Mining Relations from the Biomedical Literature

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Dr. Rer. Nat.
im Fach Informatik

eingereicht an der
Mathematisch-Wissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Dipl. Inf. Jörg Hakenberg
Düren, 15. Juli 1975

eingereicht am: 11. März 2009
Tag der mündlichen Prüfung: 11. September 2009

Abstract

Text mining deals with the automated annotation of texts and the extraction of facts from textual data for subsequent analysis. Such texts range from short articles and abstracts to large documents, for instance web pages and scientific articles, but also include textual descriptions in otherwise structured databases. This thesis focuses on two key problems in biomedical text mining: relationship extraction from biomedical abstracts —in particular, protein–protein interactions—, and a pre-requisite step, named entity recognition —again focusing on proteins.

This thesis presents goals, challenges, and typical approaches for each of the main building blocks in biomedical text mining. We present out own approaches for named entity recognition of proteins and relationship extraction of protein-protein interactions. For the first, we describe two methods, one set up as a classification task, the other based on dictionary-matching. For relationship extraction, we develop a methodology to automatically annotate large amounts of unlabeled data for relations, and make use of such annotations in a pattern matching strategy. This strategy first extracts similarities between sentences that describe relations, storing them as consensus patterns. We develop a sentence alignment approach that introduces multi-layer alignment, making use of multiple annotations per word. For the task of extracting protein-protein interactions, empirical results show that our methodology performs comparable to existing approaches that require a large amount of human intervention, either for annotation of data or creation of models.

Zusammenfassung

Textmining beschäftigt sich mit der automatisierten Annotierung von Texten und der Extraktion einzelner Informationen aus Texten, die dann für die Weiterverarbeitung zur Verfügung stehen. Texte können dabei kurze Zusammenfassungen oder komplette Artikel sein, zum Beispiel Webseiten und wissenschaftliche Artikel, umfassen aber auch textuelle Einträge in sonst strukturierten Datenbanken. Diese Dissertationsschrift bespricht zwei wesentliche Themen des biomedizinischen Textmining: die Extraktion von Zusammenhängen zwischen biologischen Entitäten —das Hauptaugenmerk liegt dabei auf der Erkennung von Protein-Protein-Interaktionen—, und einen notwendigen Vorverarbeitungsschritt, die Erkennung von Proteinnamen.

Diese Schrift beschreibt Ziele, Herausforderungen, sowie typische Herangehensweisen für alle wesentlichen Komponenten des biomedizinischen Textmining. Wir stellen eigene Methoden zur Erkennung von Proteinnamen sowie der Extraktion von Protein-Protein-Interaktionen vor. Zwei eigene Verfahren zur Erkennung von Proteinnamen werden besprochen, eines basierend auf einem Klassifikationsproblem, das andere basierend auf Suche in Wörterbüchern. Für die Extraktion von Interaktionen entwickeln wir eine Methode zur automatischen Annotierung großer Mengen von Text im Bezug auf Relationen; diese Annotationen werden dann zur Mustererkennung verwendet, um anschließend die gefundenen Muster auf neuen Text anwenden zu können. Um Muster zu erkennen, berechnen wir Ähnlichkeiten zwischen zuvor gefundenen Sätzen, die denselben Typ von Relation/Interaktion beschreiben. Diese Ähnlichkeiten speichern wir als sogenannte 'consensus patterns'. Wir entwickeln eine Alignmentstrategie, die mehrschichtige Annotationen pro Position im Muster erlaubt. In Versuchen auf bekannten Benchmarks zeigen wir empirisch, dass unser vollautomatisches Verfahren Resultate erzielt, die vergleichbar sind mit existierenden Methoden, welche umfangreiche Eingriffe von Experten voraussetzen.

Contents

1. Introduction

1.1. Motivation

The field of molecular biology has experienced a shift from experiments to collect data to experiments to test hypotheses in recent years [Blagosklonny and Pardee, 2002]. Yet, biological databases grow rapidly in information vital to experimental research in the life sciences. They provide access to vast amounts of data that can be queried and narrowed down in manageable ways, extracting pieces of information sought by each particular user. Such pieces help raising or rejecting hypotheses, and lead to further experiments if necessary. The number of biological databases and entries therein grow steadily, but first and foremost, most of the information are added by database curators and not the original researchers. The reason is that still, the choice for publishing experimental findings are journal articles. This comes with the reputation gained by publishing results in articles, as opposed to databases, the unfamiliarity of researchers with databases, and the difficulties that come with trying to express complex results in structured form. Although some journals require authors to submit their data to renowned databases, a study by Noor et al. [2006] showed that up to 15% (depending on the journal) of DNA sequence data was never submitted. Many authors submitted data described in their paper, but did not mention accession numbers, for instance, for retrieving the data in GenBank.

Information contained in publications outnumber data deposited in databases by far. As an example, consider databases on protein-protein interactions, such as IntAct, MINT, and DIP [Hermjakob et al., 2004, Chatr-aryamontri et al., 2007, Xenarios et al., 2001]. The number of entries contained in either one lies between 56,000 (DIP) and 110,000 (IntAct). In contrast, Fundel et al. [2007] found 150,000 protein-protein interactions in one million PubMed abstracts[1] (this number includes duplicates). This number might grow to over 600,000 for all circa 9.6 million abstracts available, and even more if one considers full-text publications[2]. These estimates are roughly consistent with experiments by Donaldson et al. [2003], who predicted that in PubMed abstracts, 269,000 protein-protein interactions were discussed for human, mouse, and yeast alone (out of 1.88 million abstracts containing proteins from either species). Despite sheer numbers, another factor when it comes to hypothesis-driven experimental research is the reliability of data. Regarding the aforementioned databases, as much as 70% of the data (IntAct) originate from large-scale, high-throughput screening assays (such as yeast-

[1]PubMed is a major resource for literature searches, indexing more than 4500 journals, with citations for almost 18 million life scientific articles.

[2]Currently, more than 2.3 million full-text articles are open-access; an additional 6 million are linked from PubMed citations.

two hybrid screens and bimolecular fluorescence complementation), which are much less reliable than small-scale experiments (for instance, co-immunoprecipitation and tandem affinity purification). Regarding peer-reviewed journal articles, however, one can assume that only reliable information is discussed in an abstract; data from large-scale experiments tends to appear in tables, figures, or as supplementary information only.

In addition to abstracts and full-text articles, highly informative fields in database often contain pieces of text or references to publications. However, accessing the information in text and extracting data is very effort intensive when done manually. Any form of automated systems helping database curators and researchers to access recent developments can lead to improvements in terms of time and costs. Even modest improvements in efficiency are potentially significant [Berleant et al., 2003, Kerrien, 2007].

Applications of text mining to molecular biology are among the most studied during the last years. A large quantity of systems proposed focused on the recognition of protein (or gene) names in texts, the extraction of protein–protein and protein–gene interactions from texts, or the functional annotation of proteins based on evidences found in texts. These applications help researchers and database curators to access data provided in publications in an automated way. Tasks for text mining systems include finding relevant literature for a given gene or protein, finding relevant information on a gene, and extracting data on complex relationships [Altman and Klein, 2002]. Information about relationships and full pathways are required to understand intra- and inter–cellular processes at a system level and are substantial to the development of biological models [Kitano, 2002]. An important component for most biological functions are interactions between proteins, which are the focus for applications of methods discussed in this thesis. Protein-protein interactions govern signal transduction, molecular transportation, help form the structure of a cell, among others, and thus are crucial to almost every process in living cells. Huge efforts are focusing on the detection of protein-protein interactions, with a multitude of detection methods available. However, the most reliable among those methods (mainly small–scale experiments such as co–immunoprecipitation) are quite costly, whereas comparatively cheap large–scale experiments yield high false positive rates (yeast two–hybrid screens are an example). Findings of such detection experiments are published in journal articles; and only slowly the data find their way into databases by manual curation, where they can be found by other researchers. One of the goals in biomedical text mining thus is to help speeding up this curation process. Ultimate goals in molecular biology are quantitative modeling of processes, were also enzymes and other substances are involved, understanding co–expression of genes, deeper understanding of signal transduction pathways, and interdependencies of various pathways. A further task for text mining would thus be to not only annotate proteins with functions, processes, and locations (categories from the Gene Ontology [Gene Ontology Consortium, 2006]), but to find experimental data supporting such annotations, as well as information on the type of experiment, species, experimental conditions, among others.

As mentioned before, most recent results in molecular biology and related fields are published in written language: publications in journals, books, conference contributions, and web pages. In contrast, even renowned biological databases lack these data for some time after they are published: to ensure quality, most databases are manually curated, at the cost of quantity and recentness. While databases present an easy access and enable complex querying, the access to facts in publications is hindered by several problems. The huge amount of publications would require researchers to monitor a large list of journals, pick all relevant articles, and read these to learn about important facts discussed therein. Also, complex queries over natural language text, as compared to structured databases, is still impossible for the lack of methods to automatically identify facts in text. Certain facts are hard to express in predefined, structured ways, while it is comparatively easy to explain them in natural language. This is reflected by databases organized in a way that important fields contain mere textual descriptions; for instance, descriptions of a protein's implications in diseases or functions in UniProt.

Complexity and variability of natural language complicate accessing facts in publications. Although parsers for sentences gain in quality of their predictions, is is still a hard problem to relate objects to each other in a precise and well–defined way (which would be the pendant to structured databases.) Objects in biomedical texts can refer to proteins, genes, diseases, and so on, and we will show in Chapter 2 (see Section 2.1.1) why finding such named entities is a difficult task. Relations between objects (an interaction between proteins, a protein's association with a disease, etc.) can be expressed in an infinite number of ways using. Section 2.1.4 in Chapter 2 will present the difficulties for the task of extracting such relations from text.

1.2. Contribution

As main contribution of this thesis, we present a method to automatically extract and refine sets of language patterns from unannotated data. We introduce the idea of multiple sentence alignment to find commonalities between similar patterns and express these in consensus patterns. As alignment strategy, we propose multi–layer alignment that takes into account not only part–of–speech tags (as other systems do), but also includes tokens and word stems to ensure precision.

- As a prerequisite for subsequent components, we first introduce two methods for named entity recognition, focusing on protein and gene names. One is based on a model learned from training data and solves the general problem; the other works with a pre–defined dictionary and is useful when lists of named entities are known beforehand. The later approach also implicitly helps in solving the problem of mapping recognized entities to database identifiers, which is necessary for our subsequent components. We further discuss our successful approach to gene mention normalization.
- We introduce sentence alignment as an inexact matching strategy to compute the similarity between two sentences. Most work shown previously transforms patterns into regular expressions for matching, resulting in binary decisions rather

than quantitative measures.

- We show a new method to extract language patterns from unannotated data. Systems proposed previously were based on hand–crafted pattern sets or needed annotated data to extract patterns.
- As these initial patterns do not generalize well, we present a novel method to refine sets of language patterns to allow for larger coverage. We therefore introduce a clustering method based on sentence similarity.
- We introduce multiple sentence alignment to find commonalities between groups of similar sentences. Such commonalities can then be expressed in consensus patterns, reflecting observation frequencies to aid alignment at application time.
- We present an adapted implementation of alignment; it is the first of its kind (to our knowledge) to take into account not only a single type of information (such as amino acids for protein sequences, or part–of–speech tags for sentences), but has access to multiple annotation layers (that is, part–of–speech tags, tokens, and word stems).

1.3. Outline of this Thesis

Chapter 2 introduces goals for and techniques in text mining. The main task in this thesis is information extraction, and we focus on methods in machine learning and natural language processing that contribute to this task as major building blocks.
Chapter 3 presents two approaches to the recognition and identification of proteins names. We discuss related work on named entity recognition, for protein names and other biomedical named entities.
Chapter 4 introduces the concept of language patterns. We show how they can be represented, generated, and compared to each other and to arbitrary text.
Chapter 5 presents our approach to relation mining for protein–protein interactions based on language patterns. We show how we extract patterns from unannotated corpora. We discuss comparable approaches to relation mining (for protein interactions and other tasks).
Chapter 6 presents the evaluation of the overall system for relation mining. We start with an introduction to evaluation metrics and strategies. The system was evaluated on three different external corpora, and we also present a comparison to a system that uses hand–curated patterns instead of automatically generated patterns.
Chapter 7 describes AliBaba, a tool for visualization of extracted associations by means of browsing a graph constructed from binary relations found in a document collection.
Chapter 8 summarizes this thesis and its main contributions and concludes with an outlook to future work.

2. Text Mining

Text mining aims at the extraction of knowledge from written language in the form of books, reports, journal publications, conference contributions, and so on. We refer to written language as semi-structured, as it follows a syntax, but allows for such a huge variety that it is deemed infeasible to formalize this syntax with current techniques. It is thus not an easy task to access the information given in texts automatically, and we need sophisticated techniques to analyze natural language and exploit as much of its structure as possible. It will become clear in the remainder of this thesis that it is not sufficient to analyze a text by itself. On the contrary, we need to acquire and integrate knowledge even from external sources to grasp contents properly, adding to the complexity.

The data contained in texts (we will focus on scientific texts in the following) are of different granularities. For one, each text deals with one or more topics, and we can thus categorize it by them. Next, each text provides information on certain objects, which we can try to find, and which fall into categories as well. Then, each text describes different types of associations between such objects. Associations can form transitive chains, which we can find (almost) only across multiple texts. Combined, all these facts finally might contain hidden or unknown information, such as unexpected connections between distant objects. The different levels of granularity are as follows (with examples):

- topic/category: epidemiology, molecular biology; yeast-2-hybrid assay, gene expression analysis
- objects/entities: genes, DNA, RNA, proteins, species, tissue, cells
- associations/interactions: gene-gene product, protein-protein, gene-disease associations
- interdependencies/cascades: metabolic networks, signaling pathways

Text mining falls into several steps for the extraction of information on different levels. Horizontally grouping text mining into steps, we can directly assign the aforementioned levels of information. Each respective step has to identify the corresponding information in texts, relying on information found during former steps:

- information retrieval: texts relevant to a topic
- information extraction: objects/entities, relations between objects
- collection-wide analysis: chains of relations, networks
- knowledge discovery: previously unknown knowledge

We will explain these steps in more detail in the following.

Information retrieval

We use information retrieval (IR) as a filter to identify texts relevant to the given task, prior to information extraction. Filtering out irrelevant documents not only reduces the data quantity to be analyzed in complex and time-consuming, subsequent steps. It also narrows down the number of different topics discussed in the remaining texts to relevant and related ones. Word sense and synonymity complexities (see next section) also get reduced. Ultimately, relevance filters lead to much less false positive identifications of terms. A conservative information retrieval strategy, on the other hand, reduces the recall of texts, objects, and relations discussed therein. What is lost in an early step cannot be recovered later.

Information extraction

This thesis focuses on information extraction (IE): relation mining, and, as a prerequisite, named entity recognition (NER). Named entity recognition focuses on extracting and identifying words that refer to relevant objects. This provides an overview of the content, and feeds into subsequent steps that deal with the extraction of facts on these objects (relation mining). We will motivate each sub-task of NER, show problems, and describe basic concepts in the following Section 2.1. Chapter 3 presents our own approaches for recognizing named entities, and Chapter 5 describes our approaches for relation mining.

Collection-wide analysis

Collection-wide analysis (CWA) brings together information extracted from large quantities of texts. Information extraction steps in CWA are quite similar to those used on single texts. Objects and relations between these objects have to be identified. Transitive relations, that ultimately yield complex networks of associations, mostly are spread over multiple texts. Main problems for CWA arise from ambiguities in word senses, as well as synonymies. Different authors, different communities and topics, even different publication formats, bring different usage of nomenclatures with them. In most books, one will find a proper explanation for an abbreviation, but in abstracts, you might not. While one author uses a certain name to refer to an object, other authors might use quite different synonyms for the same object. As CWA tries to bring together facts spread over many texts, techniques to identify the exact object behind a name are necessary. For CWA in the biomedical domain, PubMed[1] is the primary resource. This citation index contains information (abstracts, authors, links to full texts) on almost 17 million publications since the 1865[2]. Circa 9.5 million citations have abstracts, and 8.3 million have links to full-texts, with 2.3 million of these being open-access and thus freely available for text mining.

[1]Available at http://www.pubmed.org

[2]See http://www.nlm.nih.gov/bsd/licensee/baselinestats.html

Knowledge discovery

Knowledge discovery (KD) analyzes large sets of data to find patterns, to deduce previously unknown aspects, or to generate hypotheses. Currently, there are few solutions and applications concerning text mining, but are well known in other fields (market basket analyses as a prominent example). Proper organization of knowledge extracted from text collections is a mandatory requirement for discovering new structures. A way to analyze disjoint sets of texts (for instance, on two different topics) was introduced by Swanson [1986, 1988]. Co-occurrence calculations of words in article titles lead to intermediate concepts that in some cases are real and not only random links. They ultimately connect concepts from each disjoint set to each other using these links. Swanson was able to find a correlation between, for instance, magnesium and migraine, that was previously unknown but later confirmed in experiments. Logic reasoning over ontologies is becoming a focus of interest. Text mining helps to automatically assign actual facts to the concepts (or vice versa) or even to extract the ontologies automatically from large textual repositories.

As opposed to the horizontal grouping of text mining components according to tasks, vertical grouping shows the techniques that we use at various levels, mainly, machine learning and natural language processing. We introduce both in Sections 2.2 and 2.3, respectively. We present the main concepts and define important terms. We will use these techniques in the remainder of this thesis.

2.1. Information Extraction

Information extraction is concerned with finding facts in texts. For our purposes, such facts are either single objects mentioned in a text, or associations between these objects. Named entity recognition, named entity identification, and word sense disambiguation all deal with the identification and instantiation of such objects, in the following called entities. Named entity recognition tries to find all words that refer to entities of a relevant class (for instance, genes) in a text. Named entity identification maps all entities not only to a class, but to exact instances of this class (a particular gene). Word sense disambiguation tries to resolve cases in which multiple classes or instances are possible (as happens with homonyms and acronyms). Finally, relation mining brings entities together by searching for associations between these entities, in the underlying text. In this section, we focus on introducing named entity recognition and relation mining. Chapters 3 and 5 will pick up these subjects and explain our own strategies as well as related work in more detail.

2.1.1. Named Entity Recognition

A prerequisite of all sophisticated approaches towards information extraction is the recognition of named entities. Named entities refer to the objects of interest discussed in a text. In life scientific publications, these entities may stem from different vocabularies used by scientific communities. We can group most entities together into entity classes, which assign a semantic role to them, such as

- anatomy,
- cells, cell types, cell lines,
- compounds,
- DNA, RNA, amino acid sequences,
- drugs,
- enzymes,
- genes,
- organism and subspecies,
- proteins, or
- tissues.

Other entities describe instances of these classes and their current states and functions:

- (biological) process,
- (cellular) location,
- diseases,
- (molecular) function,
- phenotypes, or
- symptoms.

In addition, publications contain information on how these data were gathered, in general referring to:

- conditions,
- environment,
- settings,
- techniques/equipment, or
- treatments.

Only upon recognizing these entities (instances of classes) correctly, one can start to extract information on how they relate to each other. Named entity recognition (NER) includes finding ("spotting") each term as well as classifying it into (one or more) classes (protein, cell, etc.).

Definition 2.1 (Named entity)**.**
A named entity is a classified instance of a given terminology.

Definition 2.2 (Entity class)**.**
An entity class is a semantic role that a named entity can have.

Named entity recognition has to deal with the immense variety of names for all instances from all entity classes. For multiple reasons, it is not possible to simply list all names that belong to each entity class and report every matching occurrence in a text:

- lack of naming conventions: no standardized nomenclature/syntax that points to specific instances/classes
- no exhaustive collection of all names for all instances of a class is available
- variations: many terms refer to the same instance
- ambiguity: names might, taken for themselves, be valid for more than one entity class
- abbreviations: might have different long forms that refer to different meanings

Naming conventions and collections

For many domains and sub-domains in biomedicine, there are no established naming conventions. This is especially true for domains with continuous new developments and discoveries, and a high knowledge transfer between research areas. Such domains and their sublanguages both are highly productive, as new entities (even classes) are discovered permanently and new terms are formed ad hoc (see later in this section). In addition, usage of terms in or across communities is discontinuous. Better fitting terms occur, for instance describing the function of an entity more properly. Formerly symbolic names turn into illustrative, spoken language terms, for example when gene functions are determined years after their discovery (example: ”inhibitor of apoptosis”), or when names describe phenotypes (”dickkopf”, ”wingless”, ”lethal”). In other domains, such conventions exist, and only few deviations occur. For instance, main research in anatomy was done decades ago, there appear almost no new discoveries, and the community has agreed on a vocabulary, consisting mainly of Latin words. One could measure the productiveness (concerning discovery of objects and functions) and degree of naming conventions of a community by looking at university lectures: lectures in anatomy cover topics established about a hundred years ago, compared to around five to ten years in molecular biology.

Naming conventions often appear as vocabularies or nomenclatures for specific sub-domains. The HGNC/HUGO[3] nomenclature contains approved symbols for human genes [Wain et al., 2002, Eyre et al., 2006] and is coordinated with corresponding data from mouse, MGI[4] [Eppig et al., 2005]. The Saccharomyces Genome Database, SGD [Issel-Tarver et al., 2002], proposes a standard for systematic naming of genes and proteins from yeast. This naming standard encodes the exact locus, open reading frame, and even strain of *Saccharomyces cerevisiae*. However, such names are hardly human readable, and most genes also have additional gene names. SGD urges all researchers to adhere to their guidelines using abbreviations for phenotypes when searching for more memorizable gene names.

The Saccharomyces Genome database[5] proposes a syntactic naming convention for genes and proteins. Such gene symbols contain three italic lower case letters and a number, corresponding proteins are non-italic, begin with an upper case letter and have the suffix ’p’ (*cyp27*, Cyp27p). This standard even includes specifications for naming alleles and phenotypes, helping to immediately identify them [Cherry, 1995]. However, there are multiple proteins produced from one gene, resulting from splice variants with (slightly or completely) different functions, thus representing different entities, which should be distinguishable from others. This turns out to be problematic in particular for collection-wide analyses, where entities need to be distinguished exactly.

Naming guidelines remain useless until they are accepted by the respective research communities. Some publishers decided to accept manuscripts for publication only if

[3]HUGO Gene Nomenclature Committee, see http://www.gene.ucl.ac.uk/nomenclature/
[4]Mouse Genome Informatics, see http://www.informatics.jax.org
[5]SGD, see http://www.yeastgenome.org

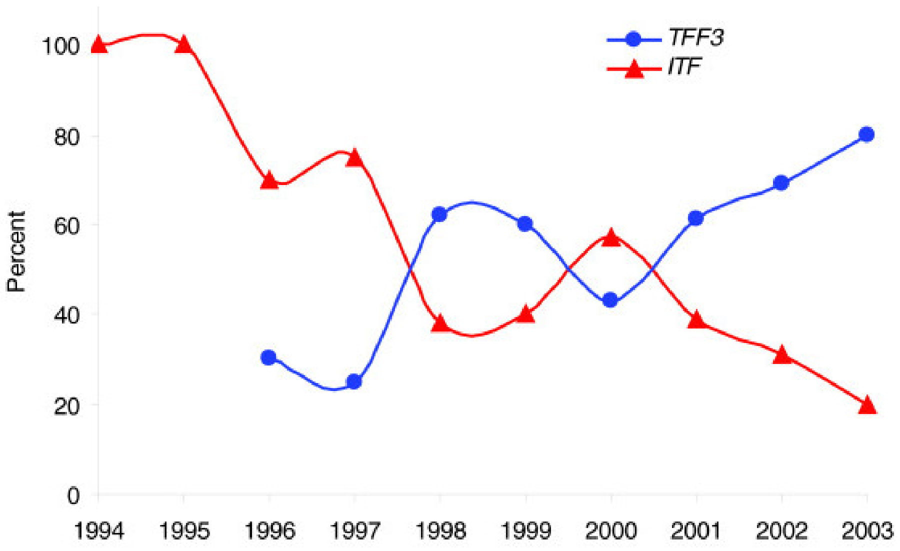


Figure 2.1.: Usage of two different names for "Intestinal trefoil factor 3", official name "TFF3", over time. After the introduction of the standardized forms, the form that occurred first still prevailed. Taken from Tamames and Valencia [2006], Figure 2a.

the naming of genes etc. concurs with standards. PLoS journals[6] ensure during reviewing that publications adhere to strict guidelines: for instance, publications have to present unique identifiers for all proteins discussed. Tamames and Valencia [2006] conducted an experiment where they checked for synonym usage and usage of proposed standard names. They compared gene names found in MEDLINE to names proposed by HUGO to evaluate the relative success of HUGO guidelines. Even ten years after the proposal of HUGO standard names in 1994, only 44% of the official names predominate other aliases (1994: 35%). In addition, Tamames and Valencia found that only 38% of highly cited genes (occurring in more than 50 publications) were preferentially named according to the standard. Figure 2.1 shows an example for the usage of two different names over time. 14% of the official names have still not been used at all. Even if nomenclatures tend to get used more often, there is still a large amount of publications that was published before these conventions; therefore, they do not adhere to these standards.

Variations, abbreviations, and ambiguity

While some entity classes might essentially be broken down to name lists (including spelling variations and known synonyms, but still excluding overly exotic variations and uncommon names), this is not true for all entity classes. In particular, it is hard to recognize 'functional entities' (see list on page 8), such as phenotypes and symptoms. These instances most often appear as complex descriptions rather than a few,

[6]Public Library of Science, see http://www.plos.org

concise words; an example is the Gene Ontology [Gene Ontology Consortium, 2006]. One often encounters *ad hoc* names for one and the same instance, with different types of variations, which we can group as follows:

- orthographic: IFN-gamma, IFN-γ; IL-1, IL1
- morphological: Fas ligand, Fas ligands
- lexical: hepatitic leukaemia, liver leukemia
- structural: cancer in humans, human cancers
- acronyms/abbreviations: FADD; TFF3
- synonyms: neoplasm, tumor, cancer, carcinoma
- eponyms: Addison's disease, Addisonian syndrome

Many of these deviations underly no recognizable rules, even simple ones such as orthographic variations. All of the above types first add to the variations in which an object might occur. Alterations of existing names or creation of new names occur constantly over time once an object has been introduced (see Figure 2.1). Names altered by paragrammatical phenomena, often orthographic errors, still refer to objects and thus ideally need be solved to spot this object (see Table 2.1). The problem of homonymity arises in particular from names that resemble or (in case of multi word names) consist of common words from English or other languages: "Bride of sevenless", "boss", "Ken & Barbie", "Dickkopf"; some names even resemble frequent common English words, such as "the", "a", "of", "Who". Also, abbreviations for different compound nouns often are identical and thus hard to discern. Synonymity decreases the recall of information extraction, as many objects cannot be recognized in texts where they occur with other names. Homonymity, on the other hand, decreases the precision, as names cannot be resolved properly to an exactly identified entity.

