. *Proc Natl Acad Sci USA*, 102(9):3192–3197, 2005.
- [11] S. Peri, J.D. Navarro, T.Z. Kristiansen, R. Amanchy, V. Surendranath, B. Muthusamy, T.K. Gandhi, K.N. Chandrika, N. Deshpande, S. Suresh, B.P. Rashmi, K. Shanker, N. Padma, V. Niranjana, H.C. Harsha, N. Talreja, B.M. Vrushabendra, M.A. Ramya, A.J. Yatish, M. Joy, H.N. Shivashankar, M.P. Kavitha, M. Menezes, D.R. Choudhury,

- N. Ghosh, R. Saravana, S. Chandran, S. Mohan, C.K. Jonnalagadda, C.K. Prasad, C. Kumar-Sinha, K.S. Deshpande, and A. Pandey. Human protein reference database as a discovery resource for proteomics. *Nucleic Acids Res.*, 32(Database Issue):D497–D501, 2004.
- [12] J.F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G.F. Berriz, F.D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, N. Klitgord, C. Simon, M. Boxem, S. Milstein, J. Rosenberg, D.S. Goldberg, L.V. Zhang, S.L. Wong, G. Franklin, S. Li, J.S. Albala, J. Lim, C. Fraughton, E. Llamosas, S. Cevik, C. Bex, P. Lamesch, R.S. Sikorski, J. Vandenhaute, H.Y. Zoghbi, A. Smolyar, S. Bosak, R. Sequerra, L. Doucette-Stamm, M.E. Cusick, DE. Hill, F.P. Roth, and M. Vidal. Towards a proteome-scale map of the human protein-protein interaction network. *Nature*, 437(7062):1173–1178, 2005.

PART V

DISCUSSION

Chapter 12

DISCUSSION AND CONCLUSIONS

In this chapter, I present a summary of the results. I conclude and provide directions for further research.

12.1. Summary of the Results

In this thesis, I presented CONAN, a text mining system that can automatically extract and display the following information: protein/gene names, protein point mutations, protein-protein interactions and biologically interesting keywords. I presented the applications where CONAN is integrated: a command-line tool to query CONAN, a webserver and the integration of protein-protein interaction data in a human gene interaction network. With the integration into a human gene interaction network, also "hidden" information can be extracted. CONAN was developed integrating some of the newest and most interesting algorithms and methods into one framework. When evaluating CONAN, we see that CONAN is one of the top-performing text mining systems at the moment. A broad range of data can be extracted in a reasonable amount of time. This data proves to be of high-quality. CONAN will be expanded so that it covers ultimately the whole MEDLINE. For the future, new techniques and methods can be easily integrated in CONAN.

12.2. Conclusions

Let us go back to the initial research question that was posed in the beginning. **Is it possible to construct a system which is suited to extract hidden information out of text while being as complete as possible?**

CONAN shows that the answer is: Yes! CONAN belongs to the top-performing text mining Systems at the moment, while providing more information than most other systems. The main advantages of CONAN are:

- CONAN, while being a complete system, can still extract hidden information and this approach does not harm the quality of the data produced.
- With the evaluation of the results (see Part IV of this thesis), we show that CONAN belongs to the top systems, when looking on real-world data.
- With the large number of articles already processed, we also show that CONAN is able to cover the complete MEDLINE, given enough CPU and memory power.
- The information extraction part is, in my personal opinion, the big strength of CONAN. No other system so far can extract interactions accurately while being this quick and producing so much useful data. This is also reflected in the Gene Interaction Network Application (see Section 11.4), where CONAN will play an important part in the future. The next steps in this project include the use of CONAN to curate and update the Gold Standard for the network.

So we can definitely state that CONAN is a complete system that produces large amounts of high-quality data.

In the Introduction, several technical challenges were raised:

- 1 *Does the combination of text-mining classifiers and algorithm increase the performance?*
- 2 *Can such a system still reach high Precision and Recall rates, resulting in a good F-measure?*
- 3 *Is it possible to include the information extracted by such a system in other systems?*
- 4 *Can the extracted information be presented in such a way that the information is helpful for experimentalists?*

Now we can answer these questions:

- 1 In Part IV we show that the combination of classifiers in CONAN really improves the performance. This can be seen in the evaluations performed on CONAN. We also show that by adding classifiers (e.g. Boosting Classifier), the performance still improves and the addition of more classifiers could be useful. However, it also has to be said that with the introduction of a new classifier, the Recall of CONAN decreases, the Precision increases and the overall F-Measure increases as well.
- 2 Throughout Part IV, we show that the Recall and Precision rates of CONAN are in the top of state-of-the-art text mining systems. Though the Precision and Recall rates are not perfect, almost no Text Mining system

performs better than CONAN at the moment. We also show in Section 10.2.6 that the performance of such a text mining System is not far behind a human reader, the difference in the F-Measure is just about 5 %. Text mining Systems in the future could automatically produce corpora with high accuracy.

- 3 We show throughout this thesis, that the combination of different data sources makes it really easy to link information together. Good examples are the Ensembl and UniProt identifiers. The results of CONAN can easily be integrated in other systems, using these identifiers. Moreover, CONAN is constructed in a way that new identifiers and/or methods can be introduced very easily.
- 4 In Chapter 11, we show that the display of text mining results is not an easy task. This is caused by the fact that experimentalists often have their own preferences on how data should be presented. However, we tried to make our applications intuitive and easy-to-use. We hope that experimentalists will use CONAN in the future and that it will be a useful tool for the community.

At this point of time, the CONAN webserver is available for testing purposes at <http://h094.niob.knaw.nl/conan>. A set of example queries which can be executed can be found in the Appendix.

12.3. Further Research

In my personal opinion, text mining in the biomedical domain will become more and more important over the coming years. As new algorithms and new techniques for the extraction of information out of text will become available, the performance of these text mining systems will definitely improve.

At this point I present some application areas, where text mining Data could be used in the future. A list of existing text mining systems can be found in Chapter 2. I pick up some challenges Hirschman et al. and Krallinger et al. raise in their reviews [2, 3].

The first challenge that text mining faces is the extraction of biological pathways. A biological pathway consists not only of interactions between proteins (as this problem has already been solved), but also of interactions between drugs, proteins and other molecules. First, the protein, drug and molecule names have to be identified with refined NER methods, which should also be able to detect the other molecule names. Next, the interactions between the different molecules have to be recognized. Finally, the relationships between the different interaction events have to be recorded. Especially the last point is not very clear, as there are no good methods to relate different interactions to each other.

Krallinger et al. [3] state that the text mining systems of the future should not work on abstracts alone, but on full-text collections. They also state, similar to Hirschman, that the future development in text mining will be most likely concerned with the construction of networks and interactions through intermediate entities, followed by the proposal of new functions. This could also be termed “Text Mining KD”. Although some applications like SUISEKI [1] have been developed, they are not yet ready for practical use. Moreover, these applications should also be able to extract indirect relations (e.g. protein-disease, protein-other molecule) automatically. This is also what is mentioned in the review by Hirschman.

The biggest challenge of text mining lies in genomic data: results from text analysis should be combined with evidence from experiments and genome analysis to improve the accuracy of results and to generate additional knowledge beyond what is known solely from literature. The Gene Interaction Network presented in Section 11.4 is only the first step in this process.

Another big challenge in the future will be how new data is to be integrated in existing systems. For example, a system like the Gene Network presented in Section 11.4, needs updating whenever new data becomes available. Making these updates automated will definitely be a major point in the future.

For the development of biomedical articles and abstracts in general, I hope that the approach of authors, publishers and editors will change in the future. Together with database curators, they should be eager to unify Protein/Gene Name Nomenclature. Moreover, authors should be urged to put protein names in a special format in their publications. I envision a metafile that is distributed with each publication, holding the information about protein names and their position in the text. This would revolutionize text mining, as Named Entity Recognition would become needless. Then, Text Miners could concentrate on the bigger problems: how to combine data and how to represent extracted data.

Finally, I want to end with a comment made by Bill Hersh (Oregon Health & Science University) while discussing the measures which should be used in the upcoming TREC: “To me, this (the endless discussion about scoring schemes, author’s comment) is a very short-sighted view to take of the research we are doing. I recognize that we all need to publish papers and get grant funding based on successful work, but I also think we need to keep our target on what we really want to do, which is build information systems that will improve the work of real people.”

References

- [1] C. Blaschke and A. Valencia. The potential use of SUISEKI as a protein interaction discovery tool. *Genome Inform Ser Workshop Genome Inform.*, 12:123–134, 2001.
- [2] L. Hirschman, J.C. Park, J. Tsujii, L. Wong, and C.H. Wu. Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, 18(12):1553–1561, 2002.
- [3] M. Krallinger and A. Valencia. Text-mining and information-retrieval services for molecular biology. *Genome Biol.*, 6(7):224, 2005.

PART VI

APPENDIX

Chapter 13

APPENDIX

13.1. Definitions

- **Gene:** The fundamental physical and functional unit of heredity. A gene is an ordered sequence of nucleotides located in a particular position on a particular chromosome that encodes a specific functional product (ie, a protein or RNA molecule).
- **Protein:** Building blocks of life. Proteins make up living material but also hormones or enzymes. Proteins are molecules composed of one or more chains of amino acids in a specific order. This order is determined by the base sequence of nucleotides in the gene coding for the protein. The three-dimensional structure of a protein is critical for its function.
- **Mutation:** A permanent change, a structural alteration, in the DNA or RNA. Mutations can be caused by many factors including environmental insults such as radiation and mutagenic chemicals. Mutations are sometimes attributed to random chance events.
- **Protein-Protein Interaction (PPI):** affect all processes in a cell. Proteins rarely function in isolation. It has been proposed that all proteins in a given cell are connected through an extensive network, where non-covalent interactions are continuously forming and dissociating. These interactions are mostly physical interactions.
- **Database:** A collection of information stored in a computer in a systematic way, such that a computer program can consult it to answer questions. The software used to manage and query a database is known as a database management system (DBMS). The properties of database systems are studied in computer science.

- MEDLINE: bibliographical information database, forms the largest subset of PubMed. MEDLINE contains approximately 13 million citations at this moment.
- Abstract: An abbreviated summary of a research article, review, or any in-depth analysis of a particular subject or discipline, and is often used to help the reader quickly ascertain the papers purpose. When used, an abstract always appears at the beginning of a manuscript, acting as the point-of-entry for any given scientific paper or patent application.
- Literature: In this thesis, literature is not synonymous with the original definition of literature. The original definition states that the term literature is only valid when the character of the text is purely fictional. In the biomedical world, however, the term literature is used for the whole of biomedical abstracts and articles which are available.
- Text Mining: Process of extracting interesting and non-trivial information and knowledge from unstructured text. Text mining is a young interdisciplinary field which draws on information retrieval, data mining, machine learning, statistics and computational linguistics. In some definitions, Text Mining describes the finding of overlooked connections in text (see also Section 2.5.2).
- Literature Mining: Literature Mining is the generic term for all methods which extract any information out of literature.
- Information Retrieval: Searching for information in documents or searching for documents themselves.
- Information Extraction: Automated Methods for extracting facts from text.
- Natural Language Processing: Natural language processing (NLP) is a sub-field of artificial intelligence and linguistics. It studies the problems inherent in the processing and manipulation of natural language. NLP also addresses the problem of natural language understanding devoted to making computers "understand" statements written in human languages. NLP techniques are used very often in literature mining.
- Corpus: A collection of texts, either full-text or abstracts.
- Impact factor: Is a statistical measure on how often an article published in a certain journal, is cited in other journals. A high impact factor means that this journal is often cited and is thus "high standard".
- Likelihood: The likelihood (or sampling distribution) quantifies the likelihood of the data given the unknown model parameters.

- **Likelihood Ratio Test:** Is a statistical test relying on a test statistic computed by taking the ratio of the maximum value of the likelihood function under the constraint of the null hypothesis to the maximum with that constraint relaxed. Given a Model H_0 (assumed to be true) and a Model H_1 , the ratio of l_1/l_0 is called the likelihood ratio and it amounts to the odds that H_1 indeed is correct as opposed to H_0 .
- **Regular Expression:** Often called a pattern, is an expression that describes a set of strings. They are usually used to give a concise description of a set, without having to list all elements. For example, regular expression "gray|grey" matches gray or grey.
- **Recall:** number of relevant and found documents/terms, divided by the number of all relevant documents/terms.
- **Precision:** number of relevant and found documents/terms, divided by the number of all found documents/terms.
- **F-Measure:** Harmonic Mean of Recall and Precision, computed by $(2rp)/(r+p)$.

Note: As Gene and Protein names cannot automatically be distinguished from each other, the term Gene Name / Protein Name is interchangeable in this thesis.

13.2. Example Queries

This section contains example queries which can be executed using the test-version of the CONAN webserver. To perform those searches, please use the "Advanced Searching" in CONAN.