To grasp the full extent of possible variations, one has to analyze names by their *term formation patterns*. This deals with discovering typical affixations, forms of compounding, composition of acronyms, and influences like foreign imports. Kageura [2002] discusses such aspects in greater detail.

fuction	59	funtion	31	functon	12	fnction	4
proten	25	potein	11	rotein	10	fanction	2
cerevisae	142	cervisiae	67	cerivisiae	15	cerevisie	3
posphatase	8	nuceotide	5	knase	1	fator	6
recptor	33	expresson	12	expressin	11	fctor	2
desease	116	theraputic	42	cencer	12	clasification	5
beast cancer	14	brestfeeding	4	beast-feeding	2		
polymerase chain rection			6	polymerase chain reation			6

Table 2.1.: Exemplary orthographic errors observed, with number of occurrences in MEDLINE (February 2006). The largest amount did not occur in new abstracts pending indexing, but rather in old ones.

2.1.2. Named Entity Identification

Named entity identification (NEI) tries to spot names and find the exact real world instance behind each name. Problems NEI deals with arise from homonymity and synonymity. A spotted name might refer to several instances of multiple classes (genes, diseases, medical terms, etc.) and thus, first the right class has to be identified. Next, there often are ambiguities even within a single class. Abbreviations in particular are a source of such ambiguities. The name "PSA" potentially refers to two different genes/proteins (protein S alpha, prostate specific antigen), a disease (psoriatic arthritis), an organization (Poultry Science Association), etc. Different instances (of the same or different classes) share one name and multiple aliases exist for one and the same instance, most times creating more ambiguities. NEI tries to map all variations of a name to a unique identifier for the referenced instance. Such identifiers would originate from representative databases that hold information on the given class. Examples are databases with protein–centered content, such as UniProt [Bairoch et al., 2005], or OMIM for hereditary disease–centered information [OMIM, 2000]. For collection-wide analyses (see introduction to this chapter) and for building association networks, this task constitutes a prerequisite to gain useful results.

Named entity identification overlaps to a great extent with word sense disambiguation, which we discuss in more detail in the following section. While WSD deals with finding the right entity class for a term, NEI tries to find the exact instance within a given class. Often, one method helps to solve the other. An initial identification that spots names and provides potential candidates helps further disambiguation, and disambiguation of words to assign general classes reduces the problem of spotting terms.

A common way to tackle named entity identification is to first spot all potential terms and to resolve them afterwards. Named entity recognition finds all candidate instances for a term. For this task, very high recall rates are necessary. NEI subsequently narrows down the set of candidates. Ultimately, this leads to the right instance and makes up for the precision lost during NER. Background information on each entity instance and the current context provide useful input for identification. For example, knowing the organism discussed in a text narrows down the set of potential candidates for a gene name by large. Other data available from knowledge repositories are annotations, functional descriptions, keyword lists, literature references, and so on. All of them contain valuable information that helps decide which exact instance was meant.

In addition, other strategies that exploit the immediate context of spotted names help improve performance. Most names appear multiple times in a text; if one occurrence is solved, this helps to solve other occurrences (other spelling variations, abbreviated forms, and so on), and even occurrences of other entities. Long names often are not as ambiguous as their abbreviations, and finding an acronym that abbreviates an already identified long form makes an immediate assignment possible (see also the remark in the next section).

2.1.3. Word Sense Disambiguation

Despite the productiveness of scientific language, ”novelties” in naming objects (real or conceptual) get less probable and appear in fewer variations. A name for a new object is very likely to have a different meaning in a different context – or a related, but not the same meaning in a related context. Thus, many words, especially in scientific texts, appear as homonyms across multiple areas and objects of research (in particular, homographs are problematic for text mining).

In the life sciences, objects often get named according to observations: for instance a phenotype or a symptom, which are influenced by the studied object. For example, a gene often is named for a phenotype observed, especially when a (natural or deliberate) mutation or knock-out of a gene results in a phenotype different from the wild type (”hairy”, ”red eyes”, ”lethal”); or as a function of *a* protein encoded by this gene (”alcohol dehydrogenase”, ”spindle pole body protein”). In text mining, it is not a trivial task to identify the true meaning of a word in a given text, for example whether the authors refer to a gene or a phenotype. There are gene names for many corresponding phenotypes, such as ”black” or ”minus”, which also are common English words, where most times the later is the appropriate meaning. Genes are also often named for diseases (for instance caused by a mutation of this gene), such as ”multiple sclerosis” or ”neurofibromatosis.” Other words very seldom appear as common words, like ”wingless”, but it is still hard to decide whether a gene, protein, mutation, or phenotype is the true meaning in the current text.

Word sense disambiguation (WSD) refers to the association of a sense (a meaning) to a word. It is related to entity mention normalization, which maps synonymous entities appearing within the same entity class to one unique instance (its ”sense”). In contrast, WSD potentially considers every sense a word has within and across multiple classes, or senses not bound to any class at all. Building a WSD system requires the identification of all possible senses for every word and the assignment of the correct sense to any occurrence. In the life sciences, WSD is necessary mainly to distinguish different entity classes (as shown in the previous paragraphs) and entities within the same class.

Different categories for senses result from the different entity classes under consideration, together with other techniques commonly discussed in biomedical publications, and common English (or other language) words. Table 2.2 lists the number of identical names some categories share. The names stem from dictionaries extracted from UniProt (gene/protein names), MeSH (cells, diseases, tissues), MedlinePlus (drugs), NCBI Taxonomy (species), and the 10,000 most frequent common English words. We find the largest overlap between gene/protein names and common English words (1423 homonyms). As the protein name dictionary is the largest (199,693 different names), these are 0.7% of all protein names. Even these 0.7% might introduce many errors, because some protein names occur quite frequently as common words (“of”, “the”). As mentioned above, these homonyms mostly result from names that are phenotypic or functional descriptions using common words. On the other hand, most names from the overlap of cells and tissues (46% of cell names also appear as tissues) actually have a very similar meaning, as cells form tissues. Overlaps within each class (figures not shown) are also very large. For proteins, there exist 74,321 entries with multiple IDs; many of those, however, are cases where an entry has no real second meaning, but there exists a second but identical instance (secondary IDs in UniProt). Multi-word names tend to decrease the ambiguity of the overall name, as every word adds more information pointing to a particular definition.

Class	Cell	Disease	Drug	Organism	Protein	Tissue	Common	Σ
Cell	–	5	2	0	0	123	0	130
Disease	5	–	5	18	4	3	3	37
Drug	2	5	–	9	67	1	15	99
Organism	0	18	9	–	175	4	38	239
Protein	0	4	67	175	–	2	55	300
Tissue	123	3	4	2	2	–	2	134
Common	0	3	15	38	55	2	–	108
#Entries	709	12,401	12,390	208,138	199,964	1356	10,000	

Table 2.2.: Number of names appearing in multiple entity classes and number of synonymous names per class. Figures include plural forms and disregard case sensitivity. *Common* refers to the 10,000 most frequent English words. *#Entries* gives the number of different names in each class. See text on page 13 for sources.

For highly specific domains of discourse (such as scientific/biomedical texts) it is sufficient to restrict the number of senses to only relevant ones, instead of trying to include all potential senses. A number of early studies confirm this observation [Gould, 1957, Panov, 1960]. The percentage of polysemous words in scientific texts was approximated to lie in the range of 30 to 43% [Harper, 1957b,a]. In a more recent study, Chen et al. [2005] found that 13.10% of all gene names have ambiguities with UMLS concepts and 1.77% with common English words (total amounts of UMLS terms and common English words considered were 1,728,455 and 74,550, respectively). This was not quite confirmed by our own study, where we found that only 0.15% of all protein names overlap with terms from five other entity classes and common English words (see Table 2.2). Overall, we found 1047 terms out of 444,958 as ambiguous (0.24%).

In essence, word sense disambiguation for restricted domains often functions as a step of false positive filtering. It is necessary to associate a name with only one of few possibilities of the few relevant senses or exclude it entirely. For example, when searching for proteins, it is sufficient to exclude any occurrence of "PCR" from the class protein, disregarding its potential meaning in other categories. Gale et al. [1992] observed that in most cases there is only "one sense per discourse", which shows that complexity is reduced on restricted sets. The meaning of a particular word seldom changes within a given text. Once an encountered word is disambiguated with a high confidence, its sense can be transferred to other, unsure occurrences.

Disambiguation often is simpler for abbreviations, because in most cases, the corresponding long forms are different and easy to distinguish. The aforementioned example, ″PCR″, can have (among others) the long forms ″polymerase chain reaction″ (a common technique in molecular biology) and ″protochlorophyllide reductase″ (a protein). As long forms are likely to appear at least once in a text (to introduce the abbreviation, or sometimes as a list of abbreviations at the end of the text), each abbreviation can be resolved to its particular long form if the mapping between both is certain; a typical pattern for such an occurrence is ″long form (abbreviation).″

On the other hand, sometimes short and long forms of different objects are identical, but have different meanings. For instance, the term ″Nf2″ can either refer to a gene, its product, a particular mutation in the gene, or to a disease resulting from this mutation. The same is true for the long form, in all cases ″Neurofibromatosis type 2″. In such cases it is hard to distinguish the different meanings from each other.

Gaudan et al. [2005] proposed a system to resolve abbreviations to their senses. They first try to detect the long form, for instance using the algorithm proposed by Schwartz and Hearst [2003]. Only in cases where no long form could be found, a machine learner tries to disambiguate the abbreviation according to a classification of the context. Thus, for every single meaning of an abbreviation known to the system, a model was trained on positive (text where the abbreviation had the right meaning) and negative (texts where the abbreviation had other meanings) contexts. Given an abbreviation and its context, the most confident decision from all relevant models decides on the meaning. This method was evaluated to achieve 98.9% precision at 98.2% recall.

Relevance filters (see introduction to this chapter) help to reduce the complexities of synonyms, homonyms, acronyms, and word senses. Especially for word sense disambiguation, filtering texts prior to named entity recognition reduces the number of different discourses, and thus meanings, of potentially ambiguous terms. While ″NER″ in a publication on text mining probably refers to ″named entity recognition″, the same term would abbreviate ″nucleotide excision repair″ (gene or pathway) in papers discussing oncology, or ″nitrogen efficiency ratio″ in texts on nutrition science.

2.1.4. Relation Mining

Relation mining deals with extracting facts in which multiple objects take part. For the remainder of this theses, we consider relations as interactions and associations between various types of entities. Relations in the biomedical domain come in manifold ways, among them

- interactions between proteins
- interactions between proteins and genes
- drugs related to (protein) targets or associated with diseases
- drugs related to side effects and contra-indications
- reactions at the molecular level involving chemicals and proteins/enzymes
- genes and mutations/SNPs associated with phenotypes
- genes/proteins assigned with functions, processes, locations
- observations (on above interactions) made in particular experiments
- experimental evidences for the above

In particular, we will focus on interactions between proteins that are expected or verified and described textually. Chapter 5 presents our own and related approaches to relation mining and discusses various techniques. Types of interactions of proteins with other substances can be summarized as follows:

- spatial interactions: proteins are part of the same complex
- proteins occur in the same metabolic pathway
- genes occur within the same operon structure
- dependent reactions
- proteins interact with proteins from other organisms
- proteins interact with other substances
- paralogous proteins: proteins with similar function that can replace each other
- proteins with similar properties and characteristics[7]
- immune complexes: antigen–antibody interactions

"Complexes" include different types, such as homodimeric and heterodimeric proteins, enzyme–inhibitor, and antibody–protein complexes [Jones and Thornton, 1996].

On top of developing systems to extract various kinds of relationships between biomedical objects, challenges are

- distinguishing between coincidental mentions and actual relations,
- finding the exact subject–object relationship,
- finding the type of interaction,
- identifying negations, hedges, and moods, as well as
- finding the method used to detect the interaction.

Regarding subject–object relationships, not all protein–protein interactions can be divided into active and passive components (agents and targets). They also include auto-interactions, where a protein regulates itself, or multiple copies of the protein form a complex. Negations, hedges, and moods add another level of complexity, where a written statement has to be re-evaluated. Consider the example "Our results do not confirm that protein A interacts with protein B." This basically means that someone else concluded from their experiments, A and B do interact, but potentially under different conditions. In addition to moods, the interaction detection method provides valuable insights into the reliability of an experimental finding. Large–scale experiments typically yield high false-positive rates; for an overview, see Shoemaker and Panchenko [2007]. Deane et al. [2002] and Schlicker et al. [2007] have proposed different techniques to asses the reliability of high-throughput data.

2.2. Machine Learning

In this section we introduce the machine learning techniques that are commonly used as components for building text mining systems. We explain the basic ideas and concepts and present those algorithms used later in this thesis in detail.

[7]Such as: pH induced, heat-shock induced, antigens, virulence factors

Machine learning builds on statistical analysis of data to infer general rules. Data in machine learning consists of sets of objects that represent the considered (real-world or imaginative) instances. In text mining, most times such objects are texts, sentences, phrases, or single words. The task of machine learning is either to extract rules from such representations that explain the structure of the underlying data, or to help distinguish objects from each other. Thus, the outcome is either the rules themselves or the result from applying the rules to data of unknown content so that they can be compared to known data. In text mining, an example would be the analysis of a set of words with known meaning (for instance, names of diseases). One outcome would be rules that explain the (typical) composition of such names (for instance, discovering common suffixes such as "-itis"), which bears interesting insights for linguists. Another outcome would be the application of such rules to decide whether a previously unseen name might actually belong to the same group of words (depicting diseases).

There are two main types of machine learning: inductive and deductive. Induction identifies similarities within a group of given objects, while deduction identifies an object by its resemblance to a set of other objects. Inductive learning is used to extract general rules from few specific examples. Deductive learning extracts specific rules inherent in a large population. Examples for either are

Induction	given two example names for diseases; both end with "-itis", so a new name that ends with "-itis" is very likely another disease name
Deduction	all disease names appear as nouns; if a new name is not a noun, it cannot refer to a disease.

Machine learning handles objects of either known or unknown content. Objects that are used for learning rules are called examples. The actual process of learning is often called the training phase. The outcome of training is a model that describes the examples seen during training. After training, thorough testing should take place in order to evaluate the learned model before finally applying it to new data.

Definition 2.3 (Example). *An example is an object that is used for training or testing.*

Often, the meaning of such examples are known previously.

Definition 2.4 (Sample). *A sample is a set of examples used for training or testing.*

Many machine learning techniques model objects as sets of characteristic properties and their respective values. Such properties are called features. The idea is that objects from each class have more features in common than objects from different classes. Such features should help to decide whether an object might belong to a specific class or not. They should thus capture the typical characteristics inherent in examples representing the different classes.

Definition 2.5 (Feature). *A feature is a typed property of an object that helps to describe this object.*

For each object, the value of a feature can be different. A feature value captures the importance of the given feature to describe the given object. The simplest form of feature values are binary features, stating if or not a certain feature applies to an object. Feature values can also reflect the relative importance of a feature, measuring the information a feature holds for distinguishing the given object from others. One such measure is the TF*IDF measure, see page 20.

A feature set represents all potential features encountered in a training sample. Single examples do not necessarily fulfill all of these features. A feature vector represents the set of features that describe a single example. When the feature vector is reduced to such features that apply to an example, it is called a sparse vector.

Definition 2.6 (Class)**.** *A class is a group of objects that belong together.*

Definition 2.7 (Label)**.** *A label assigns a class to an example. Examples from the same class all have the same label.*

A labeled example is an example of known class. A label can be either pre-assigned prior to training, or be the outcome of applying a model to an object.

Definition 2.8 (Cluster)**.** *A cluster is a set of objects that have been grouped together based on their similarity.*

Machine learning algorithms try to find the set of features and their values with the most discriminative power to distinguish all classes (defined by the objects they contain) from each other. They try to find rules that include features, values, and combinations of features. Such rules may be used to assign classes to new objects. As rules model characteristics of a sample and thereby the sample itself, we also refer to rule sets as *models*.

Definition 2.9 (Model)**.** *A models is a set of rules learned by a machine learning algorithm from examples, and that we can apply to new objects.*

There are multiple strategies in machine learning, depending on the set of objects at hand. *Supervised learning* learns a model from a set of labeled examples; such labels are pre-defined, thus the term *supervision*. and assigns labels to new objects, according to a this model. *Unsupervised learning* groups together similar objects out of a (larger) set into *clusters*. Semi-supervised learning is a mixture where only some examples in the overall sample are labeled. Techniques in supervised learning are classification and regression (decision trees, perceptron algorithms, support vector machines), generative models (expectation maximization), sequence learning (Markov modeling), and inductive transfer. Clustering is the most common form of unsupervised learning. Meta-learners (voting, bagging, boosting, random forest) constitute a set of techniques that enhance or combine other learning strategies. We describe classification, clustering, and sequence learning strategies in Sections 2.2.1 to 2.2.3.

Evaluation metrics

In this section we will show the metrics we use for evaluating system performance. We will first introduce some terminology, in continuation of the previous section. A *positive example* is an example that belongs to the current class of interest. A *negative example* thus is an examples does not belong to this class. An example predicted by the model to belong to the positive class is a *true positive* when it also belongs to the positive class in reality. An example predicted by the model to belong to the negative class is a *true negative* when it also belongs to the negative class in reality. We call an example *false positive* when the model predicted that it belongs to the positive class, but it does not in reality. An example predicted by the model to belong to the negative class is a false negative when it does belong to the positive class in reality. Table 2.3 summarizes these definitions in a confusion matrix.

		Real	
		positive	negative
Prediction	positive	TP	FP
	negative	FN	TN

Table 2.3.: Definition of true and false positive and negative predictions; this matrix if often called confusion matrix, when filled with actual figures depicting predictions of an experiment versus perfect outcome.

To measure the performance of a predictor (a model), two main indicators are used in information retrieval and extraction. The *precision* of a predictor measures the percentage of examples correctly assigned to the positive class (also known as *specificity* or *positive predictive value*, PPV, Equation 2.1). It indirectly states the number of errors made among all positive predictions. The *recall* of a predictor states the percentage of correctly recognized positive examples among all positive examples (also known as *sensitivity*, Equation 2.2). In general, most predictors can be tuned towards either higher precision or higher recall, but not both. To combine both measures, it is standard to use the *f-measure* in Equation 2.3 [van Rijsbergen, 1979]. In particular, we use the *F1-measure* (also known as *harmonic mean*), with $\beta = 1$, resulting in Equation 2.4. We plot the precision against recall in precision/recall curves, an example is shown in Figure 2.2; it also shows the F1-measure at several fixed percentages.

$$\text{precision} = \frac{TP}{TP + FP} \tag{2.1}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{2.2}$$

$$\text{f-measure} = \frac{(1 + \beta^2)\text{precision} \cdot \text{recall}}{\beta^2\text{precision} + \text{recall}} \tag{2.3}$$

$$\text{F1-measure} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.4}$$

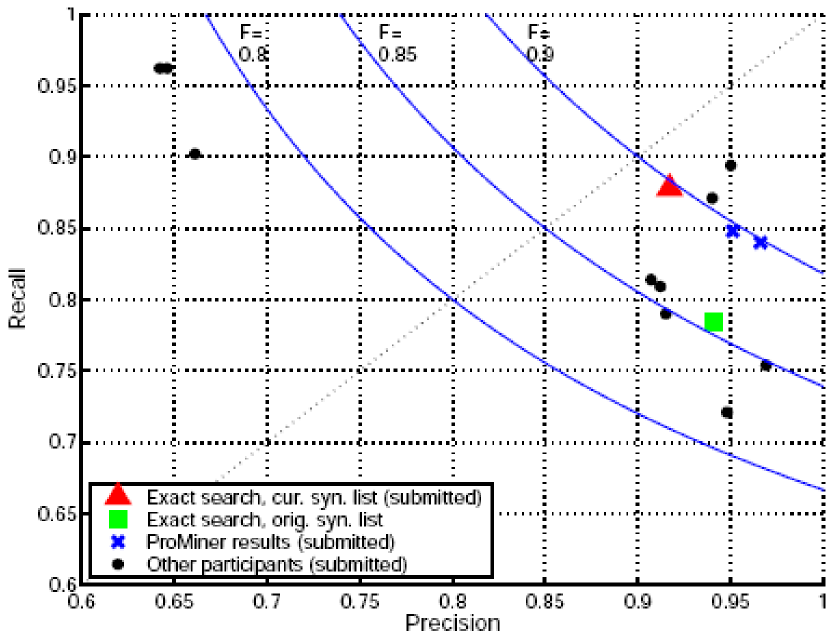


Figure 2.2.: Example for a precision/recall characteristic also showing the F1-measure (blue lines) for various fixed percentages (80, 85 and 90%). Source: Fundel et al. [2005].

Feature vectors and bag-of-words representation

The vector space model assumes that each example can be represented by a vector. The dimensions of a vector correspond to the features occurring in the example. Given the overall sample, identical features map to the same dimension of the feature space. Examples are thus distinguished by their exact respective coordinates, corresponding to feature values. Two examples can be discerned by comparing their feature vectors. We can visualize a vector space as a hyper-dimensional space were axes correspond to features and coordinates to feature values.

In text mining, examples correspond to documents. A sample thus is a collection of documents, also called a corpus. Features of documents often represent words or character n-grams such as suffixes encountered in a document collection. Sometimes, previously identified concepts that have been identified by grouping words of the same meaning together, are also used. Such concepts can be lemmata (to allow for morphological variations of a word: “protein”, “proteins”) or more complex concepts, such as Gene Ontology terms (to allow for lexical variations: “apoptosis”, “programmed cell death”). In both cases, each of the terms representing the same concept gets mapped to the same feature, instead of having multiple features for conceptually identical terms.

To measure the relevance of a single feature to an example in text mining, the idea of TF*IDF is most often used [Jones, 1972]. This takes into account the overall number of occurrences of a feature in a text (called term frequency, TF), normalized by the number of texts that also contain this feature (inverse document frequency, IDF). The more often a term appears in a text, the more relevant it is. On the other hand, if a term occurs very often throughout the text collection, this lowers its relevance for each single text. Given the term frequency $t_f(d)$ of term t in the current document d, we calculate TF*IDF as

$$\text{TF*IDF}(t,d) = \log(t_f(d)+1) \cdot \log \frac{|D|}{|\{d \in D \mid t_f(d) > 0\}|}, \tag{2.5}$$

where D is the document collection and d a document in D.

The information on the original ordering of terms in a text is completely lost when reducing terms to frequencies. This approach is called a *bag-of-words* representation, throwing all terms together [Mitchell, 1997]. To retain ordering, special sequence features that represent consecutive terms rather than single ones can be extracted. This strategy, however, is seldom applied.

2.2.1. Classification

We distinguish between two main groups of classification methods, where the problem is transformed into a problem of *estimation* or *partitioning*. Estimation determines a function that calculates probabilities for each class for a given example and its feature vector. For instance, Bayesian approaches belong to this group of methods. Partitioning subdivides a sample represented in a vector space into parts of similarity and assigning a class label to each part. Note the distinction from clustering, which we introduce in Section 2.2.2. An example for methods that solve the partitioning problem are k-nearest neighbor approaches.

The equivalent of partitioning is to determine mappings from feature spaces to class labels. The training examples appear in the form (x_i, c_i), where x_i represents a feature vector from the vector space $\mathcal{X}$, and c_i is the class label of x_i from $\mathcal{C}$. In binary classification, this label is either -1 or 1. The classifier then has to learn a mapping f from these training examples,

$$f(q) \rightarrow \mathcal{C}, \tag{2.6}$$

that assigns any representation $x \in \mathcal{X}$ for a new object q to its class c. Support vector machines try to solve exactly this problem, and we will introduce them in the following.

Support Vector Machines

The approach we use in our systems for classification tasks are *support vector machines* (SVMs; Vapnik, 1995). SVMs have been shown to outperform other techniques, both in terms of time performance and prediction quality (for example, see Joachims [1998]). The main idea behind SVMs is to find a hyperplane that separates all objects that belong to one class from all objects of the other class (in the following, we consider only binary SVMs as opposed to multi-class SVMs). Such a hyperplane is learned from a set of training examples. To classify new objects, an SVM checks on which side of the hyperplane a new object resides.

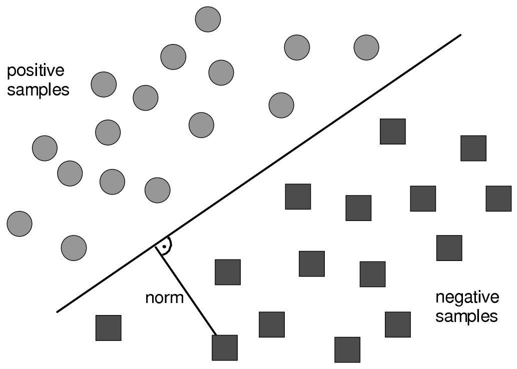


Figure 2.3.: Linear SVM with a hyperplane separating two classes of objects in a 2D space.

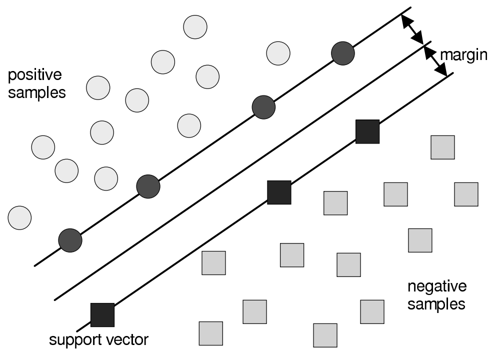


Figure 2.4.: Support vector machine with margin hyperplanes. Dark examples: support vectors.