- **Protein Name:** p53, TNFalpha, p38
- **Keyword:** "Binding Sites", "Molecular Structure" (please note that the quotes are compulsory for multi-word terms).
- **PMID:** 14871385, 14730436, 15367858
- **Interaction / PMID and Interaction / All:** p53, TNFalpha, p38
- **Mutation:** Rab21, alpha-synuclein, dnaA
- **Gene Ontology:** 0005694, 0003676, 0008151
- **Ensembl ID:** ENSG00000197822, ENSG00000173402, ENSG00000183091
- **Synonym:** CAD1_HUMAN, IL4_HUMAN
- **MeSH:** "Cell Line", "Rats"

13.3. DTD

13.3.1 Medline DTD

```

<!ELEMENT Article ((Journal — Book),
                    %ArticleTitle.Ref;,
                    Pagination,
                    Abstract?,
                    Affiliation?,
                    AuthorList?,
                    Language+,
                    DataBankList?,
                    GrantList?,
                    PublicationTypeList,
                    VernacularTitle?,
                    DateOfElectronicPublication?)
>
<!ELEMENT DataBankList (DataBank+)>
<!ELEMENT DataBank (DataBankName, AccessionNumberList?)>
<!ELEMENT DataBankName (#PCDATA)>
<!ELEMENT AccessionNumberList (AccessionNumber+)>
<!ELEMENT AccessionNumber (#PCDATA)>
<!ATTLIST DataBankList
           CompleteYN (Y | N) "Y"
>
<!ELEMENT GrantList (Grant+)>
<!ELEMENT Grant (%GrantID.Ref;, %Acronym.Ref;, %Agency.Ref;)>
<!ELEMENT GrantID (#PCDATA)>
<!ELEMENT Acronym (#PCDATA)>
<!ELEMENT Agency (#PCDATA)>
<!ELEMENT Abstract (%Abstract;)>
<!ATTLIST GrantList
           CompleteYN (Y | N) "Y"
>
<!ELEMENT Journal (%ISSN.Ref;
                  JournalIssue,
                  Coden?,
                  Title?,
                  ISOAbbreviation?)>
<!ELEMENT ISSN (#PCDATA)>
<!ELEMENT JournalIssue (Volume?, Issue?, %PubDate.Ref;)>

```

```

<!ELEMENT Volume (#PCDATA)>
<!ELEMENT Issue (#PCDATA)>
<!ELEMENT PubDate (%pub.date;)>
<!ELEMENT Coden (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT ISOAbbreviation (#PCDATA)>
<!ELEMENT DateOfElectronicPublication (#PCDATA)>
<!ELEMENT MedlineJournalInfo (Country?,
                               MedlineTA,
                               MedlineCode?,
                               NlmUniqueID?)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT MedlineTA (#PCDATA)>
<!ELEMENT MedlineCode (#PCDATA)>
<!-- Sometime in the future, MedlineCode will change to
      NLMUniqueID -->
<!ELEMENT Book (%PubDate.Ref;
               ,
               Publisher,
               Title,
               AuthorList?,
               CollectionTitle?,
               Volume?)>
<!ELEMENT Publisher (#PCDATA)>
<!ELEMENT ArticleTitle (#PCDATA)>
<!ELEMENT CollectionTitle (#PCDATA)>
<!ELEMENT VernacularTitle (#PCDATA)>
<!ELEMENT PublicationTypeList (PublicationType+)>
<!ELEMENT PublicationType (#PCDATA)>
<!ELEMENT Language (#PCDATA)>
<!ELEMENT AuthorList (Author+)>
<!ELEMENT Author ((%author.name;), Affiliation?)>
<!ELEMENT Affiliation (#PCDATA)>
<!ATTLIST AuthorList
               CompleteYN (Y | N) "Y"
>
<!ELEMENT Pagination ((StartPage, EndPage?, MedlinePgn?))>
<!ELEMENT StartPage (#PCDATA)>
<!ELEMENT EndPage (#PCDATA)>
<!ELEMENT MedlinePgn (#PCDATA)>

```

```
<!ELEMENT MeshHeadingList (MeshHeading+)>
<!ELEMENT MeshHeading (Descriptor, SubHeading*)>
<!ELEMENT Descriptor (#PCDATA)>
<!ATTLIST Descriptor
    MajorTopicYN (Y | N) "N"
>
<!ELEMENT SubHeading (#PCDATA)>
<!ATTLIST SubHeading
    MajorTopicYN (Y | N) "N"
>
<!ELEMENT PMID (#PCDATA)>
<!ELEMENT NlmUniqueID (#PCDATA)>
```

13.3.2 CONAN DTD

```
<!DOCTYPE collection >
<!ELEMENT abstract>
  <!ELEMENT abstract_header>
  <!ELEMENT abstract_id (abstract+)>
  <!ELEMENT abstract_file (#PCDATA) >
  <!ELEMENT abstract_file_position (#PCDATA) >
  <!ELEMENT PMID (#PCDATA) >
<!ELEMENT method_overview >
  <!ELEMENT method (#PCDATA) >
  <!ELEMENT parameters (#PCDATA) >
<!ELEMENT abstract_body >
  <!ELEMENT term_id (term+)>
  <!ELEMENT term_text (#PCDATA) >
  <!ELEMENT term_pos_start (#PCDATA) >
  <!ELEMENT term_pos_end (#PCDATA) >
  <!ELEMENT term_tag (#PCDATA) >
  <!ELEMENT mutation (#PCDATA) >
  <!ELEMENT term_organism(#PCDATA) >
  <!ELEMENT term_score (#PCDATA) >
  <!ELEMENT term_synonyms(#PCDATA) >
  <!ELEMENT synonym_score(#PCDATA) >
  <!ELEMENT goa_id (#PCDATA) >
  <!ELEMENT goa_function (#PCDATA) >
  <!ELEMENT goa_process (#PCDATA) >
  <!ELEMENT goa_component(#PCDATA) >
  <!ELEMENT interaction(#PCDATA) >
  <!ELEMENT Protein1 (#PCDATA) >
  <!ELEMENT Protein2 (#PCDATA) >
  <!ELEMENT Interaction_Type (#PCDATA) >
```