From a training sample $\{(x_i, c_i)\}$ an SVM learns a hyperplane that best separates the two classes c_{-1} and c_1: The closest examples of both classes should be as far away from the hyperplane as possible. This introduces a *margin* around the hyperplane, which we then call a *(maximum-) margin hyperplane*. The examples that are closest to the hyperplane, and which ultimately define the hyperplane itself (see later), are called *support vectors*. A linear hyperplane is of the form

$$\langle w, x \rangle - b = 0, \tag{2.7}$$

with the normal vector w and the input vector x, while the two margins are

$$\langle w, x \rangle = -1 \qquad \text{and} \tag{2.8}$$

$$\langle w, x \rangle = 1, \qquad \text{respectively.} \tag{2.9}$$

To ensure that no examples lie between these two margins, we need any example x_i to satisfy either

$$\langle w, x_i \rangle - b \leq -1 \qquad \text{or} \tag{2.10}$$

$$\langle w, x_i \rangle - b \geq 1. \tag{2.11}$$

Using the class labels $c_i \in \{-1, +1\}$, this transforms into

$$\forall i : \; |c_i(\langle w, x_i \rangle - b)| \geq 1\,. \tag{2.12}$$

To guarantee maximum-margin hyperplanes, we need to maximize the distance between both margins. The margin m is

$$m = \frac{1}{||w||^2}, \tag{2.13}$$

and we thus minimize

$$\min_{w,b} \frac{1}{2}||w||^2 \tag{2.14}$$

subject to Equation 2.12, called *primal optimization problem*. This minimization results in a very large quadratic programming optimization problem, depending on $|\mathcal{X}|$. In text mining, both the amount of documents and features are usually large. Some algorithms have been proposed, *sequential minimal optimization* (SMO) being the most common [Platt, 1998a,b]. SMO breaks the large optimization problem into a series problems, each as small as possible. The worst case complexity for SMO is quadratic.

To classify a new query instance q, we need a decision function f, dependent on w and b:

$$f_{w,b}(q) = \text{sgn}(\langle w, q \rangle + b) \tag{2.15}$$

With Lagrange multipliers α_i, we can transform the minimization function into a dual form that depends only on these multipliers:

$$\min_{\alpha} \frac{1}{2} \sum_{\forall i} \sum_{\forall j} \alpha_i \alpha_j c_i c_j \langle x_i, x_j \rangle \sum_{\forall i} \alpha_i, \tag{2.16}$$

subject to the two constraints

$$\forall i : \ \alpha_i \geq 0 \tag{2.17}$$

and

$$\sum_{\forall i} \alpha_i c_i = 0. \tag{2.18}$$

To calculate w and b from the training data x_i, we derive them from the Lagrange multipliers using

$$w = \sum_{\forall i} \alpha_i c_i x_i \tag{2.19}$$

and

$$b = \langle w, x_j \rangle - c_j \text{ for some } \alpha_j > 0, \tag{2.20}$$

respectively.

Extending the decision function 2.15 leads to dot products between a query instance q and all training examples x_i, which yield the class of q:

$$q \mapsto f(q) = \text{sgn}(\sum_{\forall i} \alpha_i c_i \langle q, x_i \rangle + b) \tag{2.21}$$

It is not always possible to find a linear separation for a sample set. There are two methods to circumvent this problem. One is to allow for some examples to lie between the two margins, the other to transform the problem into a higher-dimensional space where a linear solution exists.

Slack variables

A modification of the minimization problem defined in 2.14 was proposed by Cortes and Vapnik [1995]. This modification allows for some examples to lie within the two margins instead beyond (or on) them. Each such error, called *margin failure*, gets penalized using *slack variables* ξ_i and a penalty parameter C in an error term. Altering 2.14 and 2.12 accordingly, we now have to minimize

$$\min_{w,b} \frac{1}{2}||w||^2 + C \sum_{\forall i} \xi_i \tag{2.22}$$

subject to

$$c_i(\langle w, x_i \rangle - b) \geq 1 - \xi_i \ \forall i. \tag{2.23}$$

Kernel trick

Another method called *kernel trick* transforms the linear algorithm into a non-linear one; it generalizes SVMs to non-linear classifiers, even further than slack variables [Aizerman et al., 1964]. This method maps the initial observations into a non-linear space of higher dimensionality. The linear algorithm in this higher-dimensional space is the equivalent to a non-linear algorithm in the initial space. To do so, we need a *kernel* that replaces the dot product between two vectors in the original algorithm (see Equation 2.21). Such a kernel functions as a similarity function, or, conversely, a distance measure. comparing two vectors, and returns a real value reflecting this similarity:

$$\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbf{R}, \text{ where } (x, y) \mapsto \kappa(x, y). \tag{2.24}$$

Note that the original dot product also is a distance measure. When normalized, it measures the cosine angle between two objects. A function Φ performs the actual mapping of vectors, so that a kernel κ for two vectors x and y can be written as

$$\kappa(x, y) = \Phi(x)\Phi(y). \tag{2.25}$$

A number of kernels have been proposed, ranging from general to user-defined kernels. While general kernels are applicable to arbitrary problems, user-defined kernels incorporate domain knowledge to best suit specific applications. There are two main design principles for kernels: model and syntax driven. Model driven kernel directly incorporate domain knowledge (protein sequences, speech recognition, etc.), syntax driven kernels are fitted for the structure of a problem (strings, graphs, etc.). Among the more general kernels are

- linear kernel: $\kappa(x, y) = x \cdot y$
- polynomial kernel: $\kappa(x, y) = (\gamma x \cdot y + r)^d$
- radial basis function: $\kappa(x, y) = exp(-\gamma ||x - y||^2), \gamma > 0$
- gaussian radial basis function: $\kappa(x, y) = exp(-\frac{||x-y||^2}{2\sigma^2})$
- sigmoid kernel: $\kappa(x, y) = tanh(k\langle x, y \rangle + c)$, for some $k > 0$ and $c < 0$

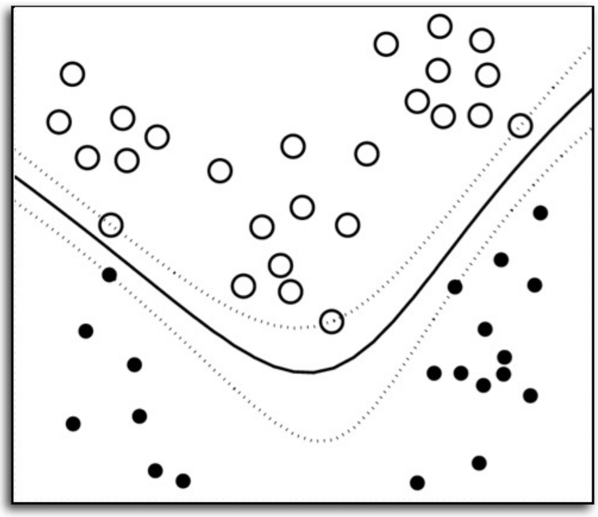


Figure 2.5.: Hyperplane resulting from an RBF kernel that separates two classes (circles and dots). Dotted lines correspond to the margins, with some support vectors on them. Adopted from Schölkopf and Smola [2003].

with kernel parameters γ, d, r. Figure 2.5 shows a margin hyperplane classifier for a radial basis function kernel (RBF kernel).

For specific applications in computational biology, Leslie et al. proposed combinatorial string kernels that measure protein or DNA sequence similarity [Leslie et al., 2002, Leslie and Kuang, 2004]. Vishwanathan and Smola [2003] presented fast string kernels for matching discrete objects, such as strings and trees, in linear time. Solutions to predict class memberships of enzymes and non-enzymes use graph kernels using sequence, structure, and chemical information on proteins [Borgwardt et al., 2005].

Support vector machines are well suited for our classification tasks for two main reasons. Typically, text categorization uses large feature spaces as inputs. Not only can every single token represent a feature, but also n-grams, suffixes and prefixes, and other additional information. Features can easily add up to hundreds of thousands. SVMs are able to handle these large feature spaces. While the input space may have a very high dimensionality, the vector for each example is sparse. For any particular example, only a few features are different from zero. In practice, only non-zero features are stored and used for the computation of models. SVMs (using SMO in particular) can be adopted quite well for sparse inputs [Platt, 1998b]. Algorithms like SVMs have been shown, both theoretically and empirically, to suit problems with sparse input very well [Kivinen et al., 1995].

2.2.2. Clustering

In clustering there are no predefined labels to assign to an examples. Instead, clustering groups together the examples that are most similar to each other into clusters. Similarity is defined using a similarity (or distance) measure (see next paragraph). To determine the similarity of clusters (rather than single objects) in order to combine multiple clusters into one, there exist several linkage rules (see later in this section). Only after the clustering we see which examples belong to the same cluster. In principle, there are two ways to cluster a data set: agglomerative or divisive. Agglomerative clustering starts at the point where there is a cluster for each single example. Iteratively, the most similar examples (and later, clusters), are joined together. Divisive clustering works in the opposite way: given a data set where all examples belong to one cluster, this strategy iteratively divides the cluster into sub-clusters.

Clustering does not depend on predefined labels, but we can perform a clustering on a data set that includes at least some labeled examples. By doing so, we can automatically assign a label to all the clusters that contain at least one such pre-labeled example. On the other hand, using some pre-labeled examples allows us to quantify the quality of a clustering, either per cluster or over all clusters. This is done by testing how many examples of the same label appear in the same cluster, and how many different labels occur in a cluster. Clustering with labeled examples also helps to identify multi-modal clusters, where clusters can have multiple (spatially distant) parts (see bi-clusters in Figure 2.7).

Similarity functions

We define similarity measures for examples represented as feature vectors, as introduced at the beginning of Section 2.2. Every example x has several features, each noted as x_i. A similarity function assigns two vectors from a vector space $\mathcal{X}$ a similarity value:

$$\text{sim: } \mathcal{X} \times \mathcal{X} \rightarrow \mathbf{R} \tag{2.26}$$

Several similarity functions have been proposed. The cosine similarity (2.27) of two examples x and y is the most used similarity function. It is sensitive to the relative importance of each feature [Chen et al., 2004]. The Euclidean distance (2.28) is the normed distance of two vectors in a vector space. The Dice coefficient (2.30) expresses similarity as the proportion of features two objects share to all features of both objects. In Jaccard coefficients (2.31), similarity is the proportion of features that two objects share, normalized with the features they do not share [van Rijsbergen, 1979]. Other similarity/distance measures are the Manhattan, Chebychev, and Power distances, and the percentage disagreement.

$$\text{Cosine coefficient} = \frac{\sum_{\forall i} x_i \cdot y_i}{\sqrt{\sum_{\forall i} x_i^2 \times \sum_{\forall i} y_i^2}} \tag{2.27}$$

$$\text{Euclidean distance} = \sqrt{\sum_{\forall i} (x_i - y_i)^2} \tag{2.28}$$

$$\tag{2.29}$$

$$\text{Dice coefficient} = \frac{2\sum\limits_{\forall i} x_i \cdot y_i}{\sum\limits_{\forall i} x_i + \sum\limits_{\forall i} y_i} \tag{2.30}$$

$$\text{Jaccard coefficient} = \frac{\sum\limits_{\forall i} x_i \cdot y_i}{\sum\limits_{\forall i} x_i + \sum\limits_{\forall i} y_i - \sum\limits_{\forall i} x_i \cdot y_i} \tag{2.31}$$

Lewis et al. [2006] evaluated the performance of different similarity coefficients on TREC data (cosine, Jaccard, dice). They showed that for text data, represented by stemmed words and excluding stop words, the cosine coefficient with a variant of TF*IDF (see earlier in this chapter) performs best among these. They also found that sentence alignment yields a slightly better precision in their data (29% as compared to 27% for cosine similarity), but is not applicable for real-time computations as required in their overall system. Later in this thesis, we will use clustering to group together similar sentences. We propose a measure for sentence similarity that is based on sequence alignment, using a normalized alignment score (see Section 4.3.1 on page 81).

Linkage rules

Linkage rules (also called amalgamation rules) define how to determine the similarity of two clusters, in order to decide whether to combine them into one ("link" them) if they are sufficiently close to each other. The cluster distance can be determined using either

- single linkage, nearest neighbor: distance of the nearest objects from the different clusters determine cluster distance
- complete linkage, furthest neighbor: distance of the furthest objects
- unweighted pair-group centroid: distance of cluster centroids
- weighted pair-group centroid: distance of cluster centroids, weighted with cluster sizes
- unweighted pair-group average: average distance of all pairwise distances of any two objects in the different clusters
- weighted pair-group average: average distance of all pairwise distances, weighted with cluster sizes

Best known representatives of pair-group distances are UPGMC, WPGMC, UPGMA, and WPGMA, introduced as *(un)weighted pair-group methods using the centroid average* or *arithmetic averages*, respectively [Sneath and Sokal, 1973].

Most clustering algorithms fall into either of three fundamental concepts: tree clustering, block clustering, or k-means clustering, which we fill discuss in the following.

Tree clustering, joining

Tree clustering recursively clusters all object into larger and larger clusters. This algorithm starts with each object representing one single cluster, and potentially ends with

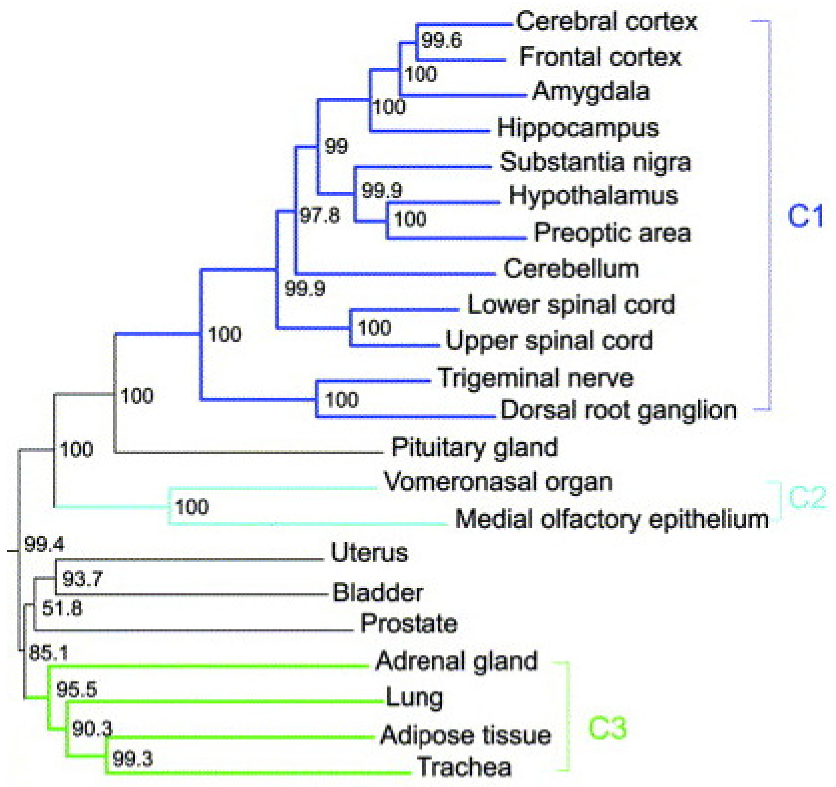


Figure 2.6.: Illustration for a cluster analysis of expression vectors of genes in various tissues. The tree was constructed using Euclidean distances between vectors with the neighbor-joining method. Taken from Yanai et al. [2006], Figure 1, in parts. ©2006 Elsevier Ltd. All rights reserved.

one cluster containing all objects when no prior termination takes place (see Figure 2.6). At the beginning, the two objects most similar to each other get combined into one cluster. This similarity is defined using pairwise similarity measures between objects. Thresholds on the outcome of the similarity function are used to determine whether two objects should be regarded as similar or not. Iteratively lowering this threshold leads to more and more objects clustered into larger and larger clusters, until only one cluster remains.

Block clustering, two-way joining, biclustering

In many cases, a set of objects is described using a set of variables rather than only a single variable. Block clustering takes place on both sets simultaneously [Hartigan, 1972, Mirkin, 1996]. In principle, when the data is arranged as a two-dimensional matrix of objects (rows) and variables (columns), permutations of both rows and columns rearrange the data. The goal is to derive a correspondence structure on the combined sets. Cheng and Church [2000] introduced biclustering of gene expression data. Microarray data often shows set of genes and their expression levels given different conditions. A row in the matrix would correspond to a gene, a column to a condition, and a spot to the gene expression level. Figure 2.7 shows an example for a two-way clustering of a gene/tissue data set.

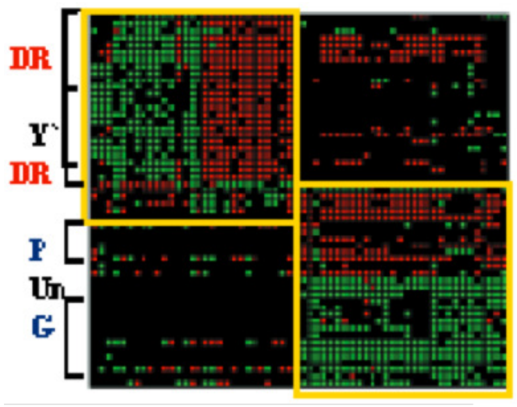


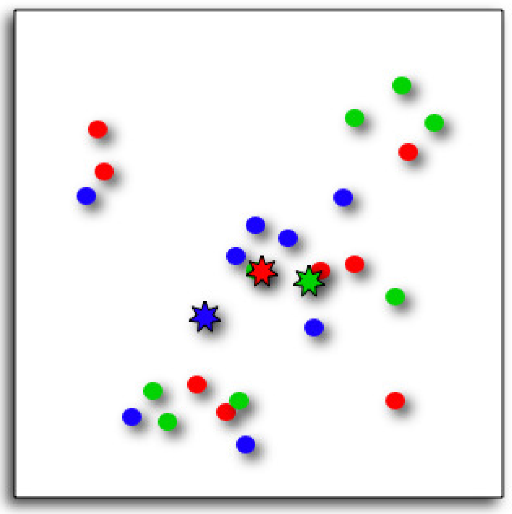
Figure 2.7.: Example for a block clustering of genes and conditions/tissues. The figure shows expression patterns in two related clusters (biclusters; marked by yellow frames). Image taken from Tanay et al. [2002].

k-Means clustering

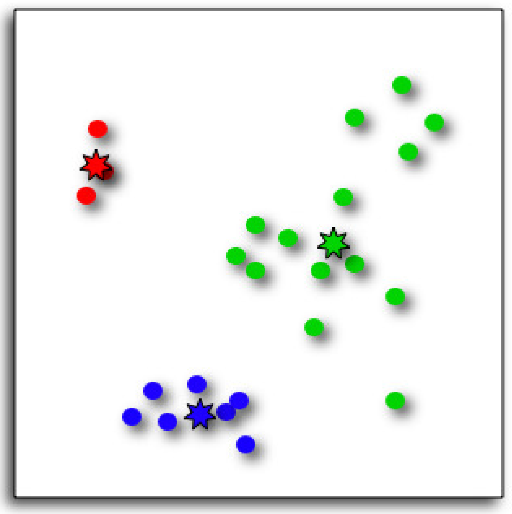
In contrast to other methods, k-means clustering assumes a predefined number k of clusters. k-Means assign each object to one of the k clusters in a way that all clusters are sufficiently distinct from each other. Most implementations start with a random assignment of objects to clusters. Based on this initial clustering, k-Means recursively moves objects from one cluster to another in order to maximize the dissimilarity between clusters and maximize the quality of each cluster. The means of the final clusters on each feature can be seen as the outcome of a k-means clustering. These data help find representative values for each cluster, and also to assess the degree of dissimilarity between clusters. Figure 2.8 illustrates the iterative reassignment of examples to clusters and recalculation of cluster means. An extension of the k-means clustering uses expectation maximization (EM; see later in this section). It calculates probabilities for cluster memberships for each object, instead of fixed assignments.

2.2.3. Sequence Learning

As we have seen in Section 2.2.1, classification assigns a label to objects given its set of features. Sequence models, on the other hand, assign a sequence of labels to a corresponding sequence of objects, all at the same time. The output is the most probable sequence of labels given the sequence of objects and their respective descriptions. This also means that, in contrast to a bag–of–words view of data, the sequence of objects plays an important role. Each sub–sequence (and its probability) influences the



(a) Random assignment of examples to three clusters with calculated initial cluster means.



(b) Eventually visible clusters and means evolve after several iterations of reassigning clusters and recalculating the means.

Figure 2.8.: Example for a k-means clustering. The initial assignment (a) is obviously very indistinct. After only six steps, solution (b) emerges. Circles: examples; colors: clusters; stars: cluster means.

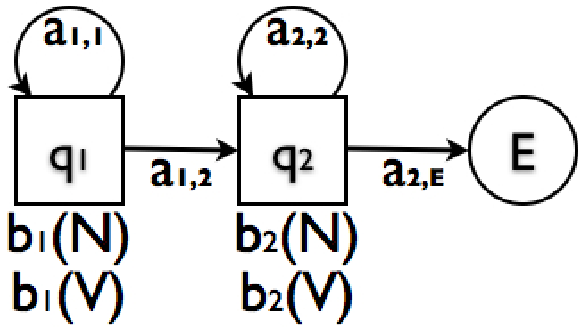


Figure 2.9.: Simple hidden Markov model with two states (q1, q2) and an end state (E). State q_1 emits labels 'N' and 'V' with probabilities b_1(N) and b_1(V). Afterwards, the model stays in q_1 with probability $a_{1,1}$, or goes into state q_2 with probability $a_{1,2}$. Start probabilities for q_1/q_2 omitted.

probability of the overall sequence. Hidden Markov models (HMMs), which had their first applications in speech recognition: the probability of a sequence of 'basic' sounds determined the sequence of words spoken. HMMs also have applications in natural language processing, for instance as part–of–speech taggers, which predict sequences of POS tags.

Hidden Markov models

Hidden Markov models (HMMs) are a form of sequence models that describe a probability distribution over all potential sequences. This distribution always sums up to one, so probabilities of sequences have a high influence on each other. This fundamental constraint of probabilistic modeling was described by Jaynes [2003]. Hidden Markov models are defined using two sequences: a sequence of outputs (called emissions corresponding to the observed objects), and a sequence of states, which is not observable ("hidden"). The sequence of states models the combined probability for the sequence of labels at hand. Beginning at a start state, we proceed through the set of states until an end state is reached. Along the sequence of states, each state potentially emits any label, so that ultimately a sequence of labels is generated. Each transition from one state to another and each emission have certain probabilities assigned to them, so that in the end, an overall probability of a sequence of states and emissions can be determined. Because every state can emit any label, it is not possible to tell in which state the model was when observing a particular label.

Figure 2.9 illustrates a simple example for a hidden Markov model. Here, the model has two emitting states, q_1 and q_2, and an end state, E. The model first stays in q_1 with a certain transition probability, $a_{1,1}$, at every step emitting a label 'N' or 'V' with a certain probability, b_1. With probability $a_{1,2}$ the model transits into state q_2, where it behaves similar as in q_1, emitting labels 'N' and 'V' with probabilities b_2. With probability $a_{2,E}$, it goes from q_2 into the end state. Given a state sequence "1, 2, 2", the model emits three labels, one observed sequence of labels could be "N, V, N". The same state sequence could produce other label sequences, as the label sequence "N, V, N" could be generated by other state sequences. All potential variations, however, potentially have different overall probabilities.

There are three key problems that need to be solved in computing HMMs. In the *scoring problem*, the probability of the observed sequence given a model is computed. The *decoding problem* involves the computation of a state sequence that best explains an observed sequence given a model (this is also called *alignment problem*). The *estimation problem* deals with adjusting the parameters in a model to maximize the probability for the observed sequence. We present algorithms that solve either problem in Appendix B.1.

2.3. Natural Language Processing

Natural language processing (NLP; NL understanding, computational linguistics) describes the process (and science) of analyzing texts in natural language for syntactic and semantic structures contained therein. Syntactic analyses determine the structure of a sentences, the dependencies and references between subjects and objects. Semantic analyses determine the meaning of a sentence, of sub-phrases, and of individual terms. Lexical, syntactic, and semantic knowledge of a language are involved in these analyses [Hahn and Wermter, 2006]. Natural language processing is not restricted to written language (as presented in this thesis) but includes the processing of spoken language as well. Here, additional information on phonology and speech becomes necessary. NLP includes from various research fields: linguistics, computer science, psychology, cognitive science, statistics, and many more [Mitchell, 1997]. For sentence analysis, the three levels of NLP refer to different subtasks:

- lexical level: tokenization, morphological analysis, lexical characterization,
- syntactic level: part-of-speech tagging, chunking, parsing, and
- semantic level: lexical semantic interpretation, composite meaning analysis, disambiguation.

After an introduction to sentence splitting, we will discuss the lexical and syntactic level in the following sections, as they provide basic input for our subsequent text mining tasks. Analysis on the semantic level is discussed in Chapters 3 to 5: for named entity recognition on Chapter 3, and for mining relationships in Chapters 4 and 5.

2.3.1. Sentence Splitting

One of the first tasks in processing natural language is to recognize sentence boundaries to split a text into single sentences. Complexity of natural language necessitates this step in order to alleviate – sometimes facilitate – further processing. It breaks down texts into smaller components of connected information, which can be sufficiently handled by NLP techniques; an example are parsers to understand the dependency structure in a sentence, as we will discuss later in this section. Each sentence boundary gives important information about the probability of the current sequence of POS tags or dependencies in a dependency graph. In addition, each start of a new sentence defines the start of a new POS tag sequence, chunk, parse structure, and so on, resetting the processing despite possible uncertainties in the previous sentence.

Sentence splitting is not as trivial a task as it might appear. A naïve approach is to split a text at every occurrence of either '.', ';', '!', or '?' (followed by a white space and an uppercase letter); problems arise from this strategy, though. Most sentences end with a full stop, this symbol is also used to mark abbreviations and proper names, occurs as a decimal point, or may actually be a part of a word. Such problems arise especially in scientific texts. The following examples present some of the difficulties (critical sites underlined):

- abbreviations: "..different from the spectra of P. aeruginosa and E. coli. We find that.." – no split after "P." and "E.";
  "..anti-TNF-alpha antibodies and cyclosporin A. In the recent past, .." – split;
  "..(c) 2005 S. Karger AG, Basel.."
- parts of terms: "..and discovered that 5′-3′; exonuclease activity of RecJ.."
- within brackets: "..CD95 system (FasL, CD95L; FasR, CD95R) and.."
- within citations: "..referred to here as the "hairpin model." More recently,.."
- paragrammatical phenomena: "..and history of past attempts.Conclusions. Both psychosocial and clinical factors.." – split despite a lacking white space

In most cases, it is better to follow a conservative strategy and prefer to not split a sentence when in doubt. Subsequent processing takes part on the sentence level, and thus might loose information whenever a single sentence is split into two. For instance, a wrong split affects approaches that try to match patterns against sentences. On the other hand, when sentence splitting misses a boundary and combines two sentences into one, this is still an acceptable input for many subsequent tasks. Further NLP processing, in particular part–of–speech (POS) tagging and chunking, helps to recover some of such errors by begin able to retain the correct sequence of tags/chunks. Note that most POS models are trained on single sentences, so that errors prevail, especially for infrequent start (POS) sequences in a sentence. Sentence parsing relies on preceding POS tagging and sometimes chunking, and depends on the correctness of the input from all three subtasks.

Most approaches for sentence splitting are based on hand-crafted rules, sometimes with lexica for typical abbreviations ("Dr.", "et al."). Candidate split sites and their respective surrounding text is analyzed further to identify sentence boundaries. A number of techniques based on machine learning have been proposed. Reynar and Ratnaparkhi [1997] presented a maximum entropy model that learned to split at '.', '!', and '?', respectively. MxTerminator achieved an error rate of 1.2% on the Wall Street Journal corpus (WSJ). The caveat was that such complex modeling for a "minor" subtask increased computation time substantially for the overall task of mining full texts or text collections. Mikheev [1998] proposed a similar method, but used simplified parameter estimation for candidate features. This ultimately produced very compact models (in terms of hundreds instead of tens of thousands of features). On the same corpus of 27,294 sentences from WSJ, they achieved an accuracy of 99.2477% (error <0.8%), which was the best performance reported so far. Schmid [2000] proposed an unsupervised learning for period disambiguation, which achieved an accuracy of 99.56% on WSJ and 99.79% on the Brown corpus. An approach proposed by Tomanek et al. [2007] uses using conditional random fields for sentence splitting and tokenization. On a combination of the GENIA and PennBioIE corpora, they achieve an accuracy of 99.8% for sentence splitting and 96.5% for tokenization (see next section). Tomanek et al. also argue that missing a split is preferable to introducing a false split; optimizing their tool to minimize false positives resulted in an FP rate of 30%.

2.3.2. Tokenization

After splitting a text into its sentences, information extraction requires the identification of the smallest units of information, that is, tokens. They are separated by white spaces from each other and, in most cases, resemble words. Inter-punctuation and other symbols (brackets, slashes) are proper tokens that often are connected to another token without an intermediate white space; for instance, the full stop at the end of a sentence is single token; alternative names separated by a slash: "TBA1/Rap1" form three tokens altogether. Hyphens can either connect two tokens (and thus are tokens themselves) or the hyphenated word can be seen as a single token. Multiple tokens might form a multi–word name or term. Most times, a multi–word term syntactically is a composite noun.

Definition 2.10 (Token)**.** *A token is the smallest unit of information that is formed by consecutive characters, figures, or inter-punctuation.*

Definition 2.11 (Word)**.** *A word is a token that bears not only syntactic information (like inter-punctuation), but also semantic information.*

Definition 2.12 (Term)**.** *A term consists of a single or multiple, consecutive tokens that all together refer to a unit of information.*

2.3.3. Part-of-Speech Tagging

Tagging assigns labels to terms and words. Part-of-speech tagging, in particular, recognizes the syntactic behavior of a word in a given sentence. It identifies the grammatical form of the word given the word and its surrounding context. Such grammatical forms are verbs, nouns, adjectives, adverbs, prepositions, pronouns, conjunctions, and interjections. These eight basic categories of part-of-speech in the English language sub-divide into more categories. A noun has singular and plural as well as possessive forms, a verb appears in different tenses and aspects. For computer linguistics, it is typical to distinguish up to 160 part-of-speeches (again, for English). There are several proposed standards to annotate words with part-of-speech tags. In the following chapters, we use the Brown tags [Greene and Rubin, 1981]; Table 2.4 shows the most common examples.

Tag	Meaning	Tag	Meaning
NN	noun, singular	NNS	noun, plural
NP	proper noun	NPS	proper noun, plural
VB	verb, base form	VBD	verb, past tense
VBG	verb, gerund; present participle	VBN	verb, past participle
VBZ	verb, present, 3rd person singular	MD	modal auxiliary
DT	determiner	WDT	which-determiner
RB	adverb	RBR	adverb, comparative
RBS	adverb, superlative	*	not
JJ	adjective	JJR	adjective, comparative
JJS	adjective, superlative	FW	foreign word
IN	preposition	TO	infinitive marker "to"

Table 2.4.: Examples for Brown part-of-speech tags to annotate words.

Named entities are another form of tags, and every named entity also appears in a grammatical form. Proteins and genes, for instance, most times occur as (proper) nouns in a text. Organism names sometimes appear as adjectives that specify a gene ("human protein kinase α"). While some words can only have one part-of-speech tag ("strongly" always is an adjective), others depend on their context ("form" can be a noun or a verb).

Various methods have been proposed for part-of-speech tagging. The Trigrams'n'Tags (TnT) tagger is a statistical tagger based on the Viterbi algorithm for second order Markov models [Brants, 2000]. Whenever it encounters unknown words, which did not occur in the training data, is guesses the POS from abstractions of words based on common suffixes. For instance, an unknown word ending with "–ing" almost certainly is either a verb (gerund) or a noun (singular). On the Penn Treebank corpus, the TnT tagger achieves an accuracy of 96.7%. The probabilistic TreeTagger that uses decision trees was described by Schmid [1994]. It uses a second order Markov modeling of trigrams, but tries to overcome data sparseness (where not enough training data is available to include every potential trigram) with a decision tree on likely tag sequences. On data from the Penn Treebank corpus, TreeTagger yields 96.36% accuracy. The Brill tagger Brill [1992] applies transformation–based error–driven learning. Predictions of an initial model are compared with the known solution and rules are extracted that transform initial output to better fit the training data. It also uses default lexica and contextual rules learned from unspecific data, namely the Wall Street Journal and the Brown corpus [Marcus et al., 1993, Francis, 1964], with no special markup for biology-related entities. On the WSJ corpus, the last version of the Brill tagger achieves an accuracy of 96.6%. A POS tagger designed for handling biomedical text is the HMM-based MedPost [Smith et al., 2004]. It includes a lexicon of the 10,000 most frequently occurring words in Medline (accounting for 92.7% of the tokens in Medline as of 2004). MedPost was trained on a set of 5700 manually tagged sentences (155,980 tokens), which where picked randomly from various subsets of Medline. On a test set of 1000 manually tagged sentences from PubMed abstracts, MedPost achieved an accuracy of 96.9% using the Penn-Treebank POS tagset, where Brill's tagger achieves an accuracy of 86.7%. Using a native tag set of 60 POS tags, MedPost achieves 97.4% accuracy.

All these POS taggers except for MedPost have been trained and evaluated on 'traditional' texts such as the Wall Street Journal or the Brown corpus, but not on domain specific texts. Their performances cannot be transferred directly to the life science domain, where slightly different sub–languages are used. Smith et al. [2004] found that almost 60% of the word types in Medline did not occur in either the Brown or Associated Press (AP) corpora (based on the 92.7% most-frequent tokens only). Paragrammatical phenomena occur more often, especially in texts written by non-native speakers. Long multi–word terms, for instance protein names, hinder a successful tagging, since they often contain adjectives and verbs ("TNF–related apoptosis inducing ligand-like (TRAIL–LIKE)"). Such names should rather be treated as long noun phrases (by pre– or post–processing stages) in order to retain the correct sentence structure. Parsers that use such identified chunks as input are then more robust towards complex sentences; they can analyze noun phrases and dependencies therein separate from building the overall parse.

2.3.4. Stemming

Stemming refers to the reduction of a word to root or stem [Baeza-Yates and Ribeiro-Neto, 1999]. For example, the words "localize", "localized", and "localization" all have the same stem, namely "local". A lemma, which is often used synonymously for stem, is the morphological stem of a word. In the example, the lemma would be "localize", hinting not only on the stem, but also on a process ("–iz"). A lemma thus contains more information to distinguish the exact sense of a word.

Porter [1980] proposed the algorithm that is most commonly used for stemming. Its original form was for stemming English words, but adaptations to many other languages exist. Porter's heuristic rule set reduces a word to its stem by iteratively altering the affix. An affix is the suffix added to a word stem to form a valid derivative of the word, called morpheme; in contrast to word stems, affixes cannot stand alone. of a word, most times shortening it. For examples, plural forms get reduced to singular forms by removing either "ss" or "s", or reducing "sses" or "ies" to "ss" and "i", respectively. Porters' rules also take into account some elongations that lead to more general forms of the affix; for example, words ending with "at" or "iz" get an extra "e" to form "ate" or "ize". Ultimately, the rules leave only the words' stem. Though there are many rules involved, the algorithm reduces every English word in only eight steps.

Stemming helps to reduce the dimensionality of a learning problem by mapping many different words to only one stem. This helps to make commonalities of these words (and thus sentences, paragraphs, and texts) more apparent to the machine learning algorithm. On the other hand, especially in specific domains like biology and medicine, stemming might introduce errors by removing valuable information. Stemming reduces both "organ" and "organization", which represent entirely different concepts, to "organ", while the respective lemmata, "organ" and "organize", differ. This might be the reason that common stemming algorithms, which do not use knowledge on a specific sub–language, often do not help to improve the accuracy of text classification, as shown by Nenadic et al. [2003].

2.3.5. Sentence Chunking

Chunking refers to splitting sentences into smaller subunits, single units of information, also called syntactic groups. It identifies noun and verb chunks. Chunking first tries to identify potential chunk heads. These are words that identify the actual object (in case of noun chunks) or predicate (in case of verbs). Other words in the chunk specify (modify or qualify) the object/predicate. For example, in the sentence

"We always used adjacent normal kidney tissue",

the noun and verb chunks (NP and VP) are:

"We [VP always used] [NP adjacent normal kidney tissue]".

In "adjacent normal kidney tissue", "tissue" is the chunk head, specified as "kidney tissue", and so on. Chunk heads, which can be resolved using part-of-speech tags and positions in the chunk, then determine the types of objects/predicates involved. Chunking is a helpful preprocessing step before parsing a sentence completely (see the following sections). In the preceding example, the information content can be broken down to

"We used tissue",

which is much easier to parse then the initial full sentence. Chen and Chen [1993] showed that a probabilistic approach can reach a 'correct chunk rate' of around 98% (with 94% of all sentences chunked correctly) on the Susanne corpus (a part of the Brown corpus). There are currently no results for assessment of chunking algorithms on biomedical corpora available. They used a tagger to pre–annotate each sentence with POS tags. The model is then trained on word bi–grams. Given a sequence of words, the system searches for the maximum probability over each candidate solution for the parse tree.

2.3.6. Shallow Parsing

Shallow parsing is another preliminary stage (as chunking) to full sentence parsing. In shallow parsing, the structure of a sentence is not resolved to the level of all single elements. A shallow parse often deals only with identifying the structure using nominal, verbal, and adjectival phrases. An example system for shallow parsing is Sundance [Riloff and Phillips, 2004], which is also used in several relation mining systems (see Chapter 5). This system builds on preprocessing steps as mentioned previously (sentence splitting, segmentation, part-of-speech tagging) and other modules (for instance, morphological analyses and dictionary lookups). The Collins parser achieves an f-measure of 89.5% on the WSJ corpus [Collins, 2000]. This system re-ranks the most likely initial parses of a sentence with additional information by boosting.

2.3.7. Full Sentence Parsing

Full sentence parsing refers to analyzing a sentence for its syntactical structure. In addition, the morphology of each word is resolved. A sentence structure can be seen as a syntax tree (parse tree) that shows the phrase structures and how they relate to (dominate) each other. In general, parsing finds the subject–predicate–object triples in a sentence that present the overall information. Figure 2.10 shows an example for a parse tree. In this example, a verbal phrase ("detected") connects a subject ("we") and an object ("gangliosides"), where the object is further specified in the remainder of the sentence. We will discuss approaches for relation mining that make use of (partial or full) dependency parsing in Chapter 5. Though sentences in scientific publications often are more complex than general sentences, Rinaldi et al. [2005, 2006] conclude that full parses can be as successfully applied to the biomedical domain when chunking, POS tagging, and NER are solved properly. Miyao et al. [2008] compare different parsers and their influence on the task of protein-protein interaction extraction, finding similar performances ranging from 55.9 to 58.3% F1 for eight different parsers on this task.

We distinguish between different types of sentence parsing: dependency parsing, phrase structure parsing, and deep parsing. 1) Dependency parsing generates a structure of the sentence where tokens are the constituents and edges may exist between tokens to depict relationships. Such relationships would, for instance, connect a predicate to its object with a (directed) 'subject' edge. Relationships thus represent an approximation of the semantics of a sentence (connecting subject to predicate to object, for instance). An example for a dependency parser is the Maximum Spanning Tree parser (MST; McDonald and Pereira, 2006). Another representative is the LinkGrammar parser [Sleator and Temperley, 1993], which is based on the original link grammar theory of English. 2) Phrase structure parsing results in a constituent tree, most often in Penn-Treebank notation (PTB), which ranges from the tokens in the bottom level over word categories, chunk information, to the sentence level. Charniak and Johnson's Reranking parser and the Stanford parser are often used implementations of this framework [Charniak and Johnson, 2005, Klein and Manning, 2003]. 3) Deep parsing computes theory-specific syntactic and semantic structures as well as predicate-argument-structures. Compared to dependency parses, a deep parse contains more in-depth and also long-distance relations between constituents. A recent implementation of deep parsers is Enju, which is also available in a biomedical version, retrained on the GENIA corpus [Miyao and Tsujii, 2008]. Examples for parses generated with LinkGrammar and Enju can be found in Figures 2.11 and 2.12 for comparison.

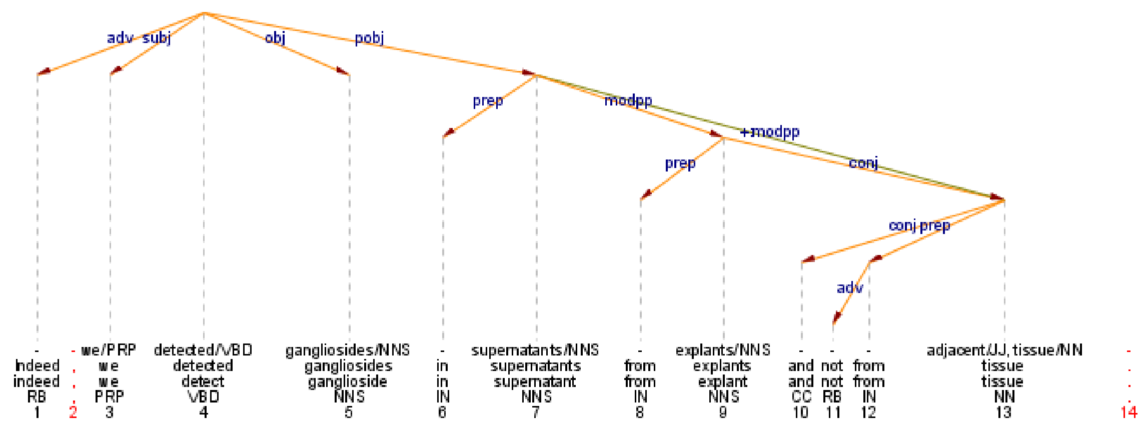


Figure 2.10.: Example for dependency relations in a sentence. Generated from PubMed citation 99423690 using OntoGene [Rinaldi et al., 2006].

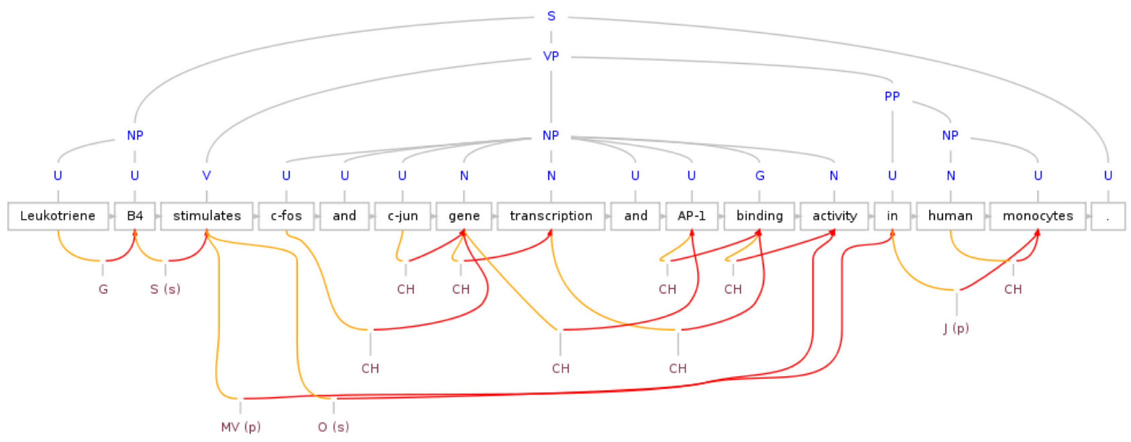


Figure 2.11.: Example of a LinkGrammar parse (sentence from PubMed 1313226). Output shows the constituent tree on top, and the linkages on the bottom.

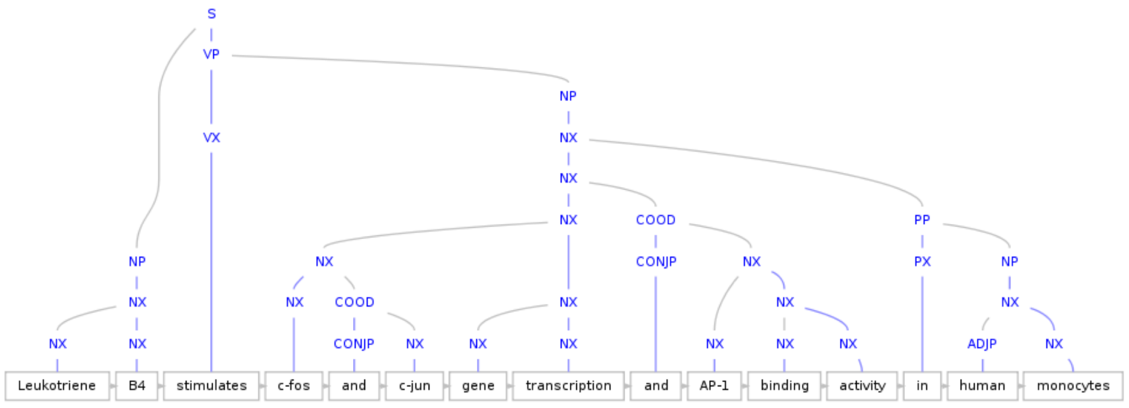


Figure 2.12.: Example of a deep parse using the Enju parser (sentence from PubMed 1313226).

3. Named Entity Recognition

Solving the problem of recognizing named entities in texts is a prerequisite for information extraction systems. In this chapter, we will present our approaches to solve this problem for gene and protein names. First, we present different strategies to handle this task, and then discuss the two approaches we use for NER in detail. We will present the methods and data needed for both approaches. The focus is on recognition and identification of gene and protein names, as this has been one of the most studied applications in bioinformatics in the last years. The chapter finishes with a discussion of related work and conclusions drawn from our evaluations.

Most approaches for NER fall into either of the following categories:

- token– or term–based classification,
- sequence modeling,
- dictionary–based approaches,
- rule–based approaches, or
- hybrid approaches.

Token-based classification takes into account information on the token and its context (for instance, the surrounding text) to decide whether it belongs to a given class or not. In most cases, this information are surface clues, which refer to the composition of a token or a whole term (that is, the sequence of letter and symbols). Such clues could be affixes of a token — for instance, "-ase" or "-tor" are typical for protein names like in "kinase", "factor", or "receptor." A large variety of classification techniques (for example, support vector machines) can be used to learn models based on such information.

Sequence models (for instance, hidden Markov models) calculate the most probable sequence of class labels (that is, part–of–speech tags and/or entity classes) given a sequence of tokens. While token-based classification decides on the label of each token separately, sequence modeling assigns the respective labels to all tokens of a sequence at once by deciding on the most probable sequence. Again, the input mostly consists of properties referring to the characteristics of tokens.

Both approaches rely on learning their respective models from a training set. Such a set has to contain (manually) labeled examples for all types of entities that are of interest. A training set consists of tokens, whole names, or sentences (depending on the type of input the approach takes into account). Most strategies not only need positive examples for a given class, but also counter-examples. Only then the learner can extract characteristics that point towards or away from a decision, both having strong influence on the final outcome.

The third category, dictionary-based approaches, consists of systems that match text against pre-compiled dictionaries (sometimes referred to as gazetteers). Mainly, there are two different strategies for such an approach. The first is to collect an exhaustive list of names referring to entities of the given class. To also find variations of these names, inexact string matching identifies all occurrences of these names in arbitrary text. The other strategy is to extend an initial list of names by adding variations for each name, which are then searched exactly.

Rule-based approaches take into account either typical features of entities of a given type, as well as contextual information, similar to token-based classification. They use mainly hand-crafted rules that refer to the composition of a name and its surrounding context. For instance, the word sequence "the *X* protein" directly hints at "*X*" referring to a protein, regardless of the exact token(s) for "*X*".

Hybrid approaches try to incorporate different strategies into one system. They try to profit from the respective beneficiary properties of either technique. Another approach to integrate multiple systems is to use different techniques in parallel and finally combine their independently achieved results, for instance, with voting schemes.

Processing pipelines enrich the annotation for each word and sentence step-by-step, for example with part-of-speech tags or sentence structure. Such information provide important additional input for subsequent steps. As we will show in Section 3.1, post-processing stages help to refine candidate terms once they have been identified.

The next two sections present two different approaches in more detail, namely token–based classification and a dictionary–based approach. Finally, Section 3.4 discusses related work and achievements in the field of biomedical named entity recognition.

3.1. Token-based Named Entity Recognition

Token-based classification transforms the problem of recognizing entity names into one decision for each single token in a sentence. A classifier decides whether a token is (part of) a name belonging to the given entity class (in our case, proteins and genes), or not. A decision function, trained on a set of examples for either class, determines class membership for new tokens.

Our system for token-based named entity recognition consists of two stages. The first tries to predict for each single token in a sentence if it is part of a protein name or not. The second stage combines multiple tokens into one compound term and expands these names to previously misclassified trailing or leading tokens, whenever necessary.

3.1.1. Classification task

Building a classification–based system starts with finding a suitable representation of each token. The idea is to describe each token not only using the token itself, but also helpful other features (see Section 2.2 for details on features). In this section we present our ideas for features and feature classes. We then describe how to eliminate useless features and features having a bad influence on the prediction performance.

Feature generation

There are various ways to represent a token and its context in a sentence using a set of features. We use classes for features that refer to the surface of a token or its occurrence in a dictionary. The surface of a token describes its overall syntactic composition, for example the sequence of letters and their capitalization, or existence of special symbols and digits. This is also referred to as the *canonical form* of a token. As capital letters, symbols, and digits typically do not appear in common English words, these bear important hints on distinguishing protein names from common words. Examples for surface features are

- A surface feature represented by “Xxxxd” captures names like “Smad2”, “Esat6”, and “Brca2”: words that start with a capital letter ("X"), followed by three lower case letters ("xxx") and end with a digit ("d"). This feature is unlikely to fit “common” English words but most times hint at some "special" word instead.
- A surface feature “xxxxx” captures words such as “which”, “refer”, and “token” and most often represents “common” English words that have many (in this case, five) consecutive lower case letters. However, such tokens might still be part of longer, more complex compound names.

Surface features can be derived from nomenclature guidelines (see Section 2.1.1) or word lists (dictionaries). Nomenclature guidelines explicitly describe the composition of proper names and their distinction. However, not all authors adhere to these standards, and texts might have been published long before a standard existed. Furthermore, it is hard to derive such surface features for long, descriptive names ("Tumor necrosis factor alpha"), as they use many common words. Statistical analyses of dictionaries help. This enables the detection of terms or letter sequences common even in compound names. Examples are typical endings of a compound name, like "1" or "alpha"; words at or close to the ending, like "receptor" or "factor"; words initiating a name, like "coupled" or "tumor". Such words might refer to other entity classes themselves, like tissues, diseases, or tissues ("Liver X receptor alpha", "Tumor protein D52", "heat shock protein 70").

The immediate context of a token could bear further clues, such as trailing words directly pointing to the type of entity ("PLOD1 gene", "PLOD1 allele", "CD8+ T cell"). Such tokens can also be part of the actual name itself ("uncoupling protein 1"). Table 3.1 shows the keywords we use as indicators for protein names, extracted manually from examples in the BioCreative 1 training data.

The different classes of features used in our system can be described as follows:

- **token**: all tokens from the training set (positive and negative examples) are represented as single features;
- **unseen token**: new texts will contain tokens that were not encountered in the training set; randomly, tokens from the training set are treated as 'unseen', to not over-fit the model towards known tokens;
- **character n-grams of tokens**: substrings of length n occurring in the training data ($n = 1..4$);

adaptor	factor	protease	nucleus
antibody	gene	protein	surface
antigen	inhibitor	receptor	receptor
disease	kinase	strand	regulator
domain	ligand	cell	upregulator
effector	pathogen	compartment	

Table 3.1.: Keywords that are strong indicators for a gene or protein name when they appear before/after an unknown token. These are the most frequent keywords observed in the BioCreAtIvE corpus, see Chapter A.

- **dictionary**: tokens and compound words matching a dictionary of protein names; we take these tokens from the training corpus, and enrich the set by including names from HUGO, SGD, MGD, and Flybase; the dictionary ignores capitalization, white spaces, hyphens, and slashes, and replaces digits with wild cards;
- **surface clues**: these features are based on the composition of tokens, representing matches against various patterns consisting of letters, digits, and special characters;
- **keyword distance**: distance in words of a token to the nearest keyword.

Tokens and n-grams are weighted using TF*IDF weights, where the set of tokens represents the document collection. All other features are binary features (0 or 1). For the full listing of features and feature classes, see Table 3.2.

Recursive feature selection

Introducing feature classes is a random procedure, especially where surface clues are concerned. We tried to find the best set of feature classes manually in a trial–and–error approach. This is unsatisfactory for a couple of reasons. First, it was infeasible to test every possible set of feature classes, but any combination could yield the best overall results. Second, we were only able to test the influence of complete feature classes, rather then single feature within this classes. It might be that only some members of some feature class ultimately would lead to the best overall performance. The last reason was that for very large feature sets (including all but some classes), time performance got worse, which made the task of testing all combinations complicated.