13.3.3 Regular Expressions used in Protein-Protein Interaction Extraction

Note: A and B denote the respective protein names which are parsed to the interaction finding method.

```

"(A.*fail|unable).*to.*(interact|associate|bind|bound|complex|precipitat|phosphorylat).*B)"
"(A(\S*\s+)\{0,3\}\S*(n\`t|not)(\S*\s+)\{0,3\}interact(\S*\s+)\{0,3\}with(\S*\s+)\{0,3\}B)"
"(A(\S*\s+)\{0,4\}\S*interact(\S*\s+)\{0,3\}\S*with.*but not with(\S*\s+)\{0,3\}\S*B)"
"(no.*interaction (between|among) \w*A(\S*\s+)\{0,6\}\S*B)"
"(no (\S*\s+)\{0,3\}\S*B(\S*\s+)\{0,6\}\S*A.*interaction)"
"(A.*(n\`t|not).\{0,18\}complex.*B)"
"(A.*(n\`t|not)(\S*\s+)\{0,6\}\S*complex.*B)"
"(A.*(n\`t|not)(\S*\s+)\{0,6\}\S*associat.*with.*B)"
"(A.*(n\`t|not) \S*associat.* with \S*sB)"
"(A.*(n\`t|not)(\S*\s+)\{0,6\}(bind|bound).*B)"
"(A(\S*\s+)\{0,6\}(bind|bound)\S* to.*but not(\S*\s+)\{0,3\}to(\S*\s+)\{0,4\}\S*B)"
"(via.\{0,9\}A.*and.*B)"
"(A.\{0,9\}a.\{0,9\}B.\{0,9\}homologue)"
"(A\S* mutant.*B)"
"(require(\S*\s+)\{0,4\}\S*A.*and.*B(\S*\s+)\{0,3\}in)"
"((neither|nor) (\S*\s+)\{0,3\}\S*A(\S*\s+)\{0,5\} nor (\S*\s+)\{0,3\}\S*B)"
"(heterodimer(\S*\s+)\{0,4\}\S*A(\S*\s+)\{0,4\}\S*B)"
"(link.\{0,24\}A.\{0,24\}to.\{0,24\}B)"
"(A.\{0,24\}attach.*to.* B)"
"(A.*dock.*with.*B)"
"(A(\S*\s+)\{0,4\}B(\S*\s+)\{0,6\}\S*are(\S*\s+)\{0,3\}\S*component)"
"(A(\S*\s+)\{0,6\}B(\S*\s+)\{0,6\}two(\S*\s+)\{0,6\}\S*component)"
"(A(\S*\s+)\{0,6\}target of B)"
"(A(\S*\s+)\{0,3\}B(\S*\s+)\{0,4\}\S*subunit)"
"(A(\S*\s+)\{0,6\}proteolysis(\S*\s+)\{0,6\}\S*B)"
"(A(\S*\s+)\{0,6\}required for the proteolysis of(\S*\s+)\{0,6\}\S*B)"
"(A(\S*\s+)\{0,4\}regulatory subunit\S* of(\S*\s+)\{0,4\}\S*B)"
"(A(\S*\s+)\{0,4\}\S*B(\S*\s+)\{0,4\}\S*heterodimer)"
"(A.*recruit.*B.*for.*pathway)"
"(A(\S*\s+)\{0,6\}\S*regulate(\S*\s+)\{0,6\}\S*B)"
"(A(\S*\s+)\{0,3\}encode(\S*\s+)\{0,6\}\S*B)"
"(A(\S*\s+)\{0,3\}inhibit(\S*\s+)\{0,6\}\S*B)"
"(A(\S*\s+)\{0,3\}\S*B(\S*\s+)\{0,6\}components of)"
"(components of (\S*\s+)\{0,5\}A(\S*\s+)\{0,3\}\S*B)"
"(A\S*.\S*B)"
"(\b|\w*A|\w*-\w*B|\w*\b)"
"(\b|\w*A|\w*-\w*B|\w*\b|\w*-\w*)"
"(A|\w*-\w*B|\w*)"
"(-|\w*A|\w*-\w*B)"
"(\b|\w*A|\w*-\w*B|\w*\b)"
"(A.*interact.*B)"
"(A.*interact.*with.*B)"
"(A\S* mutant.*interact.*with.*B)"
"(A.*interact.*with.*(whereas|while|but).*B.*not)"
"(A interact(|s|ed) with B)"
"(A.*cannot.*only.*interact.*with.*B)"
"(A.*interacting.*B)"
"(A\S*interacting.\{0,24\}B)"
"(A.*interacting(\S*\s+)\{0,6\}\S*B)"
"(A(\S*\s+)\{0,9\}\S*interact(\S*\s+)\{0,9\}\S*B)"
"(A.*interaction.*B)"
"(interact.*A.*B)"
"(interaction.*A.*B)"
"(interaction.*of.*A.*(with|and).*B)"
"(interaction of A(,| with| and) B)"
"(interaction.*(between|among).*A.*and.*B)"
"(interaction (between|among) A and B)"
"(interact(\S*\s+)\{0,3\} both with \S*A\S* and \S*B)"