With recursive feature elimination (RFE) we iteratively removed the worst performing single features. We started with a model containing all features from all feature classes, as described in Table 3.2. From that model, we selected and removed those features that had the lowest weight assigned to them by the SVM. Afterwards, we retrained the model and calculated the overall performance of the reduced feature set. RFE should ultimately remove all features (and entire feature classes) that decreased the overall performance. In addition, RFE should lead to a more concise and informative model. Feature elimination and retraining should lead to a small set of relevant

Feature	Example	Short name	Impact	
Token *	`Sro7`	Token	=54%	*- baseline -*
Unseen token *		UToken		
n-grams of token * ($n=1..4$)		1G, 2G, ..	+15%	P+, R++
($n=1..3$)			+14%	
Previous & next tokens ([1,1])		P/NToken	-5%	P+, R-
[2,2]-window			-6%	
n-grams of tokens in window		2PG/2NG/..		
Prefixes, suffixes ($length=1..4$)		1P,2P,3P,1S..	±0%	
Stop word (10,000 words)	`the, or`	Stop	-5%	P+, R-
(1000)			-1%	P+, R-
(100)			-0.5%	P+, R-
Initial upper case *	`Msp`	initCap	+0.5%	P=, R+
All chars are upper case *	`MMTV`	allCaps	+0.5%	P=, R+
Upper case letters *	`InlC, GUS`	Upper		
Upper case (skip first) *	`MsPRP2`	Upper2		
Single capital	`A`	singleCap	+0.5%	P+, R+
Two capitals	`RalGDS`	twoCaps	+0.5%	P+, R+
Capital, then mixed letters ∘	`IgM`	capMix		
Lower case, then mixed ∘	`kDa`	lowMix	+1%	P-, R+
Special symbols *	`ICAM-1`	special	±0%	P-, R+
Characters and numbers *	`p50`	CharNum		
Numbers *	`p50,HSF1`	Number		
Letters, digits, letters ∘	`H2kd`	ldl	±0%	
Digit, dot, digit ∘	`5.78`	ddd	-0.1%	P-, R-
Greek letter ∘	`alpha`	greek	+0.5%	P+, R-
Roman numeral ∘	`II, xii`	roman	±0%	R+, R-
Number followed by '%' ∘	`75.0 %`	percentage	-0.1%	P-, R-
DNA, RNA sequences ∘	`ACCGT`	DNA, RNA	-0.1%	P-, R-
Length of longest consonant chain *	`Sro7`→ 2	LCC	-2%	P-, R-
Keyword distance *		keyDist	-20%	P+, R-
Dictionary *		Dict	-3%	P-, R-
Prev./next token is PROTEIN		PTG, NTG	-18%	Prev.: P+, R-

Table 3.2.: Feature classes used in our experiments with influence on performance (4th column). Impact of each single feature class compared to the baseline (only tokens). Figures include post-processing. The fifth column shows how precision and recall are affected. *: classes used in the initial BioCreAtIvE submission that together comprise the best performing feature set, ∘: classes implemented afterwards.

features that may be beneficial for further investigations. Another advantage is an easier interpretation of the computed solution, due to the reduced dimensionality.

Recursive feature elimination was described, for example, by Guyon et al. [2002]. Recall the introduction to support vector machines in Section 2.2.1 and the decision function 2.15. Disregarding the additive threshold, b, the inner product denotes a linear combination of all features of x, where each feature's weight is its corresponding component in w. Since we allow only positive feature values, the sign of each component of w determines if the corresponding feature is an indicator for the decision. A positive sign points towards, a negative sign away from a protein name. The absolute values correspond to the features' impact on the inner product and thus on the decision.

In each iteration of RFE, we start with training and evaluating the SVM classifier. We then remove all features having a zero component in w. Afterwards, we eliminate either a fixed percentage or a fixed number of the remaining features. These features are chosen according to their lowest weights in w. Retraining the SVM on the reduced feature set starts a new iteration of the RFE procedure.

3.1.2. Post-processing to find multi-word terms

We noticed that many protein names are multi-word terms: in the BioCreative 1 Task 1A training set, 4781 out of 8876 gene names consist of two or more tokens (53.8%). Our initial classification very often misses full names, but instead marks only some (often vital) parts of the name. Most frequently, nouns and nondescript adjectives not appearing in the training examples are missing from the full names. Thus, using a token–based classification necessitates merging and perhaps expanding tokens to full names. In our system, consecutive tokens classified as protein names get merged into one multi-word term directly. To include parts of a protein name that were not initially recognized as such, we use a set of heuristic rules based on part-of-speech information. Whenever we find an initial token classified as (part of) a protein name, we apply rules involving the immediate context (surrounding tokens) of this token. These rules either extend sequences of tokens to full phrases, or they remove markup from some isolated (single) tokens. Table 3.3 list all these rules, found by manual inspection of predictions by the SVM on the training and development data. Note that these rules fit the annotations observed in the BioCreAtIvE corpus, task 1A (see Chapter A). Thus, they might be highly specific and not applicable to other corpora without adaptation.

All rules use the tokens themselves, their respective labels generated by the SVM, and their part-of-speech tags. For the latter, we use POS tags as predicted by the Brill tagger [Brill, 1992]. In general, the rules works as follows. Whenever a noun phrase —defined as a consecutive sequence of nouns, potentially following an adjective— contains a protein name (one or more tokens), then the whole noun phrase is typically a gene name. Exceptions include a number of special nouns and adjectives that never or rarely occur as part of a protein name. Such nouns and adjectives were collected from the training corpus using collocations. Every word appearing more frequently as a part of a protein name than preceding a protein name (and not being part of it) gets included in the expanded name (and vice versa). This leads to two exclusion lists for nouns – we expand a name, if the noun is none of 372 or 222 particular terms, respectively. We decided to use a negative list, as we found much more nouns included in protein names than preceding or trailing protein names. We do the opposite for adjectives, including an unmarked adjective in the expanded name if it appears in a list of 778 particular terms. This list was also collected from the training examples. The rules consider formations of protein names around slashes and parentheses, too.

As an additional false positive filter, our system removes some particular words that rarely represent protein names when occurring by themselves (see Table 3.4). These often get predicted as protein names because they are used in the same way protein names are, or their immediate context matches that of protein names. Some additional rules, shown in Table 3.5, remove false markup when the surrounding tokens are predicted as protein names.

Initial sequence		Expanded sequence	Limitation
`NN* PROTEIN`	⇒	`PROTEIN PROTEIN`	all but 222
`PROTEIN NN*`	⇒	`PROTEIN PROTEIN`	all but 372
`PROTEIN DT NN*`	⇒	`PROTEIN PROTEIN PROTEIN`	all but 372
`NN* DT PROTEIN`	⇒	`PROTEIN PROTEIN PROTEIN`	all but 222
`JJ PROTEIN`	⇒	`PROTEIN PROTEIN`	only 778
`PROTEIN JJ`	⇒	`PROTEIN PROTEIN`	only 778
`PROTEIN CD`	⇒	`PROTEIN PROTEIN`	
`PROTEIN . CD`	⇒	`PROTEIN PROTEIN PROTEIN`	
`DT .+ PROTEIN`	⇒	`DT PROTEIN PROTEIN`	
`PROTEIN / PROTEIN`	⇒	`PROTEIN PROTEIN PROTEIN`	
`/ PROTEIN`	⇒	`PROTEIN PROTEIN`	
`PROTEIN /`	⇒	`PROTEIN PROTEIN`	
`( PROTEIN , NN )`	⇒	`( PROTEIN , PROTEIN )`	
`( NN , PROTEIN )`	⇒	`( PROTEIN , PROTEIN )`	
`NN* / PROTEIN`	⇒	`PROTEIN PROTEIN PROTEIN`	
`PROTEIN / NN*`	⇒	`PROTEIN PROTEIN PROTEIN`	

Table 3.3.: Post-expansion rules to extend protein names based on a single hit in a multi-word term. `PROTEIN`: tags that mark single tokens as part of a protein name. We exclude 372/222 particular nouns from the expansion for the first four rules; we include only 778 adjectives for rules five and six; all six such rules recognize noun phrases. NN*: nouns, proper nouns, plurals; JJ: adjective; CD: cardinal digit; DT: determiner; '/' refers to a slash appearing in the text; '.': a single character; '.+': one ore more arbitrary characters, no white space.

activator, domain, factor, family, gene, heavy, kinase, locus, mRNA, protease, protein, receptor, reductase, ribosome, SNP, subunit, tRNA; alpha, beta, gamma, ...

Table 3.4.: Wrongly predicted words that never occur as a protein name by themselves; observed in the BioCreAtIvE data.

PROTEIN agonist !PROTEIN	⇒	PROTEIN NN !PROTEIN
PROTEIN antagonist !PROTEIN	⇒	PROTEIN NN !PROTEIN
PROTEIN binding !PROTEIN	⇒	PROTEIN VBG !PROTEIN
PROTEIN DNA binding	⇒	PROTEIN NN VBG
PROTEIN nucleotide sequence	⇒	PROTEIN NN NN
!PROTEIN intron PROTEIN	⇒	!PROTEIN NN PROTEIN
!PROTEIN exon PROTEIN	⇒	!PROTEIN NN PROTEIN
cell types PROTEIN	⇒	NN NNS PROTEIN
!PROTEIN [number] !PROTEIN	⇒	!PROTEIN CD !PROTEIN

Table 3.5.: Additional rules for filtering false positives that contain specific words. Whenever "agonist" is predicted as a protein, as is the previous token but not the following one, then we reset "agonist" to noun. !PROTEIN: any tag other than PROTEIN; VBG: verb, gerund.

3.1.3. Evaluation

Table 3.2 shows the influence on performance of all single feature classes compared to a baseline. For the baseline, we use only the tokens as features, which achieves a performance of 54% after post-processing. For all other results in Table 3.2, we use the token feature and the respective other feature class only. Table 3.6 shows the performance of different (manually picked) combinations of feature classes.

We found that many false positives were due to predicting very generic terms, such as "transcription factor" or "rat gene". Indeed, some of these names depicted protein families rather then members. Those also counted as false hits, because the information gained was insufficient. In addition, some names depicting families might get specified further, sometimes immediately following the family name ("Transcription factor AP-2 gamma"). We counted such errors as boundary-errors (see later in this section).

Feature	Impact	
Tokens + letter surface clues	+2%	P+, R-
Tokens + 1,2,3-grams + greek + roman + letter surface clues	+14%	P+, R++
Tokens + 1,2,3-grams + keyDist + Dict + LCC + special + combi + allCaps + initCap *	+16%	P+, R++
Tokens + 1,2,3,4-grams + keyDist + Dict + LCC + special + combi + allCaps + initCap + lowMix ○	+18%	P+, R++

Table 3.6.: Influence of combinations of feature classes on the prediction performance (including post-processing). "Letter surface clues" (bottom part) refers to the following features from Table 3.2: {special, allCaps, initCap, capMix, lowMix, ldl, ddd}.

Recursive feature elimination

Two setups should reveal the worst performing features and also the most informative and best ones. Overall, we had more than 150,000 features, depending on the set of feature classes. For the first setup, we iteratively removed the 10% of all features that had the lowest weights. Figure 3.1 shows the corresponding curve of removed features and performance. Overall, the performance dropped constantly after each iteration. Removing more than 95% of the initial features still left a model that performed only slightly less then the initial one (-2.3%). With a reduced removal rate of 1000 features instead of 10% (plus all features with weight zero), the maximum F1-measure of 71.1% could be achieved with about 23,000 remaining features, see Figure 3.1. After we removed all but 150 features, we looked deeper into their composition and meaning, as they should be the most discriminating ones. In general, the 150 features can be grouped as follows. About 70 features referred to different three- and four-grams. Around 70 other feature depicted single tokens, and ten surface clue features. Table 3.7 contains examples for either group.

Examples for the 70 remaining three- and four-grams are common suffixes such as "ing" and "ese", but also highly specific ones, such as "76-k", "Stai", and "GTTA", occurring with only a low frequency. An evidence that points towards the effectiveness of RFE was that very few three- and four-grams overlapped each other, which would share potentially redundant correlations.

To find patterns in the most important n-grams (regarding a positive decision), we stopped a new RFE run as soon as only 500 n-grams remained. We then counted their occurrences in the whole corpus, finding that most of them occur more often in protein names than outside or vice versa. For example, the 4-gram "nRNP" occurs 20 times (denoting an abbreviation for "nuclear ribonucleoprotein") in words like "hnRNP" or "snRNP-specific", all of which are (parts of) protein names. Behind most highly ranked token features we can find biological meanings, as in the previous example. On the other hand, the 4-gram "oped" (for example in "developed") has a negative weight and occurs exclusively outside protein names (55 times).

We can split the 70 single token features mainly into three groups: common stop words receiving a high negative weight (such as "the" or "are"), common important keywords receiving a high positive weight ("kinase", "protein"), and abbreviations ("Gnt", "ZII") with either positive or negative weights.

Post-processing for name expansion

Post-processing improves the performance of our system by 12% precision and 10% recall. It corrects about 300 previously incomplete gene names (more than 20% of the former false negative predictions). It additionally removes 120 false positives (10%) by explicitly deleting generic single word names (like "protein"). By manual inspection we found the post-processing never makes a mistake in the latter cases. In Figure 3.2, we present the impact of the post-expansion on recall and precision. The best setting achieved an overall precision of 71.4% at 72.8% recall (f-measure: 72.1%).

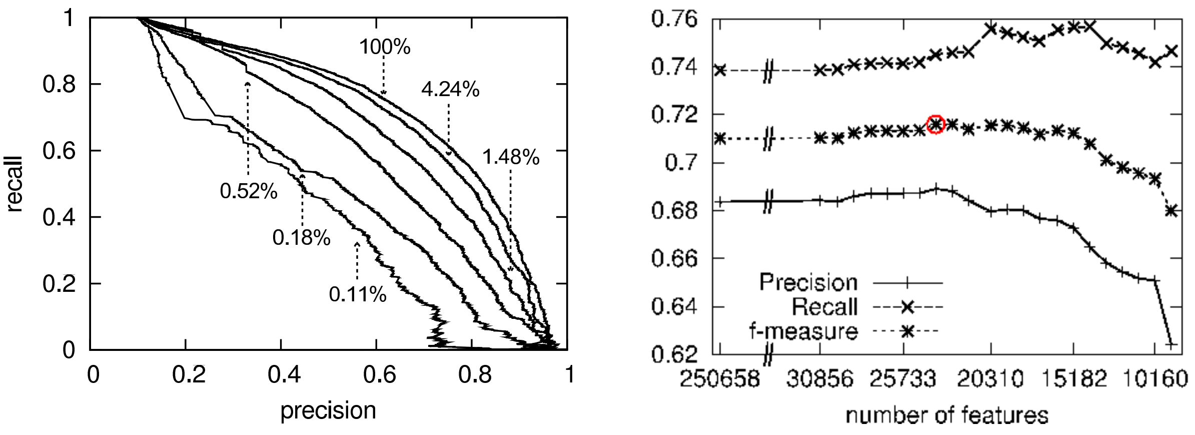


Figure 3.1.: Correlation between feature set size and performance. *Left:* percentages of remaining features and their corresponding precision/recall curve. *Right:* numbers of remaining features with corresponding performance rates. Evaluation without post–processing.

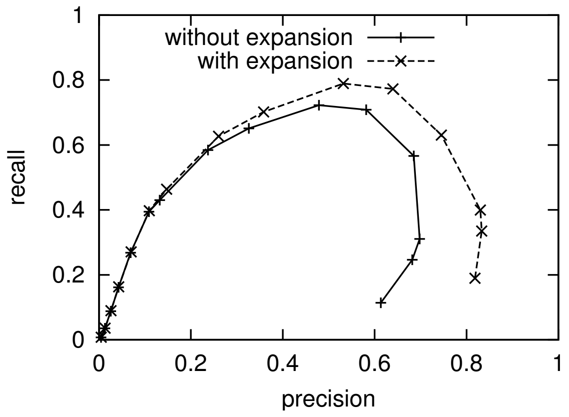


Figure 3.2.: Recall and precision with and without post-expansion. We use the full feature set (100% in Figure 3.1) for this evaluation. We obtain the different spots by parallel shifts of the hyperplane.

Feature	Class	Weight	Feature	Class	Weight
	Dict	1.497386	AACC	4-gram	0.088738
insulin	Token	0.632708	D2-m	4-gram	-0.022443
protein	Token	0.628168	Stai	4-gram	-0.082046
kinase	Token	0.608392	mig	3-gram	-0.083135
human	Token	0.536695	Reve	4-gram	-0.096548
proteins	Token	0.535368	ing	3-gram	-0.099499
	greek	0.498111	GnT	Token	-0.099619
	combi	0.489201	owl	3-gram	-0.100996
serum	Token	0.480326	231	Token	-0.104751
	lowerUpper	0.457806	ZII	Token	-0.105133
	singleCap	0.438028	had	Token	-0.106545
factor	Token	0.438028	we	Token	-0.107104
wild-type	Token	0.389359		[..]	
	initCaps	0.366269	that	Token	-0.174203
mutants	Token	0.340689	scre	4-gram	-0.175351
genes	Token	0.340352	OH	Token	-0.179445
promoter	Token	0.327395	ims	3-gram	-0.182513
receptor	Token	0.323412	be	Token	-0.186265
polymerase	Token	0.305972	.	Token	-0.188904
complex	Token	0.292019	To	Token	-0.189576
receptors	Token	0.292019	acyc	4-gram	-0.191766
c-myc	Token	0.292019	the	Token	-0.192838
sites	Token	0.243349	off	Token	-0.197588
mutant	Token	0.243349	rank	Token	-0.198915
domain	Token	0.231541	Dar	Token	-0.205479
sequence	Token	0.216691	(	Token	-0.206405
sequences	Token	0.216683	omit	4-gram	-0.220064
domains	Token	0.215116	nost	4-gram	-0.223077
	special+number	0.205077	spit	4-gram	-0.238335
isoforms	Token	0.194679		**allCaps**	-0.243183
	special+upperCase	0.179926	oped	4-gram	-0.246457
	capMixLetters	0.179394	The	Token	-0.246535
	[..]		aged	Token	-0.253814
lare	4-gram	0.105354	are	Token	-0.267228
bicu	4-gram	0.103185	ssif	4-gram	-0.272211
bea	3-gram	0.100539	encoding	Token	-0.447471
[	Token	0.097113	which	Token	-0.535368
ntei	4-gram	0.093310	activate	Token	-0.535368
GTTA	4-gram	0.088738	contain	Token	-0.640844

Table 3.7.: Examples for features and feature classes remaining after 64 iterations of recursive feature elimination. We show features with highest, medium, and lowest weights and omit others. Higher weighted features are more likely to apply to positive examples (protein names), lower weighted features to common words. Names in bold indicate binary orthographic features and the dictionary (Dict), in contrast to single features, like a particular 3-gram. The feature named *special* in Table 3.2 consists of four sub–features that all state if the token contains special symbols together with other characters; two of these are present in the list of top ranking features: *special+number* and *special+upperCase*.

Evaluation on other corpora

We noticed a strong bias toward the underlying corpus. The post–expansion step does not work properly on, for example, the GENIA corpus [Ohta et al., 2002]. We achieved an f-measure of 61.7% using our system trained on 75% of the corpus' sentences and tested on 25%. After the post-processing, the f-measure dropped down to 55.5%, mainly loosing precision performance. The GENIA corpus contains different labels for more than one class of entities, however. It separates proteins from protein families, protein complexes, peptides, substructures of proteins, DNA, RNA and many other sources. The evaluation we ran on GENIA took only into account the protein class, but other names would have been labeled as gene or protein names in BioCreAtIvE. Our system additionally made errors where GENIA annotates nested compound names.

On the YAPEX corpus [Franzén et al., 2002], our systems performed even worse. The f-measure is 37.8% before post-processing, and 37.3% afterwards. 200 abstracts comprise this corpus (970 sentences for training and 941 sentences for testing). In many complete abstracts, a gene name appears more often than only once. If our system was not able to predict this name correctly, the evaluation led to multiple errors. Thus, for corpora that do not consist of isolated sentences but larger discourses, our method should include a feature depicting previous predictions; we could also scan the same piece of text twice, the second time including all previous predictions to also consider tokens following the current position.

However, for the experiments on other corpora, we used the feature set and parameters tuned on the BioCreAtIvE corpus. For the other corpora, the matches had to be exact, without any alternative gene name compositions like in the BioCreAtIvE corpus. This means that in the BioCreAtIvE data set, some names had variable boundaries, for instance, "Her2B" and "Her2B protein" both were valid matches. A thorough analysis, especially of the post-processing step, should reveal the main sources of errors the system makes on other corpora.

Kernel parameter optimization

During feature engineering, we made use of the default linear kernel setting of the Support Vector Machine. However, applying other types of kernels to this specific problem might improve the classifier to some extent. We evaluated the performance of different kernels on the best selection of feature classes identified during the engineering described above.

Disappointingly, all other kernels lead to decreasing classifying abilities. Using polynomial kernels of degree two and three, and a radial-basis-function kernel reduced the F1-measure to 67.1%, 62.2%, and 36.0%, respectively (compared to 70.6% of a linear kernel). Especially the figures for recall dropped significantly using these other kernels. All these experiments were done with the respective default kernel parameters.

Boundary detection

We found that 25% of all errors were caused by imprecise recognition of compound names. This means that our NER procedure finds only parts of a name, or identifies too many consecutive tokens as a compound name. Most times, the post-processing stage did not extent all initial hits to full names properly, adding too many tokens into a name or falsely removing (or not adding) tokens. Only about 1% of boundary-related errors originated from the SVM, were post-processing did not remove initial hits. This happened, for instance, when tokens directly adjacent (but not part of) a protein name were classified wrongly. In 27% of all boundary errors, the tagged sequence exceeded the real boundary to the left, and in 28% the real phrase started left of the predicted boundary. In 32% of the cases, the tagged sequence exceeded the real boundary to the right, and in 31% the real phrase ended after the predicted boundary on the right. In some cases, errors at the left and right happened at the same time. Errors introduced by the post-processing were almost balanced regarding the position (left/right) and also the type (exceeding/insufficient tagging).

3.2. Dictionary-Based Named Entity Recognition

Another approach we explored to recognize named entities is based on matching texts against dictionaries. This method used pre-compiled name lists of instances that belong to the given entity class. The name lists most times originate from renowned databases that explicitly curate data on such entities. For instance, the UniProt database lists (among detailed information on each protein) protein names, gene names, and synonyms for each entry [Bairoch et al., 2005]. Each of these can be taken as a name variant that refers to the respective entity.

We observed that many variations of names exist that are not explicitly curated in the database. We encountered many orthographic, morphological, and structural variations. Nevertheless, these were quite similar to each other, so that a database would curate only one variation as an overall representative (like for "IFN-gamma", "IFN-γ", and "Ifn γ"; see Section 2.1.1). In addition, it is impossible to collect or guess all (existing and future) spelling variations to include in a list of names, as discussed in Chapter2.

There are two main strategies to deal with orthographic, morphological, and structural variations in dictionary-based approaches. The first is to compile a concise dictionary with names as gathered from the database. Recognizing names than relies on inexact string matching. The second strategy is to generate a “loose” dictionary that already includes typical deviations of spelling. These terms then are searched for using exact matching.

In our dictionary-based approach, we follow a hybrid strategy. Starting with a list of terms reflecting names for entries in a biological database, we compile regular expressions that allow for variations such as plural forms, hyphenation, and capitalization. Variations for obvious abbreviations (“IFN”) are introduced as well. For example, the following regular expression covers many variations of the protein name “IFN-gamma":

```
(I(FN|fn)|ifn)[ \-_]?(g(amma)?|{gamma})s?
```

(with `{gamma}` depicting the Greek letter γ).

We built a finite state automaton covering all expressions that we generated in the described way, using the Monq package [Kirsch et al., 2005]. Arbitrary text is then streamed against this automaton. We applied the same strategy for multiple types of entities, in addition to protein and gene names as described before. Table 3.8 lists the data sources we exploit to compile dictionaries for different entity classes. We use all these different dictionaries in the AliBaba tool, fully described in Chapter 6.

Entity class	Database	Number of entries
Cells	MeSH, tree A11	709
Compounds	KEGG	22,635
Diseases	MeSH, tree C	12,401
Drugs	MedlinePlus	15,341
Enzymes	KEGG	23,063
Nutrients	-various-	ca. 90
Proteins and genes	UniProt	199,964
Species	NCBI Taxonomy	208,138
Tissues	MeSH, tree A2-10	1356

Table 3.8.: Data sources that comprise the dictionaries for each entity class. The number of entries does not coincide with the number of entities: synonyms count as different entries, one synonym might represent multiple entities.

Most databases adhere to some nomenclature and use standardized names, while names in literature in much more varieties. In particular, this addresses the problem of lexical and structural variations (see Section 2.1.1). For instance, generating spelling variations for the previously mentioned "IFN-gamma" would not include the long form of the abbreviation "IFN", "interferon". As another example, the collection of Medical Subject Headings[1] prefers the term "lymphocytes", while most authors prefer to use "cells" ("B-lymphocytes" versus "B-cells"). Such problems can only be tackled when synonyms of essential naming components (different for each entity class) are known to the system, for instance using (domain-specific) ontologies or comparable manually compiled resources. An evaluation of our system to recognize (and subsequently identify) human gene names revealed a maximum recall of 91.7% using conversion to regular expressions, as compared to 72% using exact matches on the BioCreAtIvE 2006/07 GN task data [Morgan et al., 2008].) Using the whole set of proteins available in UniProt (see Table 3.8), we obtained 73% precision at 54% recall (f–measure of 62%) on the BioCreative 2006/07 GM task [Smith et al., 2008].

[1]MeSH, see http://www.nlm.nih.gov/mesh/

3.3. Extension of Dictionary-Based Approaches to Named Entity Identification

An extension to dictionary-based approaches for named entity recognition results from the implicit mapping of names (entries in the dictionary) to their origin (entry in the database(s) used to compile the dictionary). If such a mapping exists, namely in the form of identifiers associated with each name, it poses as a step towards named entity identification, which aims at not only spotting entity names, but attempting to resolve them to unique identifiers. This process is often also called *normalization* or *grounding*; find an introduction to this task in Section 2.1.2. Note that this step is independent of whether regular expressions are generated from dictionary entries to allow for variations, or the dictionary is used in an inexact matching strategy. However, for each recognized mention, usually a set of candidates remains, so the mention has to be disambiguated further.