```

"(A.*B.*interact)"
 "(A.*B.*interaction)"
 "(A(\S*\s+){0,9}\S*B(\S*\s+){0,6}\S*interaction)"
 "(A(\S*\s+){0,3}and(\S*\s+){0,6}\S*B(\S*\s+){0,6}\S*interact(?!ion))"
 "(A\S* and B\S* interact(?!(\S*\s+){0,4}with))"
 "(A.*B.*interact.*in.*complex.*with)"
 "(interact.*with.*A.*with B)"
 "(protein.*interact.*with.*A.*B)"
 "(A.*complex.*B)"
 "(A.{0,24}complex(|es) with.*\w*B)"
 "(A\w* complex(|es) with \w*B)"
 "(A.*form.*complex.*with.*B)"
 "(A complex.*contain.*B)"
 "(A.*(has|have).*activity.*in.*complex.*contain.*B)"
 "(A.*B.*complex)"
 "(neither.*A.*nor.*B.*complex.*formation)"
 "(A(\S*\s+){0,9}\S*B(\S*\s+){0,9}\S*complex)"
 "(A(\S*\s+){0,3}\S*B(\S*\s+){0,3}\S*complex)"
 "(A(\S*\s+){0,6}\S*B \S*complex)"
 "(A.*B.*form.*\w*complex)"
 "(A.*B.*each.*form.*complex.*with)"
 "(A.*component.*of.*B.*complex)"
 "(A.*and.*B.*part.*of.*complex)"
 "(A.*B.*subunit.*of.*complex)"
 "(A.*B.*reconstitute.*complex)"
 "(A.*B.*comprise.*complex)"
 "(A.*function.*with.*B.*in.*complex)"
 "(A(\S*\s+){0,4}require(\S*\s+){0,5}\S*B.*complex)"
 "(A.{0,12}B.{0,12}complex)"
 "(complex.*A.*B)"
 "(complex(\S*\s+){0,4}contain(?!ing)(\S*\s+){0,3}A(\S*\s+){0,3}B)"
 "(form.*complex with.*A.*B)"
 "(complex.*A.*and.*B)"
 "(complex.*form.*between \w*A and .*B)"
 "(complex.*contain.*A.*B)"
 "(complex A.*B)"
 "(complex.*in.*which.*A.*B)"
 "(complexes.*containing.*A.*B)"
 "(complex of.*A \w*/\S*B)"
 "(complex.{0,12}A \w*/\S*B)"
 "(complex with(\S*\s+){0,3}A(\S*\s+){0,6}by(\S*\s+){0,3}B)"
 "(complex of(\S*\s+){0,3}proteins(\S*\s+){0,3}\S*A(\S*\s+){0,4}\S*B)"
 "(complex(\S*\s+){0,4}A(\S*\s+){0,4}\S*B)"
 "(A.*associat.*B)"
 "(A.* associat \w* with .*B)"
 "(A associate \w* with .*B)"
 "(A associate \w* with B)"
 "(A.* is .*associated.* with .*B)"
 "(associat.*A.*B)"
 "(association(\S*\s+){0,3}(between|among|of)(\S*\s+){0,3}\S*A(\S*\s+){0,3}B)"
 "(association between A and .*B)"
 "(association between(\S*\s+){0,3}\S*A \S* and \S*B)"
 "(association of(\S*\s+){0,3}\S*A \S* with .*B)"
 "(association of(\S*\s+){0,3}\S*A \S* and .*B)"
 "(A.*B.*associat)"
 "(A(\S*\s+){0,6}\S*B(\S*\s+){0,6}\S*association)"
 "(A(\S*\s+){0,3}\S*B \w*-associated protein)"
 "(A.*B association)"
 "(A.{0,24}B association)"
 "(A.* its.* association .*with .*B)"
 "(A.* and .*associated .*B)"
 "(A.*activat(?!ion).*B)"
 "(A \w*-dependent.*activation.*of.*B)"
 "(A.*for.*activation.*of.*B)"
 "(A.*is(\S*\s+){0,3}for activation of(\S*\s+){0,3}\S*B)"
 "(A(\S*\s+){0,6}\S*activat(?!ion)(\S*\s+){0,6}\S*B)"
 "(activation of.*A.*by.*B)"
 "(A.*B.*activat)"
 "(A.*phosphorylat.*B)"

"(A\S*(\S*\s+){0,6}(|de|de-)phosphorylat(e|ion|es|ed) . *B)"

"(A\S* (|de|de-)phosphorylate(|s|d) \S*B)"

"(A.*modulat.*B.*phosphorylation)"

"(A\w*-dependent.*phosphorylation.*of.*B)"

"(phosphorylat.*A.*B)"

"(phosphorylation.*of \S*A.* by .*B)"

"(phosphorylation\S*(\S*\s+){0,4}A.*convert.*B.*into)"

"(A.*B.*phosphorylat)"

"((co)|(-)(|immuno)|(-)precipitat.*A.*B)"

"(precipitation.*A.*B)"

"(precipitat(?|ion) .*A.*B)"

"(A.*(co)|(-)(|immuno)|(-)precipitat\w* with.*B)"

"(A.*(co)|(-)(|immuno)|(-)precipitat\w*.*B)"

"(A.*B.*(co)|(-)(|immuno)|(-)precipitat)"

"(A.*conjugat.*B)"

"(A(\S*\s+){0,3}conjugate.*with.*B)"

"(A(\S*\s+){0,6}conjugate(\S*\s+){0,6}\S*B)"

"(conjugat.*A.*B)"

"(conjugat(\S*\s+){0,6}A(\S*\s+){0,6}\S*B)"

"(A.*B.*conjugat)"

"(A\w*-B\w* conjugat)"

13.3.4 Databases used in BLAST search

Table 13.1. Databases used in BLAST search

<i>Database</i>	<i>Database</i>
Acquired Abnormality	Laboratory Procedure
Alga	Laboratory or Test Result
Amino Acid Sequence	Lipid
Amphibian	Mammal
Anatomical Abnormality	Mental or Behavioral Dysfunction
Anatomical Structure	Molecular Biology Research Technique
Animal	Molecular Function
Antibiotic	Molecular Sequence
Archaeon	Natural Phenomenon or Process
Bacterium	Neoplastic Process
Biologically Active Substance	Neuroreactive Substance or Biogenic Amine
Bird	Nucleic Acid, Nucleoside, or Nucleotide
Body Location or Region	Nucleotide Sequence
Body Part, Organ, or Organ Component	Organ or Tissue Function
Body Space or Junction	Organic Chemical
Body Substance	Organism
Body System	Organism Attribute
Carbohydrate	Organism Function
Cell	Organophosphorus Compound
Cell Component	Pathologic Function
Cell Function	Pharmacologic Substance
Cell or Molecular Dysfunction	Invertebrate
Chemical	Physiologic Function
Chemical Viewed Functionally	Plant
Chemical Viewed Structurally	Protein
Clinical Drug	Qualitative Concept
Congenital Abnormality	Quantitative Concept
Disease or Syndrome	Receptor
Eicosanoid	Reptile
Element, Ion, or Isotope	Research Activity
Embryonic Structure	Research Device
Enzyme	Rickettsia or Chlamydia
Experimental Model of Disease	Sign or Symptom
Finding	Spatial Concept
Fish	Species
Functional Concept	Steroid
Fungus	Substance
Gene or Genome	Temporal Concept
Genetic Function	Therapeutic or Preventive Procedure
Hazardous or Poisonous Substance	Tissue
Hormone	Vertebrate
Human	Virus
Immunologic Factor	Vitamin
Indicator, Reagent, or Diagnostic Aid	genedata
Injury or Poisoning	

Index

- Abbreviation, 57
- AbGene, 8
- Abstract, 67, 143
- AdaBoost, 56
- Agreement Scores, 84, 92, 98
- Ambiguity, 7, 9
- Anaphora, 8
- Anaphoric Expressions, 8
- Annotator Agreement, 84, 92, 98
- Area under the Curve, 125, 127
- Attribute, 54
- AUC, 125, 127