The task of *gene mention normalization* was first pursued in the BioCreAtIvE challenge held in 2003/04 (as Task 1B) for genes from the three organisms mouse, fruit fly, and yeast [Hirschman et al., 2005]. A dataset for human genes became available with BioCreative II in 2006/07 (GN Task, see Morgan et al. [2008]). We extended our approach of dictionary-based protein name recognition with a step for the disambiguation of candidates found by the dictionary matcher. Previous approaches all attempted to solve the disambiguation problem based on the string level, picking the name in a dictionary that was most similar to a recognized name. Limitations of such approaches arise from identical names shared among multiple genes (ambiguity). This is most often the case for homologous genes from different species, but intra-species ambiguities exist also. In a set of 3.5 million genes (EntrezGene, comprising ca. 4700 different species/strains), we found an average of 1.83 genes per name (case–insensitive, no symbols). The average number of species per name was 1.4. If we add names of the referenced proteins, we get an average of 2.13 genes and 2.23 species per name. Fundel and Zimmer [2006] presented a more exhaustive study of inter-species ambiguities, finding, for instance, an overlap of 15.1% between human and mouse genes.

Our approach thus took into consideration not only names of genes, but largely makes use of background information known for each gene [Hakenberg et al., 2008b]. We obtained such background information from various databases, such as EntrezGene, UniProt, and GOA, which all curate different aspects of genes and gene products, and compiled a *gene context profile* for each gene. We compared the profile for each candidate gene with the discourse under consideration (in the case of BioCreative GM, an abstract) and picked the gene which context best fitted the current text. We collected external knowledge from EntrezGene, UniProt, and GOA (EntrezGene: summary, GO terms, GeneRIFs, chromosomal location, interaction partners; UniProt: diseases, keywords, functions, protein mutations, protein length, protein domains, GO terms, interaction partners, tissue specificity; GOA: GO terms.)

For calculating the similarity based on Gene Ontology terms, we searched for GO terms in the current abstract and compared them to the set of GO terms assigned to each gene candidate. For each potential tuple taken from the two sets (text and gene annotation), we calculated a distance of the terms in the ontology tree. These distances yielded a similarity measure for two terms, even if they did not belong to the same sub–branch or were immediate parents/children of each other. The distance took into account the shortest path via the lowest common ancestors, as well as the depth of this lowest common ancestor in the overall hierarchy (comparable to Schlicker et al. [2006]). The distances for the closest terms from each set then defined a similarity between the gene and the text.

We compared the other annotations from each gene's context model according to their type (terms or full texts.) For terms (such as keywords), we calculated the fraction of terms occurring in the abstracts among all terms. For texts (for instance, descriptions of a gene's implications in diseases), we calculated the cosine distance of both bag–of–word representations (TF–weights) and the normalized overlap (fraction of tokens from the disease description that also occur in the abstract.) We excluded 154 stop words and other, non–discriminative tokens such as "protein", "observe", "detected" (see supplementary information.)

All such comparisons yielded likelihoods stating the similarity of the current text with the knowledge available on each gene. For each type of annotation, we normalized all likelihoods for all genes (where applicable) with the highest score, so all values were between 0 and 1. We combined all likelihoods for each gene into confidence measures (between 0 and 1), and picked the EntrezGene ID with the highest likelihood, if this was above a certain threshold. Some annotations (protein mutations and chromosomal locations), if they occur literally in the text, triggered a confidence of 1.0. This was due to the fact that such annotations were sufficient to immediately identify a gene; identical annotations for different genes/proteins were highly unlikely.

In the BioCreative II GN challenge, our approach reached an f-measure of 81%, outperforming all other 20 such systems (statistically significant improvement over the 3rd best, $p < 0.05$); note the challenge was for human genes only. As a recent extension, we not only target genes of a single species, which was the case for the corresponding BioCreative 1 and 2 tasks, but formulate the problem of inter-species gene normalization [Hakenberg et al., 2008a]. As an initial result on a small benchmark adapted from BioCreative 1 and 2 we achieve an f-measure of 81.2% for the overall task of recognizing and normalizing genes from any species. Other modifications in that same system pushed the f-measure for human genes to 86.4%, an improvement of ca. 5% over previous results (measured on the BioCreative II GN test data).

3.4. Related Work

In this section, we will discuss related work in the field of biomedical named entity recognition. We start with descriptions of various systems, sorted by the main strategies underlying each system. We conclude with a comparison of all strategies in general, regarding performance, manual workload, manageability, and insights into predictive models.

Entity type	Studied in
cell lines	JNLPBA'04, see Kim et al. [2004]
cell types	JNLPBA'04, see Kim et al. [2004]
diseases	Rosario and Hearst [2004]
DNA, RNA	Nobata et al. [1999]; JNLPBA'04, see Kim et al. [2004]
enzymes	Schmeier [2005]
genes and proteins	Song et al. [2004b], Mitsumori et al. [2004], Zhou et al. [2004b], Mika and Rost [2004], Krauthammer et al. [2000], Nobata et al. [1999]; Hsu et al. [2008]; also see the overviews regarding BioCreAtIvE, see Yeh et al. [2005]; JNLPBA'04, see Kim et al. [2004]; and BioCreative II, see Smith et al. [2008]
GO terms	Survey in Blaschke et al. [2005], Camon et al. [2005]
proteins	Hanisch et al. [2005]
species	Gaizauskas et al. [2003], Schmeier [2005], Kappeler et al. [2009]
sources (cell [-line], nucleus)	Nobata et al. [1999]

Table 3.9.: Overview of systems proposed for the recognition of various types of named entities. Some systems consider multiple types at once.

The first systems for biomedical named entity recognition have been proposed since 1998 [Fukuda et al., 1998, Nobata et al., 1999]. Systems started with the recognition of protein names in text, which still is one of the most studied tasks, and soon were expanded to gene names, DNA, and RNA. To distinguish between these four types is not trivial, and sometimes even human experts can not decide between either. Thus, very often, all four get combined into a single category. Table 3.9 shows systems for the recognition of different kinds of entities.

For the recognition of protein names, Table 3.10 lists systems by their reported performance. We give performance measures as presented by the authors; however, most system were evaluated on different, often proprietary corpora, and performance measures are not comparable. Some systems try to recognize different classes of entities, where some types might be easier than others, and a system might yield completely different results for different entity classes. Even the definition of an entity class varies from system to system. Evaluation procedure most times is strict, meaning entire names have to be recognized, but sometimes loose, where detection of any part of a name is regarded sufficient for successful recognition. Other systems try to solve not only NER, but also normalization of named entities, which poses a much harder challenge.

Published values for the performance of biomedical NER range from 40% to up to 95% in f-measure. The best performing systems in the BioCreAtIvE challenge 2003/04 achieved an f-measure around 83% for gene and protein names [Zhou et al., 2004a, McDonald and Pereira, 2004]. The best system in the BioNLP/NLPBA competition 2004 reached about 70% f-measure, combined for the recognition of proteins, DNA, RNA, cell types, and cell lines [Zhou and Su, 2004]. Published systems for identification of protein names (EMN) yield up to 92% F1-measure, strongly depending on the species associated with a name. High performance is only achieved for species for which highly standardized and well-used nomenclatures exists, for instance yeast (also evaluated in the BioCreAtIvE challenge 2003/04; see Hirschman et al., 2005). The most recent and largest benchmark, which is now used in most studies, was prepared for the BioCreative II Gene Mention task [Smith et al., 2008]. Among 19 participants, the best performing system [Ando, 2007] reached an f-measure of 87.2%, a statistically significant improvement over the 4th best system ($p < 0.05$). Since 2007, other systems have been evaluated on BioCreative II GM, currently achieving an f-measure of 88.3% [Hsu et al., 2008].

Studied in	Basic method	F1	Test collection
Fukuda et al. [1998]	Rule sets	95.9	80 abstracts
Gaizauskas et al. [2003]	Rule sets	85.7	1513 abstracts
Nobata et al. [1999]	C4.5	87.5	100 abstracts
Nobata et al. [1999]	naïve Bayes	70.3	100 abstracts
Chang et al. [2004]	naïve Bayes	≈82	Yapex
Krauthammer et al. [2000]	BLAST	75.1	1162 gene names
Chang et al. [2004]	SVM	82.5	Yapex
Takeuchi and Collier [2003]	SVM	79.0	100 abstracts
Mitsumori et al. [2004]	SVM	78.1	BioCreAtIvE'03/04
Song et al. [2004b]	SVM	73.8	BioCreAtIvE'03/04
Zhou et al. [2004a]	2 SVMs + HMM	82.6	BioCreAtIvE'03/04
*Hakenberg et al. [2005a]	SVM	72.4	BioCreAtIvE'03/04
Zhou and Su [2004]	2 SVMs + HMM	72.6	JNLPBA'04
Mika and Rost [2004]	4 SVMs + Dict.	75.5	Yapex, biased
Mika and Rost [2004]	4 SVMs + Dict.	70.8	GENIA, biased
Ando [2007]	ASO	87.2	BioCreAtIvE II GM
Kuo et al. [2007]	CRF + Dict.	86.8	BioCreAtIvE II GM
Klinger et al. [2007]	CRF	86.3	BioCreAtIvE II GM
Hsu et al. [2008]	CRFs	88.3	BioCreAtIvE II GM
Leaman and Gonzalez [2008]	CRF	86.4	BioCreAtIvE II GM
*Our approach	Dictionary	73.1	BioCreAtIvE II GM
Song et al. [2004a]	SVM & CRF	66.3	JNLPBA'04
Friedrich et al. [2006]	CRF	71.5	JNLPBA'04
McDonald and Pereira [2004]	CRF	82.4	BioCreAtIvE'03/04
Seki and Mostafa [2003]	Prob. model	81.4	Yapex
Finkel et al. [2004]	MaxEnt	70.1	JNLPBA'04
Chang et al. [2004]	MaxEnt	82.1	Yapex
Hanisch et al. [2003]	Dictionary	92.4	470 abstracts

Table 3.10.: Performances (F1-measure in %) reported for different systems for named entity recognition of protein and/or gene names. The JNLPBA'04 test collection is a subpart of GENIA. Because evaluation methods and corpora differ by large, most results are not comparable, but only give hints on the effectiveness of approaches. Studies marked * refer to our own approaches, as discussed in this chapter.

3.4.1. Dictionary-based approaches

Dictionaries provide large collections of exemplary names for an entity class. Exactly matching dictionary entries against text is a simple and very precise NER method, but generally yields only very poor recall. To compensate, one can either use inexact matching techniques, try to loosen the syntax of entries in a dictionary, or try to integrate more and more data sources into the dictionary (cmp. Section 3.2). The latter approach refines the dictionary by automatically generating typical spelling variants for every entry. The extended dictionary is then used for exact matches against the text [Hanisch et al., 2003, 2004]. When dictionaries are extracted from biological databases, such approaches bear the important advantage that they are able to recognize and normalize each name in a single step. Homonyms in a dictionary of course lead to ambiguities even within an entity class. At least the set of possible instances is reduced to very few candidates, which might be refined further (see 2.1.3 in Chapter 2).

Hanisch et al. [2003] proposed a dictionary-based approach that relies on a large, curated dictionary and a token-based search algorithm. They observed that the ordering of tokens in a compound name is only semantically significant up to a certain limit. Thus, re-orderings of tokens appear quite frequently. Tokens are grouped into six categories that bear different semantic significances. They can either be common ("and"), non-descriptive ("precursor"), modifiers ("receptor"), specifiers ("alpha"), standards ("TNF"), or delimiters ("."). Based on match and mismatch values for either group, the system analyzes a sliding window for candidate names. When a candidate term gets above or below certain threshold, it gets compared against the full dictionary or pruned, respectively. The dictionary extracted from HUGO, OMIM[2], SWISSPROT and TREMBL[3]. Exhaustive curation generated new synonyms (spelling variations) and removed ambiguities and irrelevant entries. The final dictionary consisted of 151,700 synonyms for 38,200 entries. On a set of 470 abstracts with 147 different objects derived from TRANSPATH[4], a maximum precision of 90% at 95% recall was reported.

The WHATIZIT system uses a pre-compiled list of names that all refer to one entity class [Kirsch et al., 2005, Rebholz-Schuhmann et al., 2005]. From each entry, it generates a regular expression recognizing this entry and common spelling variations using heuristic rules (mostly rules that involve variations in suffixes, capitalization, and Roman vs Arabic numbering). All such regular expressions taken together are represented as a finite state automaton that parses arbitrary text. All initial entries are taken from the UniProt database [Wu et al., 2006], and the automaton maps matching strings in a text to the respective UniProt-IDs. We extended this approach by adding more rules to create typical spelling variations of proteins, such as generating structural variations ('CD95 receptor' ↔ 'receptor of CD95') and lexical/morphological variations ('R' ↔ 'receptor'), which are not included in Whatizit (see described in Section 3.2).

Inexact matching using BLAST

An approach using BLAST has been proposed by Krauthammer et al. [2000]. This system encodes the English alphabet, together with numbers and special symbols, as fourtuples of DNA bases. In this way, a dictionary of named entities (derived from GenBank [Benson et al., 1998]) and the text get encoded as DNA sequences. The original BLAST algorithm [Altschul et al., 1990, 1997], an inexact string matching algorithm,

[2]http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM
[3]http://www.expasy.org/sprot/ for SwissProt and TrEMBL
[4]http://transpath.gbf.de

searches similarities between the encoded phrases and named entities in the dictionary. This approach yields a precision of 71.7% and a recall of 78.8% on a review article with 1162 genes by two human annotators.

3.4.2. Rule-based approaches

Rule-based approaches use hand-crafted heuristic rules or machine-learned rules to describe the composition of named entities and their immediate context. Compared to other approaches, rule-based systems are often not robust towards unseen names and unseen surrounding contexts. Another severe drawback is the time-consuming process of manually deducing and tuning patterns from examples, and the substantial amount of interference between rules. Sophisticated rule-based systems often reach very high precision; but the more specific the chosen rules are, the lower the achieved recall is.

Fukuda et al. [1998] use surface clues (capital letters, symbols, digits) to extract candidates for protein names. In a subsequent step, these candidates are connected to surrounding terms using syntactic rules. For example, candidates get expanded to contain functional trailing terms such as "domain", "kinase" or "receptor". This expansion includes previously unconnected terms appearing between the initial candidate and the new functional term. Non-adjacent candidates get connected if all terms in between them are only nouns, adjectives, or numerals. The system even deals with enumerations and occurrences in parentheses. No additional information other than surface clue rules are needed for finding candidates. Fukuda et al. use the Brill part-of-speech tagger trained on common English texts to obtain syntactical information for the rule-based term expansion [Brill, 1992]. The approach achieves a precision of 94% at 97.7% recall, as evaluated on 80 abstracts. Main false positives errors were due to virus and cell names. The authors noticed that about 95% of all names actually include a candidate core term (meaning a term with capital letters, numbers, etc.) or a functional term, and that the remainder are common simple protein names such as "insulin".

PASTA combines heuristic with machine-learned rules [Gaizauskas et al., 2003]. The system ultimately tries to fill a set of templates for twelve different entity classes (ranging from "protein" to "quarternary structure"). This is done in two major steps, one for the initial terminology identification and the second for the actual template filling. A subset of the rules are terminological rules. These rules identify head term (for instance based on a frequent affix, such as "-ase", or by look-up in a dictionary). A set of subsequently applied grammar rules then incorporates expansions of head terms, for instance using modifiers or numerals, to also find compound names. PASTA learns some rules from a rather small training corpus. However, combining learned rules and heuristic rules to one set overcomes the incompleteness of the small sample. For protein names, this approach achieves a precision of 91% at 81% recall,

One system presented at BioCreAtIvE 2003/2004 used a basic set of four patterns in which protein names frequently occur in text [Kirsch and Rebholz-Schuhmann, 2004]. Very often, a protein/gene name is embedded in a sentence structure like

- "the X protein" or
- "a Y-like gene".

Such methods works quite well when the recognition of one occurrence of a name in a whole text is sufficient. Given a large amount of occurrences, it gets highly probable that at least one occurrence will be embedded in a frequent pattern. As we will see in the later part of this thesis, for relation mining and language patterns, it is thus the hard task to find the "right" (precise enough, general enough) set of patterns (see Chapter 4).

3.4.3. Classification-based approaches

The most popular technique in biomedical NER is to transform the NER task into a classification problem, either for single words or for multi-word phrases. Applying a naïve Bayes classifier, Nobata et al. [1999] trained a model to distinguish between four different categories of named entities: source, protein, DNA, and RNA. They took terms occurring in names taken from SwissProt and GenBank as features for their model. Nobata et al. also applied there system to a decision tree learner, C4.5., and report that the latter outperformed the Naïve Bayes model. On a test collection containing 4248 protein names, naïve Bayes achieved an F-measure of 70.3%, while C4.5 reached 87.5%. They also studied different strategies for term identification, and report an F-measure of 72.1% on proteins using a decision tree similar to that used for term classification.

Song et al. [2004b] use a modification of the classes separating gene names from other words to the BIO-markup. This includes not only the differentiation between tokens inside and outside a gene name (I/O), but they add a class for tokens at the beginning of a compound gene name (B). This distinction transforms the binary decision (named entity or not) into a three-class problem, distinguishing the first word of a named entity from following words. Our own system presented in Section 3.1 uses the I and O classes only. A significant contribution to the feature space is a gazetteer lookup for tokens and phrases based on edit-distances. The edit-distance does consider lexical variations. In our own approach, we use exact matches of candidate terms, but against a gazetteer containing spelling variants. Song et al. try to enlarge the training sample artificially. The system reuses sentences from the training corpus with replaced gene names. Assuming that every named entity appears only in base noun phrases, all tokens outside noun-phrases plus determiners are excluded completely. The method achieves 73.8% F1 without and 66.7% with the additional training examples, both on the BioCreAtIvE'03/04 data set, and 66.3% on the JNLPBA'04 data. This indicates that the restriction to noun-phrases does not improve the overall performance. The recall is about 1-2% higher when using artificial examples, but the predictions contain much more false positives (precision drops by 17%).

YAMCHA uses the same extension to BIO [Mitsumori et al., 2004]. The method uses a context window to extract features for training a support vector machine (surface clues and morphosyntactic properties). SWISS-PROT and TrEMBL provide data for a dictionary-lookup feature. This feature matches token n-grams against phrases in the dictionary. YAMCHA achieves 78.1% f-measure on the BioCreAtIvE 2003/04 data set. The context window accounted for 3% of the overall performance of the system.

GAPSCORE scores single words based on a statistical model of gene and protein names that quantifies their appearance, morphology and context [Chang et al., 2004].

The systems expands detected words using part-of-speech information. GAPSCORE computes individual weights for features by measuring differences in frequency of appearance in positive examples and arbitrary MEDLINE words, respectively. The authors compared naïve Bayes, maximum entropy, and SVM classifiers, and found that the latter outperforms the others slightly. GAPSCORE achieved an f-measure of 57.6% on the YAPEX corpus for exact matches, and 82.5% for sloppy matches. A false positive filter, accounting for 11.8% of the overall performance, removed common non-gene names ("alpha", "1") from the initial words. Such terms are recovered in the subsequent expansion again. Chang et al. found that the contextual features had the lowest impact on the overall performance.

The system described by Seki and Mostafa [2003] uses surface clues to detect potential protein name fragments and applies an additional false positive filter. A probabilistic model expands single names to compounds. For exact matches, the system achieves 63.6% f-measure on the YAPEX corpus, and 81.4% for sloppy evaluation.

Takeuchi and Collier [2003] propose another approach that uses four different sets of features to train an SVM classifier for multiple classes (Protein, DNA, RNA, and various sources). Starting with orthographic, POS, head noun, and surface word level features, the combination of orthographic and head-noun features proved to perform best. A 10-fold cross-validation yielded an f-measure of 74.2% (combined-class performance); for proteins, the system scored 79% f-measure, both on a corpus of 100 journal abstracts.

As we saw already in Section 3.1, classification-based approaches highly depend on a careful selection of features to represent both (or even more) classes. Choosing the best performing set of features breaks down in a time-consuming and error-prone process. This hardly identifies sufficiently enough and distinctive single features and feature classes. Classification-based approaches sequentially classify each token, but mostly do not take into account the order in which features (or tokens) occur. Features generated from a token or a context window are represented as an order-independent bag of tokens. Some systems differentiate between features seen before, inside, or after a named entity [Mitsumori et al., 2005]. We discuss methods gaining information from the sequential order of tokens the next section.

3.4.4. Sequence-based approaches

In contrast to classification-based approaches working on words or phrases, sequence-based systems predict the complete sequence of part-of-speech tags and classes of named entities for each sentence. Such systems perform a statistical analysis of the training corpus and deduce the most probable sequence of tags for the given word sequence.

Retraining part-of-speech taggers

Kinoshita et al. [2005] use a simple but efficient way of including sequential information by retraining the TnT-Tagger on a biomedical corpus using standard POS tags and tags for entity names. Some post-processing stages are added for correcting errors. Rules-based alterations of the names predicted by TnT make up for word boundary-related errors. In addition, abbreviation resolution detects missed hits when either the long or short form is tagged, but not the other.

Conditional random fields

Conditional random fields (CRFs) are another probabilistic sequence tagging framework [McDonald and Pereira, 2005]. While the TnT HMM uses its built-in feature generation methods tested on non-scientific text, using generic implementations enables the use of tuned feature sets. McDonald et al. extract features for words, orthography, and n-grams, all from a context window, to represent each token. On the BioCreAtIvE corpus, CRFs proved to be one of the best performing systems for NER of gene and protein names.

Friedrich et al. [2006] compare conditional random fields and support vector machines trained on similar sets of features. On the JNLPBA'04 corpus, CRFs outperformed SVMs by 6.3% in f-measure (71.5% over 65.2%). Surface clue features, 3-gram prefixes and suffixes, word stems (Porter), part-of-speech tags, begin delimiters ("the"), and different dictionaries for other classes of entities (tissues, amino acids) comprise the feature set. Like we also discovered in our token-based approach, integrating POS information decreased the overall performance. Friedrich et al. [2006] also tested other techniques, namely instance based learning (TiMBL), naïve Bayes, and maximum entropy markov modeling (MEMM). CRF was found to outperform all of these.

BANNER [Leaman and Gonzalez, 2008] is an attempt to provide a portable NER system using a second-order CRF and a feature set that is not specific to any (biomedical) task. Features include, however, dictionary–lookups; this may have an influence on performance at it depends on the availability of such dictionaries for the required entity types. The system supports setting up multiple, distinct dictionaries that result in distinct features; this allows users to include dictionaries for common entity types, providing features with a negative weight for entity types for which no exhaustive dictionary is available. For protein name recognition, BANNER runs with an overall set of 500,876 features (protein name dictionary only) and achieves an f-measure of 86.4% on BioCreative II GM test data.

Hsu et al. [2008] recently presented a system that combines multiple CRF models: first, second, and third order CRFs, one each for reading a sentence in forward or backward direction (as a union of six models), and an intersection of two models generated by two different CRF implementations. Hsu et al. do not give time estimates for training and application; however, each model consists of slightly more than 5 million features. Their method achieves an f–score of 88.30% on the BioCreative II GM test data, which outperforms systems previously proposed by about 1%. Recent results suggest that backward parsing outperforms the more common form of forward parsing by around 1% for the task of protein name recognition [Li et al., 2009].

3.4.5. Hybrid approaches

Very recent NER systems often do not only rely on a single technique, but use processing pipelines with multiple stages. Pre-processing prepares and enriches texts, using NLP methods for proper sentence-splitting and POS tagging, and adds annotation from external resources. Later, several machine learning techniques are applied in parallel to mark entities. Post-processing stages deal with refinement of predicted candidates, resolution of abbreviations, and exploitation of multiple occurrences of the same named entity within the text.

Whenever more than one classifier is used in parallel, post-processing must also take decisions in case of conflicting results. This may be implemented using meta-learners, which are based on the predictions of other models. The simplest forms of meta-learners are voting schemes, based on simple majority voting or weighting of the single predictions. The POWERBIONE system, presented in the following section, uses an ensemble of two HMMs and one SVM classifier with majority voting. NLProt uses three SVMs trained on different feature sets. A forth SVM functions as a meta-learner to combine the result of the three initial predictions and a dictionary.

Combination of HMMs and SVMs in a voting scheme

Zhou et al. [2004a,b] propose an ensemble methods of two HMM and one SVM classifier. The two HMMs are trained on different corpora (BioCreAtIvE and GENIA). This seems to enable the system to adapt to different corpus properties. The SVM uses a feature set similar to ours (see Table 3.2). Rather than having one feature representing the appearance of a candidate token in a vocabulary, POWERBIONE has one dimension for each word in the vocabulary. Orthographic features for the representation of tokens are comparable to ours, but include checks for parentheses, punctuation, and stop words as well. Our own implementation removes the 100, 1,000, and 10,000 most common English words (parameterizable), but the composition of POWERBIONE's stop word list was not reported. Zhou et al. distinguish two kinds of triggers indicating whether a token is included in two separate lists of words. One consists of words typically occurring inside gene names, and the other contains words typically found in the local context of gene names. Our own system measures the distance to such keywords and sets the value for one single feature according to this distance. However, as the authors propose an ensemble method, the exact difference in prediction performance compared to our system is not known. The same is true for the influence of different feature sets.

Meta-learning over SVMs and dictionaries

Another hybrid approach was proposed by Mika and Rost [2004]. This system uses three different SVMs that were trained on on different inputs: the protein names, their environment, and on both information, respectively. A fourth SVM then takes the decisions of the other former SVMs as input, plus an additional score derived from matching strings against a protein-dictionary. From these values, a final model is learned that outperforms each single SVM. Mika and Rost report a performance of 75.5% F1 (balanced precision and recall) on the Yapex corpus, and 70.8% (63% precision at 81% recall) on GENIA [Franzén et al., 2002, Kim et al., 2003]. Dropping the protein name information resulted in 6% lower F1-measure, dropping the information taken from the environment decreased F1 by 12%. Without the dictionary matches, F1 dropped to 60% on Yapex and 53% on GENIA.

3.5. Conclusions

Many ideas showed up in the last years that help tackle the problem of biomedical named entity recognition. Influenced by the field of computational biology, systems proposed focused primarily on the recognition of protein names. Balanced performances around 85% for precision and recall are state of the art when solving the problem in general. For specific subtasks and restricted domains, systems can get as precise as 98%.