- Bayesian Classifier, 124
- Bayesian Network, 124
- bigram, 56
- Binning, 124, 127
- BioCreative, 93
- Biological Process, 33
- BioPerl, 53
- Biopython, 41, 53
- BLAST, 39, 69
- BLAST Extraction, 40
- BLAST parameters, 41
- boosted Regression, 55
- BoosTexter, 56
- Boosting, 55, 75

- C4.5, 56
- Cellular Component, 33
- Classification, 54
- Co-occurrence, 10, 126
- Cocitation, 126
- Command Line Tool, 109
- CONAN, 109
- CONAN Output, 71
- Corpus, 89, 93, 95, 96, 143
- Corpus Construction, 89, 96
- Cross References, 68
- Cross-references, 31, 32, 36, 38
- Cross-Validation, 128

- Data Integration, 70
- Data Integrator, 70
- Database, 143
- Definitions, 143
- Dictionary, 57

- ENSEMBL, 29, 38, 68
- ENSEMBL Identifier, 31
- Entrez, 5
- Evaluation, 88

- F-Measure, 14, 83, 85, 143
- Feature, 54

- GAPSCORE, 8
- Gene, 143
- Gene Interaction Network, 123
- Gene Ontology, 32, 36, 68
- GO, 32
- GO codes, 33
- GOA, 68
- Gold Standard, 124
- Graphs, 122

- Harmonic Mean, 83

- IE, 10
- iHop, 10
- Impact Factor, 128, 143
- Indexing, 110
- Information Extraction, 10, 43–45, 143
- Information Retrieval, 5
- Inter Annotator Agreement, 84, 92, 98
- Inter-Species Ambiguity, 9
- Interaction, 45, 95, 96, 104, 119, 120
- Interaction Network, 123
- International Protein Index, 68
- Intra-species Ambiguity, 9
- IPI, 38, 68

- IR, 5
- kappa value, 84
- KDD, 11
- Keyword Extraction, 40
- Keyword Search, 116
- Keywords, 116
- Kinase Pathway, 13
- Knowledge Discovery, 11, 125
- Knowledge Problem, 3
- Lexicon, 6
- Likelihood, 143
- Likelihood Ratio, 128
- Likelihood Ratios, 128, 143
- Likelihood Scores, 128
- Literature Mining, 143
- Medical Subject Headings, 28
- MEDLINE, 27, 66
- MeSH, 28, 67, 75
- Metathesaurus, 33
- Molecular Function, 33
- Mutation, 44, 71, 73, 104, 117, 143
- Mutations, 69
- n-Gram, 56
- Named Entity Recognition, 8, 42, 43
- Natural Language Processing, 6, 10
- NER, 8, 42, 43
- NLP, 6, 10, 42
- NLProt, 8, 73
- Noun Phrase, 6
- Ontology, 32, 33
- Overfitting, 54, 128
- Parser, 7, 42
- Partial Matches, 9
- Perl, 53
- Perl Script, 110, 112
- PPI, 143
- Pre-Processing, 69
- PreBIND, 10
- Precision, 14, 81, 143
- Predictor Variable, 54
- Prodisen Corpus, 89
- Protein, 143
- Protein annotation, 35
- Protein Interactions, 12
- Protein Name Abbreviations, 35
- Protein Tagging, 42, 43, 69, 73, 99, 116
- Protein-Protein Interaction, 45, 75, 95, 96, 104, 119, 120
- PubCrawler, 8
- PubGene, 10
- PubMed, 27
- Python, 53
- Python Script, 110, 112
- Queries, 113
- Recall, 14, 81, 143
- Regular Expression, 44, 45, 143
- Research Question, 4
- Retrieval, 5
- s-Gram, 56
- SDI Services, 7
- Semantic Network, 34, 40
- Semantic Types, 40
- SLOPPY, 9, 95
- Small Datasets, 13
- Species Extraction, 44
- STRICT, 9, 95
- Strong Hypothesis, 58
- Support Vector Machines, 43
- SwissProt, 35, 44
- Syntax, 6
- Tags, 49
- Target Variable, 54
- Test Set, 54
- Text Mining, 4, 10, 143
- Text Mining Definition, 11
- Text Mining Problems, 13
- Text Mining Systems, 5
- Textpresso, 13
- Tissue Extraction, 44
- Tokens, 44
- Training Set, 54
- UMLS, 33
- UMLS Metathesaurus, 40
- UniParc, 32
- UniProt, 31, 35, 38
- UniRef, 32
- Variable, 54
- Verb Phrase, 6
- Vocabulary, 33
- Weak Learner, 56
- World Problem, 3
- XML, 49, 71, 72, 110
- XML Attributes, 51
- XML Delcarations, 51
- XML Elements, 51
- XML Entities, 51
- XML Nodes, 52
- XML Path, 52
- XML Tag, 72
- XPath, 112
- XPath Expression, 52

Curriculum Vitae

Rainer Malik was born on June 19th, 1978 in Linz, Austria. He grew up in Traun, Austria, where he attended elementary school and high school. Traun is still the place where his parents live. After high school, in 1996, the author started studying biology with the special subject of molecular biology in Salzburg, at the University of Salzburg, Austria. In 2000, he started writing his master's thesis in the group of Prof. Dr. Manfred J. Sippl at the Center for Applied Molecular Engineering at the University of Salzburg. The title of the thesis is "Protein Structure Comparison and the Application in the CASP experiment". In June 2002, the author obtained the master's degree under the supervision of Prof. Dr. Peter Lackner from the University of Salzburg. In September 2002, Rainer Malik started his Ph.D. studies at the University of Utrecht in the group and under the supervision of prof. dr. A.P.J.M. Siebes (Large and Distributed Databases).

Publications in the Ph.D period:

- Guryev V, Berezikov E, Malik R, Plasterk RH, Cuppen E.(2004) Single nucleotide polymorphisms associated with rat expressed sequences., *Genome Res.* 2004 Jul;14(7):1438-43.
- Knobbe A, Ho E, Malik R (2005). ILP Challenge 2005: The Safari MRDM environment. ILP05, LNAI 3625, 2005
- Malik R, Siebes A (2005). CONAN: An Integrative System for Biomedical Literature Mining. EPIA05, LNAI 3808, 248-259, 2005.
- Malik R, Siebes A (2006). Combination of Text-Mining Algorithms increases the performance. *Bioinformatics. In Press.*
- Krallinger M, Malik R, Valencia A (2006). The Prodisen corpus: exploring the construction and applications of a protein description corpus. Proceedings of ISMB SIG on biomedical text data mining. *Submitted.*

Zusammenfassung in deutscher Sprache

Text Mining ist ein neues und aufregendes Forschungsgebiet, das in den letzten Jahren einen überraschenden Boom erfuhr. Ziel ist, interessante und wichtige Information aus Text zu holen (= zu extrahieren) und diese gut und deutlich darzustellen. Text Mining findet Verwendung in vielen verschiedenen Gebieten, wie zum Beispiel der SPAM-Filterforschung, im Gesetzestextbereich und auch in der biomedizinischen Domäne. In dieser Dissertation beschränke ich mich ausschliesslich auf dieses letztgenannte Gebiet.

In den letzten Jahren hat die Anzahl der wissenschaftlichen Publikationen und der Fachmagazine stark zugenommen. Es ist ein exponentieller Anstieg zu verzeichnen. Mittlerweile geht man von ca. 20 Millionen Artikeln in PubMed/MEDLINE aus. PubMed/MEDLINE ist die primäre Datenbank für biologische und biomedizinische Fachartikel. Diese enorme Masse an Information ist für einen Wissenschaftler, der diese Information benötigt, kaum zu überblicken. Text Mining Systeme sind daher von hoher Relevanz, um die Vielfalt der Information in überschaubare Bahnen zu lenken.

Das Ziel ist, ein System zu erstellen, das die für (Molekular-)Biologen und Mediziner wichtigste Information automatisch aus Text extrahiert. Zu dieser Information zählt das Erfassen von Protein- und Gennamen, das Erfassen von Mutationen, das Extrahieren von Protein-Protein-Wechselwirkungen und das Erfassen von biologisch wichtigen Schlüsselwörtern wie zum Beispiel Krankheiten oder Gewebsformen. Diese Information soll so gut wie möglich dargestellt werden, um experimentell arbeitenden Forschern die Arbeit zu erleichtern. Ausserdem soll so ein System so komplett wie möglich sein und alle verfügbaren Artikel prozessieren können.

Zum heutigen Zeitpunkt bestehen viele Text Mining Systeme, die aber entweder zuwenig Information aus den Texten holen oder aber zuwenig Texte prozessieren. Deshalb wurde von mir ein System entwickelt, das sowohl so komplett wie nur möglich ist, als auch hochqualitative Information liefert. Darüber hinaus sollten die Resultate einfach dargestellt werden, sodass jeder Wissenschaftler die gewünschte Information ohne Probleme finden kann. Dieses System nennen wir CONAN.

In dieser Dissertation zeige ich, wie so ein System konstruiert wird. Weiterhin enthält diese Dissertation Evaluierungen dieses Systems. Diese Evaluierungen werden mithilfe sogenannter Corpora (=Testsysteme) bewerkstelligt. Im Laufe dieser Evaluierung zeige ich zudem, wie so ein Corpus entwickelt wird. Schlussendlich lege ich dar, in welchen Anwendungen CONAN seinen Einsatz findet. Diese Anwendungen sind: ein Kommandozeilenprogramm, ein Webserver und die Integration von Interaktionsdaten in ein menschliches Interaktionsnetzwerk.

Die Resultate, die CONAN erreicht sind sehr gut. In allen Evaluierungen, die durchgeführt wurden, ist CONAN im Spitzenfeld zu finden. Dies zeigt,

dass CONAN gut konstruiert wurde und die Resultate eine hohe Qualität zeigen.

In Kapitel 1 gebe ich eine Einleitung in das Text Mining Feld. Viele wichtige Begriffe und Definitionen wie Natural Language Processing oder Information Extraction werden hier erklärt. Weiter gebe ich einen genauen Überblick, was alles in dieser Dissertation enthalten ist.

In Kapitel 2 erkläre ich viele Methoden und Datenbanken, die bei CONAN zum Einsatz kommen. Diese Methoden sind die Bausteine von CONAN (BLAST, AbGene, NLProt, PreBIND und MuText) und die Techniken, die im Laufe von CONAN implementiert wurden (Support Vector Machines und Boosting). Darüberhinaus erkläre ich alle Datenbanken, die in CONAN verwendet werden (Ensembl, UniProt, etc.).

In Kapitel 3 erkläre ich, wie CONAN aufgebaut ist. Ich erkläre, wie die Eingabe und die Ausgabe des Programmes genau aussehen.

Kapitel 4 beschäftigt sich mit der Evaluierung des Systems und dem Aufbau dieser Testsysteme. Ferner werden interessante Ergebnisse von CONAN dargestellt.

In Kapitel 5 zeige ich, in welchen Anwendungen CONAN zur Zeit im Einsatz ist. Eine Konklusio und ein Ausblick auf das Gebiet des Text Mining bilden Abschluss meiner vorliegenden Arbeit.

Samenvatting in het Nederlands

Tekst Mining is een nieuw en interessant onderzoeksgebied, dat de laatste jaren steeds meer in de belangstelling staat. Het doel van tekst mining is om interessante en belangrijke informatie uit tekst te halen (=extraheren) en deze informatie correct en helder weer te geven. Tekst Mining wordt toegepast in verschillende domeinen, zoals bijvoorbeeld voor het bouwen van SPAM filters, in het juridische domein en voor biomedische toepassingen. In dit proefschrift wordt alleen het laatste onderdeel behandeld.

De laatste jaren is zowel in het aantal wetenschappelijke publicaties als in het aantal wetenschappelijke tijdschriften een exponentiële groei te zien. Inmiddels gaat men uit van ongeveer 20 miljoen artikelen in PubMed/MEDLINE. PubMed/MEDLINE is de belangrijkste bron voor biologische en biomedische artikelen. Door de enorme hoeveelheid aan informatie is het voor wetenschappers bijna onmogelijk om een overzicht te krijgen van relevante informatie. Daarom zijn tekst mining systemen belangrijk; voor een overzicht van de aanwezige informatie.

Het doel is, een systeem te construeren, dat de belangrijkste informatie automatisch uit de tekst haalt. Dit omvat het extraheren van: gen- en proteïnenamen, van mutaties, van proteïne-proteïne interacties en van biologisch belangrijke sleutelwoorden zoals namen van ziektes of weefsel. Deze informatie moet zo goed als mogelijk weergegeven worden, om het werk van

experimentele wetenschappers makkelijker te maken. Bovendien moet een tekst mining systeem zo compleet als mogelijk zijn en alle beschikbare artikelen kunnen verwerken.