Conditional random fields [Sutton and McCallum, 2006] have come out as the method of choice for modeling NER problems. The best performing system at BioNLP/JNLPBA [Zhou and Su, 2004] used a combination of SVM and CRF, and two methods relying only on CRFs came close behind [Friedrich et al., 2006, Settles, 2005]. The BioCreative 1 and 2 protein NER tasks led to basically the same insights, although a semi-supervised learning strategy [Ando, 2007] was the best system during the BioCreative challenge in 2006/07. The currently best performing system on this benchmark was proposed by Hsu et al. [2008], again based on CRFS, achieving an f-measure of 88.3%.

There are some reasons to use dictionary-based approaches rather than classification of words. The first is the inherent mapping of names to identifiers, when a dictionary is derived from a database. This helps when not only recognition is needed, but a proper identification. Classification-based systems initially are not able to solve this problem, though Mika and Rost [2004] showed a hybrid approach that included identification. The second reason is their quick adaptation to include more terms or to build systems for entirely different sets. Most strategies for inexact matching and generation of potential synonyms for terms can directly be transferred to other domains. Third, we have to consider the time performance necessary to train classifiers on large features sets, or apply models to new texts. Especially for on-demand queries, one cannot always afford to build large feature sets out of terms and texts, which takes a comparably large amount of time.

The disadvantages of dictionary approaches originate from their reduced flexibility and restricted possibilities to exploit context information. Unknown synonyms and overly exotic variations cannot be detected even by inexact matching strategies such as alignment and Levenshtein distances. Classification-based approaches also draw valuable information from the immediate context of a candidate term ("A kinase phosphorylates B").

It shows that systems have to be robust towards unseen names. We saw that systems are able to deal with orthographic variations, exotic variants, and even paragrammatical phenomena such as spelling errors. Regarding these, refined dictionaries and n-gram features (characters and tokens) provide the best resources, in particular with respect to named entity identification. Entirely new names that appear rather *ad hoc* and do not follow any nomenclature convention (like some abbreviations) can be recognized only when systems pay attention to the immediate context (surrounding sentence structure). Only after new names get adopted by the community they will occur in knowledge repositories. This, on the other hand, necessitates a permanent curation of dictionaries over time; again, this is needed to track arising data on exactly identified entities. Hybrid approaches involving dictionaries and machine learning seem to produce the best performing systems so far, especially when trained and tested on various sources.

Identifying names can not rely on the analysis of single passages of text only. It necessitates the integration of external knowledge to distinguish terms from different vocabularies. Such knowledge stems from annotation stores curated for the objects of interest, for instance knowledge repositories such as UniProt or OMIM. These provide facts than can be compared with the current text and an occurrence of a name to draw conclusions concerning relevance, exact meaning, and further facts (entity class, entity identifier, entity variations).

Design of named entity recognizers acts in accordance with the definition of the exact task to tackle. More general systems that shall provide overviews over a broader field of knowledge typically work on large text collections. The output leans towards a collection of more basic information. In such cases, systems should ensure very high precision rates in order to distinguish different (classes of) entities from each other. However, domain experts are able to spot obvious mistakes immediately and quickly remove falsely extracted entities. The principle idea is that recall comes with the huge amount of texts to analyze. To loose a single occurrence of an entity is not at all a problem, because large text collections, even in very specific sub-domains, tend to be very redundant concerning basic facts. Introductions and discussions (of related work) in published articles always roll-up previously gained knowledge, and always use a different form of presentation (names, grammar, sentences). Such information is relatively easy to extract from collections.

The other direction is building very specific text mining systems (starting with NER) that are concerned with restricted sub-domains only. Domain experts want to know everything about a very small set of objects. Predominately, of course, they are interested in new and exotic facts. Text mining needs to incorporate background knowledge to catch up with the experts, and often will produce "boring" and well established facts. By exploiting the experts' knowledge and interest in specific subsets helps to design systems. Good starting points (knowledge sources like annotation databases and dictionaries for synonyms) are already given; these are much easier to maintain when only restricted sets are involved. We see that for such systems it is mandatory to ensure a high recall, so no single –however peculiar– fact will slip through, but is instead recovered from existing publications and retrieved immediately when appearing as a novelty.

4. Language Patterns

Written scientific language usually is rather complex. For instance, the average number of words in a sentence from a biomedical abstract is about 26.5[1]. Authors explain experimental setups, describe results, and assess known and new information in the context of complex interrelationships. Even human readers are not always able to grasp all information and their meaning at first passage. Deeply nested sentences, many relative clauses, and rather decorative attachments hinder quick and easy parsing of sentences. However, we observe basic sentence structures that occur quite often across scientific texts of a particular domain. Descriptions of similar types of observations often follow a certain pattern, which we refer to as language patterns.

In the simplest case, language patterns are sequences of terms that are commonly used to describe similar facts, in our case, protein-protein interactions. Words are combined with annotations into placeholders; such annotations include

- positions of entities (protein names),
- type of a relation (activation, bond), and
- dependency between the partners (agent and target, if any), also called direction.

Note that the terms in a pattern need not be words as they appear in natural language, but can also be formed by linguistic tags. An example for a simple language pattern consisting only of natural words is the following sequence; each single term is enclosed in brackets to separate it from the others, and annotation is omitted:

'`(FADD) (activates) (the) (procaspase-8)`'.

Once we have identified a pattern, we can match it against arbitrary text. If the text (usually, a sentence) looks similar to the pattern, we assume a similar meaning – in this case, a protein-protein interaction being described in the text.

A pattern matches a new sentence if the sequence of terms in the pattern is similar to a sub–sequence of terms in the sentence. This similarity can be defined and computed in various ways. One example are patterns encoded as regular expressions, which yield a binary decision (match, non–match); another example are metrics that calculate a quantified similarity of two sentences, and the matching criteria is defined using a threshold (also see similarity measures discussed in Section 2.2.2). These sentence-similarity metrics originate from string–similarity; for instance, Levenshtein or Hamming distance, or alignment score. We also distinguish between exact and inexact matching: using regular expressions, a sentence has to match a pattern exactly, although the regular expression might allow for some variations (e.g., wild cards); using string–similarity measures, mismatches within the sentence are allowed, but result in penalty scores. If a

[1]Statistics on the GENIA corpus, also see http://www.ifi.unizh.ch/cl/kalju/download/depgenia/.

pattern of known meaning, for instance, a protein–protein interaction, matches a given sentence, we deduce that a protein–protein interaction is described also in this sentence. In many cases, a pattern will not match the whole sentence, but rather a part of it. This is especially, but not only, true when sentences contain decorative attachments like "Our experiments show that .." only after which the actual statements occur.

The previous pattern matches only in very specific cases, because it requires the appearance of four concrete terms in a given sequence (plus it will never retrieve new information). To generalize the pattern, we may replace terms comprising protein names with a special terms:

'`(PROTEIN) (activates) (the) (PROTEIN)`'.

This pattern bears no constraints towards the exact name of the proteins any more. In the same manner, we may generalize other words with their respective *part-of-speech tags* (POS tags) as terms:

'`(PROTEIN) (VERB) (DETERMINER) (PROTEIN)`'.

The inner part of this pattern matches any verb followed by any determiner. Thus, the following (parts of) sentences both match this pattern:

'TGF-alpha binds the EGF receptor' (a different verb is used) and
'IL-1 activated an NF-kappaB' (different tempus and determiner).

However, we do not want to allow for any verb, because not every verb indicates a protein-protein interaction. To avoid such false positives, we use sets of tokens that describe as many examples from a reference corpus as possible, while keeping the precision of each pattern on a high level. Lists of verbs, nouns, adjectives, and adverbs that we use to describe types of protein-protein interactions can be found in Appendix A.3. We will distinguish such verbs, nouns, etc. from ordinary ones with a preceding '`I`', for example, `IVERB` indicates a verb that indicates a protein-protein interaction.

In a similar manner, we may generalize tokens by using their respective word stems. This allows us to find similarities when the part-of-speech tag (for instance, the tempus) differs, but the basic word is the same.

'`(PROTEIN) (activates) (the) (PROTEIN)`'

and

'`(PROTEIN) (activated) (the) (PROTEIN)`'

in principle describe the same situation, so a generalization could be

'`(PROTEIN) (activat*) (the) (PROTEIN)`',

which would also capture the token 'activating', for example. Note that in principle, the noun 'activation' would also match the given word stem. This might make sense only in a longer sentence. We have identified three different types of annotations for terms:

- token: surface words as they appear in the text
- stem: surface word reduced to its stem (if any)
- tag: part-of-speech annotation or entity class (if any)

While token and stem define the meaning of a term, tags define the dependencies. Patterns contain terms and from their sequence and annotations, we are able to deduce the meaning of and dependencies within matching sentences.

We can view language patterns as regular expressions using tokens, word stems, and part-of-speech tags, which include semantic types (here: proteins). Similar patterns may be combined, so that the combination of the initial patterns

`'(PROTEIN) (activates) (the) (PROTEIN)'` and

`'(PROTEIN) (activated) (PROTEIN)'`

results in the pattern

`'(PROTEIN) (activates|activated) (the)?  (PROTEIN)'`,

where '?' indicates the now optional term 'the'.

This example already shows one of the advantages of using language patterns. The resulting pattern now recognized the (partial) sentence

`'PMA activated the PKC'`,

which would not have been found using the single initial patterns (and exact matching).

In the following, we first explain how patterns can be represented in Section 4.1, introducing consensus patterns and multi-layer annotations. We then describe how to match patterns against new text, see Section 4.2. Lastly, we explain strategies for inexact matching of sentences, and for computing consensus patterns via sentence alignment. Section 4.2.2 proposes sentence alignment as a strategy that allows for additional variations (as compared to other string–similarity measures; alignment allows for pre–defined mismatches that do not or only minimally decrease the matching score), and yields a confidence score rather than a boolean expression depicting matches and non–matches. In an actual application scenario discussed in Chapter 5, we show how to identify and refine sets of patterns from a large, unannotated corpus in Section 5.1. Experiments and results will also be discussed in Chapter 5, focusing on the extraction of protein-protein interactions.

4.1. Pattern representation

Formally, a pattern P consists of a sequence S of terms, a set $\mathcal{R}$ of relations, and a set $\mathcal{C}$ of conditions. In the sequence of terms, each term t_i has multiple annotations from a fixed set $\mathcal{A}$ assigned to it. For our experiments, the set of possible annotations consists of tokens, word stems, part–of–speech tags, and entity classes. To distinguish between each of these types, we introduce annotations layers. Each layer contains the information on one particular type of annotation. We assign to each term of pattern the values allowed at this position for each possible type of annotation. For example, a term annotation might contain all tokens allowed at its particular position in the "token"–layer. The set of relations for each pattern stores the positions of the interaction partners in the sequence S, the positions of interaction types, and the direction (the agent–target dependency, if any). The set of conditions contains parameterized thresholds that determine if a new sentence matches the pattern or not. For the sentence alignment, this would be a minimum alignment score, for instance (see Sentence alignment section, 4.2.2).

4.1.1. Consensus patterns

As described above, patterns result from observations in typical sentences describing the required kind of information. Comparing observations, that is, multiple evidences that share the same meaning, yields variations typical to certain positions among similar patterns. In the above examples, patterns might differ only in that they allow for the tokens 'activates' or 'activated' at the same position, or include an optional determiner. The comparison of multiple similar patterns will identify such variations, which we may all encode in a single pattern that generalizes these original patterns. We call the resulting patterns *consensus patterns*, as they reflect the commonalities and differences of all initial patterns. This is conform with the naming in sequence alignment, where multiple sequence alignment (MSA) ultimately defines consensus sequences. We will discuss MSA in Section 4.3.

Figure 4.1 presents an example (using part-of-speech tags instead of actual tokens for simplicity) that combines five patterns into one pattern. In the last row, CP represents the resulting consensus pattern, showing possible variations at two of the five positions; it also assigns weights according to occurrence frequencies to each single observation (in this case, a particular part-of-speech tag). Figure 4.2 shows another representation of a consensus pattern, as a sequence logo showing the information content (bits) per position.

Figure 4.1.: Combination of five initial patterns (1–5) into one pattern (CP) for POS and entity class tags. Weights represent POS tag frequencies for each position.

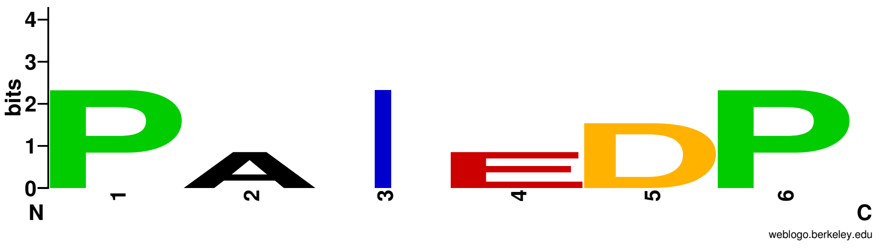
1	PTN	SYM	PTN	IVBD	PTN
2	PTN	CC	PTN	IVBD	PTN
3	PTN	SYM	PTN	IVB	PTN
4	PTN	CC	PTN	IVBD	PTN
5	PTN	CC	PTN	IVBD	PTN
CP	$\text{PTN}_{5/5}$	$\text{CC}_{3/5}$ | $\text{SYM}_{2/5}$	$\text{PTN}_{5/5}$	$\text{IVB}_{1/5}$ | $\text{IVBD}_{4/5}$	$\text{PTN}_{5/5}$

4.1.2. Multi–layer patterns

A pattern can be represented in different ways. In the previous examples, the representation was one–dimensional: a pattern was a sequence of terms, and each term was either an entity class (`PROTEIN`), a part–of–speech tag (`DETERMINER`, `IVERB`), a token (`activates`), or a stem (`activat*`); these examples thus reduced the representation of a pattern to a single layer of mixed content. A scheme including POS tags and entity classes only has been proposed by many other systems so far (for instance, Blaschke and Valencia [2002], Huang et al. [2004], Hao et al. [2005]; also see Section 5.4). Note that often tags for POS and entity classes are collapsed into a single layer, tags for entity classes replacing POS tags; e.g., a noun that is also a protein is replaced with the tag for protein.

Figure 4.2.: Sequence logo of a consensus pattern, with original phrases on top. P, protein; D, determiner; E, preposition; I, i-verb or i-noun; A, adverb. Created with WebLogo, http://weblogo.berkeley.edu/.

P		binds			P
P		interacts	with	the	P
P	strongly	binds	to	the	P
P		bound	to		P
P	immediately	associates	with	the	P
P	strongly	binds		the	P
P	never	bound		the	P



Let us consider the latter scheme from two perspectives relevant to effectiveness: precision and recall. As we have mentioned in the previous section, it is not advisable to allow any noun (verb, adjective, adverb) at a position of a pattern that requires the mention of a specific type of interaction (for example, ‘interaction’). Restricting each pattern (or all patterns at once) to limited lists of potential interaction–indicating nouns etc. helps to rule out 'random' mentions of two proteins that do not point to a specific interaction. A pattern

```
'(INOUN) (PREP) (PROTEIN) (CC) (PROTEIN)'
```

would match

‘interaction of IL-1 and NF-kappaB’,

but also

‘analysis of IL-1 and NF-kappaB’,

which is not (necessarily) an evidence for an interaction between these two proteins. A restriction in this case yields an increased precision. Also, not every noun is suitable for every pattern (‘A-B binding’, but not ‘A-B phosphorylation’), the same holds for verbs (‘A is activated by B’, but not ‘A is interacted by B’); thus, large and unspecific lists can decrease the precision. On the other hand, such restricted lists have to include every possible form of all nouns, and are not easy to define manually. With respect to the automated collection of language patterns (see Section 5.1), we propose a way to allow for variations of a pattern that were not observed before, and maintain precision at the same time.

Representing a pattern only as a sequence of part–of–speech tags looses valuable information. We have shown previously the problem of allowing for too many verbs, nouns, and adjectives. In the same manner, prepositions often are specific to a certain position in a pattern and cannot be exchanged with each other. Using tokens to describe a position in a pattern might be too strict; this could be loosened using word stems as shown before. We propose to capture different aspects of a single position of a pattern using multiple annotation layers:

- one layer defines the sequence of possible part–of–speech tags, and thus resembles traditional approaches;
- one layer stores tags of entity classes (empty when a term does not belong to any entity class under consideration);
- one layer contains all tokens allowed at each position;
- one layer contains all word stems or lemmata.

We may merge the first two layers, POS tags and entity tags, into a single layer; all following examples will use this convention.

When comparing a new text to a pattern, all layers of a term contribute to the decision (match or mismatch) for this position, and thus influence the overall decision for the whole pattern. Whenever a token in a sentence does not match the proposed position in the pattern, at least the word stem or POS tag might fit. This would result in a lower score of the decision function. Figure 4.3 shows an example for a multi-layer pattern. It resulted from two different initial observations (described as patterns):
`'(PROTEIN) (blocks) (PROTEIN)'` and
`'(PROTEIN) (regulated) (PROTEIN)'`.
The combination of both patterns enlarges the list of observed tokens for the verb-position, as shown before. It also introduces variability of a different kind by combining word stems and part-of-speech tags: this consensus pattern accepts tokens that were not observed before, but the word stem of which was observed, and also the part-of-speech tag fits any tag observed. The multi-layer pattern in Figure 4.3 recognizes the sentence
`'UPI regulates 11beta-HSD2,`
because the stem of 'regulates' was observed in an initial pattern, and also the tempus of this verb (present tense) was observed before – but not in this exact combination. Note that to make this example work, an inexact matching strategy is needed. At least, a vote for each considered position is necessary: in the example, two of the three layers would report a match for 'regulates', namely the stem and POS tag layer.

Figure 4.3.: Example for a multi-layer language pattern. IVBZ, verb in present tense; IVBD, simple past; *, arbitrary tokens/stems; definitions of POS tags used in this thesis are given in Appendix A.4.

Token layer	*	blocks | regulated	*
Word stem layer	*	block | regulat	*
POS layer	PROTEIN	IVBZ | IVBD	PROTEIN

4.1.3. Weighted multi–layer patterns

The pattern presented in Figure 4.3 represents observations (tokens, stems, POS tags seen at each position) without dealing with their actual occurrence frequencies. However, some tokens, stems, or POS tags are more likely to occur at a given position (of an arbitrary sentence) than others. To refine a pattern (again, to increase precision while maintaining recall), we assign weights to each observation that reflect frequencies for their respective occurrences. When a pattern resulted from two sentences, like in the example of the previous section, only observations that occurred in both sentences are treated as perfect matches if they also occur in the new sentence. We showed an initial example of weights in Figure 4.1 for the case of POS tag patterns. Here, we extend the idea of combining patterns to multiple layers, as shown in Figure 4.4.

Figure 4.4.: Multi–layer consensus pattern (CP) combining five patterns (1–5). Three layers, represented in the different rows of each pattern, are used in this example: tokens, stems, and POS tags, where entity class tags replace POS tags. The weights represent the distributions per position and layer.

1	*	/	*	blocked	*
	*	/	*	block	*
	PTN	SYM	PTN	IVBD	PTN
2	*	and	*	regulated	*
	*	and	*	regulat	*
	PTN	CC	PTN	IVBD	PTN
3	*	/	*	binds	*
	*	/	*	bind	*
	PTN	SYM	PTN	IVB	PTN
4	*	and	*	regulated	*
	*	and	*	regulat	*
	PTN	CC	PTN	IVBD	PTN
5	*	and	*	blocked	*
	*	and	*	block	*
	PTN	CC	PTN	IVBD	PTN
CP	$*_{5/5}$	and$_{3/5}$ | /$_{2/5}$	$*_{5/5}$	binds$_{1/5}$ | blocked$_{2/5}$ | regulated$_{2/5}$	$*_{5/5}$
	$*_{5/5}$	and$_{3/5}$ | /$_{2/5}$	$*_{5/5}$	bind$_{1/5}$ | block$_{2/5}$ | regulat$_{2/5}$	$*_{5/5}$
	PTN$_{5/5}$	CC$_{3/5}$ | SYM$_{2/5}$	PTN$_{5/5}$	IVB$_{1/5}$ | IVBD$_{4/5}$	PTN$_{5/5}$

4.2. Matching patterns against text

It is impossible to list all patterns that may capture a certain fact, as this would mean to list all possible variations of natural language. It would require to not only include all textual descriptions that can be found in publications already, but would imply to forecast all other deviations, minor or major. Thus, it is necessary to find a way to apply a limited number of patterns to arbitrary text and still cover as many variations as possible. Two main strategies have been applied to fulfill this requirement. The first uses regular expressions as pattern. Terms and gaps between terms get parameterized, allowing for repetitions and arbitrary new terms. Any deviation from the given sequence results in a mismatch of the whole expression. The second strategy is to have patterns with fixed terms and length, but apply an inexact matching against new text. Here, the matching technique accepts parameterizable deviations. In the next two sections, we will present regular expression matching and inexact matching using alignment, respectively.

4.2.1. Patterns as regular expressions

The classical approach to pattern matching is to encode each pattern as a regular expression. All regular expressions are then subsequently matched against new text and each reports a match or mismatch. The aforementioned frequencies of observations will be disregarded using a default formulation of regular expressions. Benefits from using regular expressions are their simple encoding and potentially reduced testing time, because mismatches in early stages of the test directly lead to non–match decisions. A usual way to implement regular expressions and matching them to new text is to use finite state automata (FSAs). In these automata, each position in a matched sentence is represented by a transition from one state to another. Transitions thus model the sequence of allowed terms in a sentence.

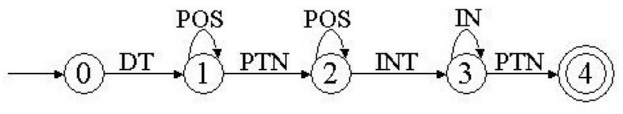
Definition 4.1 (Finite state automaton, FSA)**.** *A finite state automaton is given by the tuple* $(S, s_0, F, \Sigma, \delta)$*, where*

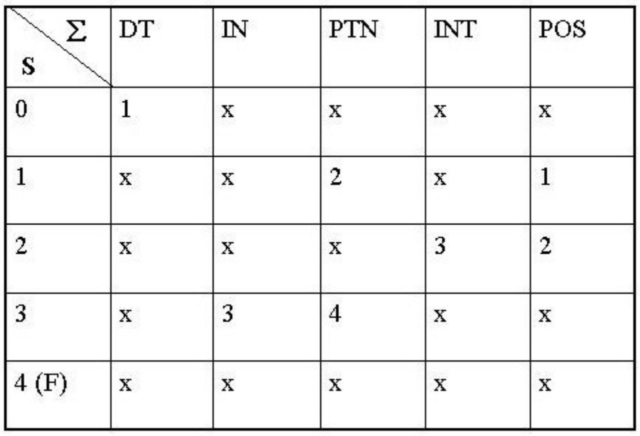
- S *is a non-empty set of states,*
- s_0 *is a start state,*
- F *is a set of end-states,* $F \subset S$
- Σ *is an input alphabet, a non-empty and finite set of symbols, and*
- δ *is a state transition function,* $\delta : S \times \Sigma \rightarrow S$*.*

[Definition adapted from Wikipedia.] An FSA has a dedicated start state, depicting the position before the first term in a sentence. Each allowed transition between two states, consuming an input symbol from the alphabet Σ, is stored in a matrix for δ. For each state in the FSA, this matrix contains all transitions possible to a new state using the next term in the sentence. Figure 4.5 shows an example, where columns represent the alphabet (in this case, POS tags, with additional tags for a protein and interaction-indicating verbs), and rows the states. Each cell denotes whether a transition from the current state along a particular POS tag to a new state is possible or not; a transition might also lead to the same state.

To build an FSA from a set of patterns, we start with one pattern and add states and transitions according to its sequence of terms. After the last term, we add a transition to an end state of the automaton. Iteratively, we go through all other examples and expand the automaton to finally fit all patterns. As we use wildcards in the patterns/regular expressions to depict optional additional words, the resulting FSA will be *non-deterministic*. That means that from some states, different transitions to different

Figure 4.5.: Finite state automaton for an input sentence of part-of-speech tags (top) and transition matrix (bottom). Σ = {DT, IN, INT, PTN, POS}; DT, determiner; IN, preposition; PTN, protein; INT, interaction verb; POS, arbitrary part-of-speech tag. Transition matrix for the states ∈ S (first column) and symbols ∈ Σ (first row), 'x' marks disallowed transitions. From state 1, a transition is possible either to state 2 along 'PTN', or back to state 1 consuming any POS tag. State 4 is the only final state.





S \ Σ	DT	IN	PTN	INT	POS
0	1	x	x	x	x
1	x	x	2	x	1
2	x	x	x	3	2
3	x	3	4	x	x
4 (F)	x	x	x	x	x

states are possible using the same symbol. An example is given in Figure 4.5, where from state 1, the automaton can consume a 'PTN' or an arbitrary part-of-speech tag, leading to state 2 or back to state 1, respectively. Note that in our case, all patterns have a fixed minimum sequence of transitions, none of which can be skipped; additional transitions are possible, as mentioned before,

Each non-deterministic FSA (NFA) can be converted into a deterministic FSA (DFA), as shown, for instance, by Salomaa and Yu [1997]. However, the set of resulting states in the DFA potentially is $(2^n - 1)$ when the NFA has n states. In case patterns are very diverse, it is possible to use a set of automata rather than encoding all regular expressions with a single automaton. Hopcroft et al. [2001] shows how an automaton can be optimized with respect to finding the minimal number of states so that the resulting FSA accepts exactly the same language. In contrast, we will show how to optimize automata with respect to precision and recall of the underlying patterns in Section 4.5. Such optimizations will alter the language that the resulting FSA accepts. We can restrict each automaton using an upper limit for states and/or transitions, or by using a fixed number of states/transitions each.

To parse complete sentences, we test all sub–phrases starting at the first word, the second word, the third word, and so on. If an FSA matches a sub–phrase of a sentence, then index positions define the interaction partners and its type.

4.2.2. Pattern matching with sentence alignment

Sequence alignment techniques are methods long established in traditional bioinformatics and play a key role to sequence analyses in molecular biology [Gusfield, 1997]. Their main purpose is to search large databases for homologues protein or DNA/RNA sequences and to perform sequence comparisons. An alignment shows how and how well two or more sequences fit together. For every pair of aligned positions, a score specific to the amino acids or nucleotides occurring at this position is pre–defined in a substitution matrix (see next section). This score contributes to an overall alignment score. An optimal alignment maximizes (or minimizes, depending on alignment strategy) this alignment score. Section 4.2.4 presents the standard solution to this problem using dynamic programming.