Op het huidige tijdstip bestaan vele text mining systemen, maar die halen of te weinig informatie uit de tekst of baseren de weergegeven informatie op een te klein deel van de collectie. Daarom is door mij een systeem ontwikkelt, dat naast hoog kwalitatieve informatie tevens zo compleet als mogelijk is. Bovendien zijn de resultaten heel overzichtelijk weergegeven zodat elke wetenschapper de gewenste informatie kan vinden. Dit systeem noemen wij CONAN.

In dit proefschrift laat ik zien, hoe CONAN is opgebouwd. Tevens bevat dit proefschrift verschillende experimentele evaluaties van CONAN op diverse corpora. Bovendien beschrijven we hoe een corpus (=test systeem) wordt opgebouwd. Vervolgens bespreek ik de toepassingen van CONAN, deze bestaan ondermeer uit: een command-line interface, een webserver en de integratie van interactie data in een interactie netwerk.

De met CONAN bereikte resultaten zijn uitstekend. In alle experimentele evaluaties behoort CONAN tot de top. Dit betekent dat CONAN goed ontworpen is en de resultaten van hoge kwaliteit zijn.

Hoofdstuk 1 is een inleiding in tekst mining. Veel belangrijke termen en definities zoals Natural Language Processing of Information Extraction worden in dit hoofdstuk beschreven. Verder geef ik een precies overzicht over de verdere inhoud van dit proefschrift.

In hoofdstuk 2 beschrijf ik de methodes en databases waaruit CONAN is opgebouwd. Dit zijn naast de bouwstenen van CONAN zoals BLAST, AbGene, NLProt, PreBIND en MuText tevens de methodes die in CONAN geïmplementeerd zijn zoals Support Vector Machines und Boosting. Daarnaast worden ook de databases beschreven die gebruikt worden door CONAN zoals Ensembl, UniProt, etc.

In hoofdstuk3 bespreek ik het ontwerp van CONAN. De in- en uitvoer van CONAN wordt hierbij uitgebreid besproken.

Hoofdstuk 4 bevat de experimentele evaluatie van CONAN, teven wordt de constructie van testsystemen (corpora) behandeld en interessante resultaten getoond.

In hoofdstuk 5 worden de mogelijke toepassingen van CONAN beschreven. Ten slotte trek ik conclusies en geef een overzicht van verdere ontwikkelingen in het onderzoeksgebied, tekst mining.

SIKS Dissertatiereeks

=====
1998
=====

- 1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects
- 1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information
- 1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations
within the Language/Action Perspective
- 1998-4 Dennis Breuker (UM)
Memory versus Search in Games
- 1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting

=====
1999
=====

- 1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling;
Automated modelling of Quality Change of Agricultural Products
- 1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets
- 1999-3 Don Beal (UM)
The Nature of Minimax Search
- 1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects
- 1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven
Specification of Network Information Systems
- 1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems
- 1999-7 David Spelt (UT)
Verification support for object database design
- 1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent
Mechanism for Discrete Reallocation.

=====
2000
=====

- 2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance
- 2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management
- 2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie;
een procesbenadering en actorperspectief.

- 2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design
- 2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.
- 2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication
- 2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management
- 2000-8 Veerle Coup (EUR)
Sensitivity Analysis of Decision-Theoretic Networks
- 2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization
- 2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations,
Algorithms and Architecture
- 2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

====
2001
====

- 2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks
- 2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models
- 2001-3 Maarten van Someren (UvA)
Learning as problem solving
- 2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with
Instance-Based Boundary Sets
- 2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style
- 2001-6 Martijn van Welie (VU)
Task-based User Interface Design
- 2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization
- 2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.
- 2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models,
Views of Packages as Classes
- 2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice
BRAHMS: a multiagent modeling and simulation language
for work practice analysis and design
- 2001-11 Tom M. van Engers (VUA)
Knowledge Management:
The Role of Mental Models in Business Systems Design

=====
2002
=====

- 2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis
- 2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections
- 2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval
- 2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining
- 2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments
inhabited by Privacy-concerned Agents
- 2002-06 Laurens Mommers (UL)
Applied legal epistemology;
Building a knowledge-based ontology of the legal domain
- 2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
- 2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative
E-Commerce Ideas
- 2002-09 Willem-Jan van den Heuvel(KUB)
Integrating Modern Business Applications with Objectified Legacy Systems
- 2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble
- 2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications
- 2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems
- 2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications
- 2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and
Verifying Multi-Agent Systems
- 2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling
- 2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications
- 2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

=====
2003
=====

- 2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments
- 2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems
- 2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
- 2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology
- 2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach
- 2003-06 Boris van Schooten (UT)
Development and specification of virtual environments
- 2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks
- 2003-08 Yongping Ran (UM)
Repair Based Scheduling
- 2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour
- 2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction
between medium, innovation context and culture
- 2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
- 2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval
- 2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models
- 2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
- 2003-15 Mathijs de Weerd (TUD)
Plan Merging in Multi-Agent Systems
- 2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to
Digital Media Warehouses
- 2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing
- 2003-18 Levente Kocsis (UM)
Learning Search Decisions

=====
2004
=====

- 2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

- 2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business
- 2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
- 2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures
- 2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity
- 2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques
- 2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes
- 2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politieële gegevensuitwisseling en digitale expertise
- 2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning
- 2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects
- 2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies
- 2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents
- 2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play
- 2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium
- 2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining
- 2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning
- 2004-17 Mark Winands (UM)
Informed Search in Complex Games
- 2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models
- 2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval
- 2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

=====
2005
=====

- 2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications

- 2005-02 Erik van der Werf (UM)
AI techniques for the game of Go
- 2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language
- 2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data
- 2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing
- 2005-06 Pieter Spronck (UM)
Adaptive Game AI
- 2005-07 Flavius Frasinca (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems
- 2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications
- 2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages
- 2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
- 2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
- 2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry
- 2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
- 2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
- 2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes
- 2005-16 Joris Graaumans (UU)
Usability of XML Query Languages
- 2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components
- 2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks
- 2005-19 Michel van Dartel (UM)
Situated Representation
- 2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives
- 2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

=====
2006
=====

- 2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting

- 2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations
- 2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems
- 2006-04 Marta Sabou (VU)
Building Web Service Ontologies
- 2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines
- 2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools
for Graphical Service Modeling
- 2006-07 Marko Smiljanic (UT)
XML schema matching – balancing efficiency and effectiveness by means of clustering
- 2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web
- 2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor's Opinion
- 2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems
- 2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types