The alignment function *align* maps two sequences s and t to their aligned forms s' and t', a consensus sequence c, and an alignment score $a \in \mathbb{R}$. The alphabet $\mathcal{A}$ used by s and t is extended to $\mathcal{A}'$ for s' and t' and includes an additional character '-' that depicts a gap:

$$\text{align}(s,t) \mapsto s', t', c, a. \tag{4.1}$$

In later sections, we will need direct access to the consensus pattern of two sequences s and t (see Section 4.1.1), as computed via alignment. For simplicity, we thus define the function *consensus* as

$$\text{consensus}(s,t) \mapsto c. \tag{4.2}$$

Different algorithms for pairwise alignment have been proposed. Most commonly used are (variations of) the Smith-Waterman algorithm for global alignment, and the Needleman-Wunsch algorithm for local alignment. In global alignment, the goal is to match every position in one sequence to a position (or gap) in the other sequence. This strategy is most useful for sequences of similar length, when the goal is to find whether the two sequences have similar content. In contrast, local alignment assumes that one sequences will occur as a subsequence of the other, and large parts of the other sequence are thus not used. (Note that *subsequence* corresponds better with the *substring* concept in computer science). An alignment of sequences is recursively defined over the comparison of positions within the sequences. Comparing positions (that is, two nucleotides or residues in DNA or protein sequences, or two terms in sentences) requires the definition of a scoring function, depicting the similarity for every possible pair. This similarity score is detailed in the section on substitution matrices.

Alignment may also be used in computing the similarity between sentences, an approach we will detail in the following. When aligning a pattern against a sentence, we allow for differences between them in the same manner as sequence alignment introduces insertions, deletions, and mismatches between DNA or protein sequences. Furthermore, the decision whether or not a term of a pattern matches a term of a phrase

Figure 4.6.: Extract from our substitution matrix for part–of–speech tags. NN, noun; CC, conjunction, —, gap.

	—	PTN	NN	VBZ	VBD	IVBZ	IVBD	DT	CC
—	∞	**-10**	-6	-7	-7	-10	-10	**-1**	-10
PTN	**-10**	4	-8	-10	-10	-10	-10	-10	-10
NN	-6	-8	2	-3	-3	-5	-5	-2	-5
VBZ	-7	-10	-3	**2**	**1**	-1	-2	-2	-2
VBD	-7	-10	-3	**1**	**2**	-2	-1	-2	-2
IVBZ	-10	-10	-5	-1	-2	3	2	-10	-10
IVBD	-10	-10	-5	-2	-1	2	3	-10	-10
DT	**-1**	-10	-2	-2	-2	-10	-10	2	-2
CC	-10	-10	-5	-2	-2	-10	-10	-2	2

is not binary. Instead, the specific term pair is assigned a score by looking up a substitution matrix, which we will discuss in the next section. Thus, in contrast to regular expressions that either match or do not match a sentence, an alignment yields an overall score that provides a confidence measure. Setting or lowering thresholds for this confidence score influences the method towards higher precision or recall, respectively. Later in this section, we will explain how to derive consensus patterns from a set of initial patterns. Therefore, we use local alignment, assuming the initial patterns have roughly the same length. When matching patterns against a sentence, however, we would prefer global alignment, as the sentences are usually longer than the patterns, and we want to find the subsequence of the sentence best matching the pattern.

4.2.3. Substitution matrices

For every pair of terms of any annotation layer, a substitution matrix determines the costs for this particular substitution. It can be a perfect match, imperfect match (replacement of one term with another), or a deletion or insertion. Sequence alignment uses scoring matrices, such as BLOSUM [Henikoff and Henikoff, 1992], that define costs for every possible pair of amino acids (nucleotides), including gap penalties for inserted/deleted amino acids. In the same manner, we use substitution matrices that contain costs for term substitutions. A substitution matrix thus gives the costs for aligning two terms or a term and a gap. Figure 4.6 shows an example for a substitution matrix containing some important part–of–speech tags. The full matrix we use for our experiments is given in Appendix A.5.

These costs are combined with the observed frequencies of annotations of terms in all initial patterns, and stored for every position p in each consensus pattern, in the following presented for the single-layer case:

$$\text{freq}(i,p) = \frac{\#\text{occurrences of } i \text{ at } p}{\#\text{all observations at } p} . \tag{4.3}$$

For substituting a term j at the position p of the pattern P with a term i in a sentence, the cost is calculated as follows:

$$c(i, j, p) = \text{score}(i, j) * \text{penalty}(i, p), \tag{4.4}$$

where score(i, j) refers to the score from the scoring matrix. Penalty(i, p) penalizes a term i according to its observation frequency and is defined as

$$\text{pen}(i, p) = \begin{cases} \text{freq(i,p)}, & \text{if } i \in p \\ \min_{t \in p} \text{freq}(t, p), & \text{otherwise.} \end{cases} \tag{4.5}$$

This penalty includes a smoothing for cases where i was never observed at p in P. When aligning two consensus patterns, Equation 4.4 changes to

$$c(i, j, p_1, p_2) = \text{score}(i, j) * \text{pen}(i, p_2) * \text{pen}(j, p_1). \tag{4.6}$$

We estimated the substitution values in Figure 4.6 by trial–and–error. These values should reflect certain constraints we observe when matching patterns with arbitrary sentences for information extraction. Certain key positions cannot be replaced with other values; for instance, the alignment should always map proteins (PTN) to each other, and disallow replacing proteins with any other word category. As the initial NER to recognize protein names might fail sometimes, but the part–of–speech tagger may still finds such names as proper nouns (NNP) or foreign words (FW), the matrix should reflect these potential substitutions. Replacing interaction–indicating words in a pattern with other words from the sentence should be reflected in high costs, as such a substitution might alter the meaning of the sentence (compare "LRP6 and Wnt <u>interact</u> in the WNT–signaling pathway" and "LRP6 and Wnt <u>appear</u> in the WNT–signaling pathway"). An extension of substitution costs could include the observed entropy for each position in P; this would better reflect the variability than selecting the lowest frequency in Equation 4.5.

Deciding on the right costs for matches and replacements is certainly crucial to the overall approach. However, as Huang [2004] mentioned, it is sufficient to encode the overall tendencies, while searching for optimal values has only a minor impact on performance. Aforementioned crucial tendencies can also be observed in derived matrices such as BLOSUM62. Substitution costs range from -4 to 11, the majority between -2 and 1, with lower/higher values reflecting untypical replacements and enforced matches. Joachims [2003] showed an approach to learn the cost parameters for scoring an alignment of two sequences. Solving the optimization problem

$$D_\omega(s_1, s_2) = max_a[\omega^T \phi(s_1, s_2, a)] \tag{4.7}$$

then finds the optimal alignment a for two sequences s_1 and s_2 with the cost vector ω and ϕ the feature vector generated for s_1, s_2, and a. While this problem has an exponential number of constraints, Joachims presented an algorithm that returns an arbitrarily close approximation.

4.2.4. Dynamic programming to solve the alignment problem

First algorithms to efficiently solve the alignment problem using dynamic programming were proposed in the 1970s, see Gusfield [1997] for an overview. We also use this technique in our approach. Figure 4.7 shows a solution for the alignment of a pattern with a sentence given as a dynamic programming matrix. Figure 4.8 shows the alignment of two sentences with the pattern from Figure 4.1, where only POS tags were used.

Figure 4.7.: Dynamic programming matrix resulting from aligning a pattern (vertical) with a sentence (horizontal). Arrows indicate possible trace-back paths, the best spanning from 14.4 to 4. For simplicity, only POS tags are shown in the first row/column.

		NN	VBZ	DT	PTN	CC	PTN	IVBD	DT	PTN
	0	0	0	0	0	0	0	0	0	0
PTN	0	0	0	0	4	0	4	0	0	4
CC	0	0	0	0	0	5.6	0	0	0	0
PTN	0	0	0	0	4	0	9.6	0	0	0
IVBD	0	0	0	0	0	0	0	12.4	10.4	0
PTN	0	0	0	0	4	0	4	1.4	1.4	14.4

4.2.5. Weighted multi–layer alignment

If a term contains multiple annotations in different layers (see earlier in this chapter), the scoring function has to consider all these values(tokens, stems, tags) for comparing terms and combine them properly into one value. We calculate the multi–layer alignment like single–layer alignments shown previously. For each pair of terms, a scoring

Figure 4.8.: Alignment of the pattern CP, from Figure 4.1 with two sentences, S_1 and S_2. This example uses only the POS sequences, given in **bold**. For the fourth position (underlined), matching two 'IVBD' (S_1) yields a score of 3 according to Figure 4.6 and 'IVBD' has a frequency of $\frac{4}{5}$ in CP. In sentence S_2, 'IVBD' yields the score 2, multiplied with a penalty of $\frac{1}{5}$.

CP	$PTN_{5/5}$	$CC_{3/5}$ | $SYM_{2/5}$	$PTN_{5/5}$	$IVB_{1/5}$ | $IVBD_{4/5}$	$PTN_{5/5}$	
	hCDC20	and	hCDH1	bound	APC	
S_1	**PTN**	**CC**	**PTN**	**IVBD**	**PTN**	
	4 * 5/5	2 * 3/5	4 * 5/5	$\underline{3 * 4/5}$	4 * 5/5	Σ=15.6
	hCDC20	and	hCDH1	binding	APC	
S_2	**PTN**	**CC**	**PTN**	**IVBG**	**PTN**	
	4 * 5/5	2 * 3/5	4 * 5/5	$\underline{2 * 1/5}$	4 * 5/5	Σ=13.6

function determines the substitution score. The only difference is that this scoring function considers not only one substitution matrix, but several. For each layer, there is a dedicated substitution matrix. The overall score for each pair of terms is a linear combination of all scores for the different layers. We use weights w_l for each layer to control their respective influences on the overall score. The score s for substituting two terms i and j with each other are determined by an extension of Equation 4.4:

$$s(i,j,p) = \sum_{l \in \text{layers}} w_l * \text{score}_l(i,j) * \text{pen}_l(i,p), \tag{4.8}$$

where w_l refers to the weight of layer l, and $score_l(i,j)$ is the score in the respective scoring matrix for l. When we set the weight for the token and stem layers in the examples from Figure 4.9 to zero, we would exclude these layer completely. This would resemble the examples from above, where we use only POS tags to annotate a term. Figures 4.8 and 4.9 show examples for aligning the consensus pattern from Figures 4.1 and 4.4, respectively, with a given sentence. The first is an example for single–layer alignment, the second for multi–layer alignment.

To decide whether or not a sentence pattern matches a new sentence, each consensus pattern is assigned a minimum alignment score as a threshold. We calculate this based on the maximum alignment score possible for each respective pattern, that is, the highest score achievable for this pattern; essentially, this refers to normalizing over the length of the pattern. It is computed by aligning each consensus pattern with itself. From this maximum score, we subtract a (globally) parameterizable value to obtain the final threshold. This value can be adjusted either by exhaustive search or optimization procedures (see Section 4.5).

Figure 4.9.: Alignment of the consensus pattern CP from Figure 4.4 (top) with a new sentence (middle). The score for each layer (token, stem, and tag) is shown at the bottom. The overall score is 22.9 if we assume a weight of 1 for every layer. This alignment also contains an example for mismatches: 'bound' was not observed in the token layer and gets a score -1. The penalty of $\frac{1}{5}$ reflects the variability at this position, so the overall cost is weighted down to -$\frac{1}{5}$. In contrast, a mismatch at a position with a single observation would thus result in a cost of -1.

$*_{5/5}$	and$_{3/5}$ $\mid$ /$_{2/5}$	$*_{5/5}$	binds$_{1/5}$ $\mid$ blocked$_{2/5}$ $\mid$ regulated$_{2/5}$	$*_{5/5}$	
$*_{5/5}$	and$_{3/5}$ $\mid$ /$_{2/5}$	$*_{5/5}$	bind$_{1/5}$ $\mid$ block$_{2/5}$ $\mid$ regulat$_{2/5}$	$*_{5/5}$	
PTN$_{5/5}$	CC$_{3/5}$ $\mid$ SYM$_{2/5}$	PTN$_{5/5}$	IVB$_{1/5}$ $\mid$ IVBD$_{4/5}$	PTN$_{5/5}$	
hCDC20	and	hCDH1	bound	APC	
hCDC20	and	hCDH1	bind	APC	
PTN	CC	PTN	IVBD	PTN	
1 * 5/5	1 * 3/5	1 * 5/5	-1 * 1/5	1 * 5/5	3.5
1 * 5/5	1 * 3/5	1 * 5/5	1 * 1/5	1 * 5/5	3.8
4 * 5/5	2 * 3/5	4 * 5/5	3 * 4/5	4 * 5/5	15.6
					22.9

4.3. Learning patterns with multiple sentence alignment

As we have shown with the previous examples, combining similar patterns (that originate from partial sentences bearing the same meaning) helps to represent commonalities as well as variation among initial patterns. Finding variable terms across patterns and assigning scores based on occurrence frequencies increases the recall as compared to using only single patterns, while keeping the precision at a high level by restricting the potential noise. In this section we will describe how we can automatically derive consensus patterns given a set of initial patterns. There are two mandatory steps: *i)* identification of all patterns that are sufficiently similar to each other and that should thus be combined; *ii)* computation of the consensus pattern for each set of similar patterns. For the first step, we perform a clustering of the initial input set into groups of similar patterns. For the second step, we use multiple sentence alignment to compute the consensus pattern.

4.3.1. Clustering of initial patterns

Clustering of a data set relies on a similarity function, as discussed in Section 2.2.2 on page 25. This similarity function determines how similar (or dissimilar) two examples from the input sample are. In this work, we use the alignment score, as described in the previous sections, as the similarity function. In this scenario, we assume that the two input patterns are roughly of the same length, and we thus use a global alignment strategy. The alignment score at first yields only the score of the optimal alignment. Aligning two long patterns usually yields a higher score than aligning two short patterns, though the short patterns might be more similar to each other. To transform the alignment score into a similarity measure, we have to adjust for the variable length of patterns; also, because we use a complex substitution matrix (for POS tags), costs for matches/replacements vary a lot. We propose the following normalized scheme as the similarity function:

Definition 4.2 (Pairwise pattern similarity)**.** *The similarity of two patterns is defined by their pairwise alignment score, normalized with the maximum of the alignment score one of the patterns can potentially yield. This maximal alignment score results from an alignment of each pattern with itself.*

$$sim(p_1, p_2) = \frac{align(p_1, p_2)}{\max\{align(p_1, p_1), align(p_2, p_2)\}} \tag{4.9}$$

The strategy for clustering that we chose is an agglomerative, single-linkage clustering, which defines the similarity of two clusters by the minimum similarity between patterns p_1 and p_2 from clusters C_1 and C_2, respectively:

$$sim(C_1, C_2) = \min\{sim(p_1, p_2) : p_1 \in C_1, p_2 \in C_2\} \tag{4.10}$$

Clustering starts with each pattern implicitly assigned to its own cluster. Clusters are then subsequently joined using the single-linkage criterion. In a first step, we need to compute all pairwise similarities between all initial patterns; this leads to a pairwise alignment library (PAL). Sorting these pairwise similarities in a descending order yields a so–called guide list, with the most similar pairs of patterns on top, and the most dissimilar patterns on the bottom. Pairs occurring at the top of the list should be clustered together, while pairs at the bottom should not end up in the same cluster. The algorithm, adapted from Jain et al. [1999], is explained in pseudocode in Figure 4.10.

Let us consider the following example, where we have a set of patterns, {A,B,. . . ,G}. Assume pairwise alignment yields the similarities for the pairs of patterns as follows:

	A	B	C	D	E	F	G
A	1.0	1.0	0.9	0.8	0.5	0.4	0.2
B		1.0	0.9	0.8	0.5	0.4	0.2
C			1.0	1.0	0.6	0.3	0.1
D				1.0	0.6	0.3	0.1
E					1.0	0.8	0.6
F						1.0	0.7
G							1.0

The corresponding guide list would be

1.0	(A,B) (C,D)
0.9	(A,C) (B,C)
0.8	(A,D) (B,D) (E,F)
0.7	(F,G)
0.6	(C,E) (D,E) (E,G)
0.5	(A,E) (B,E)
0.4	(A,F) (B,F)
0.3	(C,F) (D,F)
0.2	(A,G) (B,G)
0.1	(C,G) (D,G)

Using this guide list as input, the clustering proceeds as follows. Going down the guide list, each pair either *i)* forms a new cluster (if neither of the patterns was assigned to a cluster before), *ii)* assigns one of the patterns to a cluster the other already belongs to, *iii)* combines two clusters into one, if both patterns belong to different clusters, or *iv)* is skipped because the similarity is below a certain clustering threshold (called guide-list cutoff). Note that this example differs slightly from the algorithm in Figure 4.10: we do not start at the level of one cluster per pattern. For the shown guide list, the example continues in the following way (for a guide-list cutoff of 0.7):

Figure 4.10.: Pseudocode for agglomerative, single-linkage clustering of initial patterns.

```
Input: set of annotated patterns, $P$;
       minimum pairwise similarity $t_s$;
       guide-list cut-off $t_g$;
  create matrix $m$: $|P| \times |P|$, $P_{i,j} = 0 \; \forall \; i, j$
  $\forall i$: create cluster $C_{p_i}$ with $p_i$ as only element
  for all $p_i, p_j \in P, i < j$ do
    if $sim(p_i, p_j) \geq t_s$ then
      $m_{i,j} = m_{j,i} = sim(p_i, p_j)$
    end if
  end for
  sort $m_{i,j}$ in descending order $\rightarrow$ guide-list $G$
  for all $g \in G, g \geq t_g$ do
    get the corresponding pair(s) $(p_i, p_j)$
    for all such pairs $(p_i, p_j)$ do
      if $C_{p_i} \neq C_{p_j}$ then
        join clusters $C_{p_i}, C_{p_j}$
      end if
    end for
  end for
Output: set of clusters $C$
        pairwise alignment library $m$
```

(A,B)	new cluster U
(C,D)	new cluster V
(A,C)	combine clusters U and V
(B,C)	already in same cluster, UV
(A,D)	already in same cluster, UV
(B,D)	already in same cluster, UV
(E,F)	new cluster W
(F,G)	assign F to W
Result:	clusters UV = {A,B,C,D} and W = {E,F,G}

4.3.2. Multiple sentence alignment

Once all clusters have been identified, all patterns in each cluster are combined to form a consensus pattern. We use a technique frequently used in bioinformatics called multiples sequence alignment (MSA), adopted to sentence alignment. For first algorithms proposed to solve MSAs, refer to Gusfield [1997]. For performing MSA on clusters of patterns, we iteratively go through all pairs of patterns in the cluster (again sorted by similarity). The alignment of the most similar pair in a cluster defines the initial consen-

Figure 4.11.: Pseudocode for multiple sentence alignment to identify consensus patterns.

```
Input: set of clusters $C$ containing similar patterns;
       pairwise alignment library: matrix $m$
  for all cluster $c_i \in C$ do
    get patterns $P \in c_i$
    get list of pattern pairs $pr_i$, sorted by pairwise similarity, from $m$
    consensus pattern $cp_i$ = align($pr_{1_1}, pr_{1_2}$)
    for all pairs $pr_j$, $j > 1$ do
      if $pr_{j_k} \notin cp_i$, $k \in \{1, 2\}$ then
        $cp_i$ = align($cp_i, pr_{j_k}$)
      end if
    end for
    add $cp_i$ to set of consensus patterns $CP$
  end for
Output: set of consensus patterns, $CP$
```

sus pattern. Progressively, all other pairs define which pattern(s) next to combine with the current consensus pattern. This is achieved by aligning the current consensus pattern with the next pattern from the initial set. The alignment itself thus defines the extended consensus pattern. Each round potentially introduces new deletions/insertions and replacements into the resulting consensus pattern; the frequencies of tokens, stems, and POS tags also get adjusted according to the observations in the next pattern. Figure 4.12 shows the result of an MSA with five initial patterns. The introduction of gaps is an important feature of our approach, because it allows skipping terms at a low cost (in particular, we want to achieve a low cost for replacing certain types of terms, such as determiners, adjectives, and adverbs). The pseudocode for this procedure is shown in Figure 4.11, in part derived from standard algorithms for multiple sequence alignment; see, e.g., Hirosawa et al. [1995].

4.4. Pattern generation by pairwise alignment

Apart from multiple sentence alignment, another strategy to generate consensus patterns from an initial set of patterns is based on pairwise alignment. The method we use is comparable to the one described in Huang et al. [2004] and uses annotated examples. Subsequent pairwise alignment of sentences ultimately yields patterns with as much support in the training data as possible. The pattern generating algorithm iterates over all pairs of sentences and calculates the best alignment for each pair. Each consensus sentence from the optimal alignment of these two sentences then forms a pattern. We count the occurrences of all such patterns in the training data to calculate the support for each pattern. The maximum number of patterns for n sentences is $n(n-1)/2$, but in

Figure 4.12.: Consensus pattern (CP) resulting from MSA of five initial patterns (1–5). The consensus pattern reflects the occurrences of adverbs (RB) and determiners (DT), though they were not observed in all initial patterns. —: potential gap position.

	*		blocked		*
1	*		block		*
	PTN		IVBD		PTN
	*		regulated	the	*
2	*		regulat	the	*
	PTN		IVBD	DT	PTN
	*	strongly	binds		*
3	*	strong	bind		*
	PTN	RB	IVB		PTN
	*		regulated		*
4	*		regulat		*
	PTN		IVBD		PTN
	*		blocked	the	*
5	*		block	the	*
	PTN		IVBD	DT	PTN
	$*_{5/5}$	$—_{4/5} \mid \text{strongly}_{1/5}$	$\text{binds}_{1/5} \mid \text{blocked}_{2/5} \mid \text{regulated}_{2/5}$	$—_{3/5} \mid \text{the}_{2/5}$	$*_{5/5}$
CP	$*_{5/5}$	$—_{4/5} \mid \text{strong}_{1/5}$	$\text{bind}_{1/5} \mid \text{block}_{2/5} \mid \text{regulat}_{2/5}$	$—_{3/5} \mid \text{the}_{2/5}$	$*_{5/5}$
	$\text{PTN}_{5/5}$	$—_{4/5} \mid \text{RB}_{1/5}$	$\text{IVB}_{1/5} \mid \text{IVBD}_{4/5}$	$—_{3/5} \mid \text{DT}_{2/5}$	$\text{PTN}_{5/5}$

practice not all are generated, since a set of different alignments can lead to the same consensus sequence. On the other hand, in case there were multiple optimal alignments, all alignments were taken to form (different and/or identical) patterns. The algorithm in Figure 4.13 shows the pattern generating algorithm as pseudo-code; see Plake et al. [2005]. To distinguish patterns from each other, is is also necessary to store the positions of agent, target, and interaction type with each consensus, in addition to the consensus sentence. Two consensus sentences might contain different relations.

This method searches for the most similar sentences, and subsumes these with a more general expression, if necessary. As we take alignments for this step, we use the respective consensus sentence as generalized sentence. This general expression is then added to a set. This set functions as a model to explain the training data, and can also be seen as a set of patterns. Only patterns with at least two proteins and one interaction-indicating word (specific verb or noun) are included in the final set. From the final set, all patterns below a minimum support (that is, the number of aligned sentence pairs producing this pattern) are removed.

4.5. Optimization using genetic algorithms

In cases where a set of patterns (regular expressions, single patterns, or consensus patterns) was computed from a small initial sample, we need ways to improve it. For small

Figure 4.13.: Pseudocode of the pattern generating algorithm. It loops over all pairs of initial patterns and builds their consensus pattern using alignment. Patterns with low support in the training data get removed afterwards.

```
Input: set of annotated patterns, $S$;
       threshold $d$
  for all $s_i, s_j \in S, i < j$ do
    $c_{i,j} = \text{consensus}(s_i, s_j)$
    $K = (pos_a, pos_b, pos_i)$ = positions of
      partners $a, b$, and interaction-word $i$ in $c_{i,j}$
    $p = (c_{i,j}, K)$
    if $p \notin P$ then
      add $p$ to $P$; $occ_p = 1$
    else
      $occ_p$++
    end if
  end for
  remove all $p \in P$ with $occ_p < d$
Output: set of patterns, $P$
```

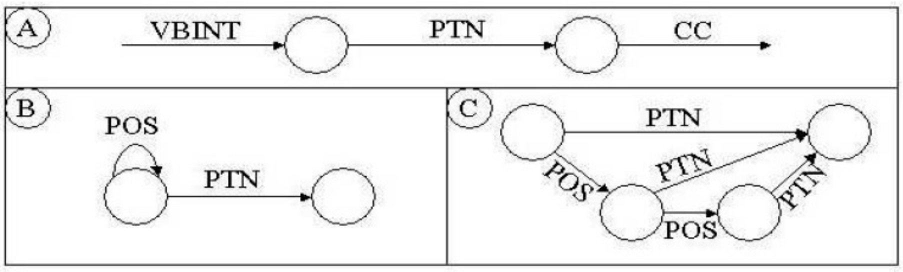
samples, this most often means to generalize the patterns, as they over-fit on this sample and do not recognize previously unseen examples. When given a training set, it is possible to use this set for optimization. We can learn the patterns from one part, and optimize them on the other part. Optimization often means tuning to a specific quality measure, to enhance the patterns' precision and recall, or the balance between both (see Section 2.2). When a training set was manually annotated, this results in tuning the set to the specific annotator's opinion.

We performed an optimization of patterns represented as regular expressions (encoded in finite state automata). Optimization included, for instance, finding the optimal size of word gaps in between fixed positions. The strategy was to optimize the set of automata on a training sample using a genetic algorithm to achieve optimal results. We introduced word gaps of variable length between states (that is, between positions in a sentence). The FSA thus contains not only sentences fitting phrases exactly, but allows for insertions (for example, short subordinate clauses or expressions in brackets); see Figure 4.14 for examples. Intuitively, a more "strict" FSA (no word gaps) would yield more precise results, while a "loose" FSA (word gaps of infinite length) would gain a higher recall, by fitting more sentences.

We encoded the transition matrix of an FSA in a single array. The value in each cell was transformed into a binary representation, depicting the index of a new state, when the transition using a particular POS tag was possible, or zero otherwise. In addition, we added a bit to this cell value. Whenever this bit was set to one, the transition led not only to a next state, but in addition, this state was an end-state. Table 4.1 shows an example for the binary representation of state 3 from Figure 4.5 on page 75. In addition, the positions for agent(s), target(s), and interaction type(s) in the sentences had to be encoded. Taken together, this led to a binary representation of FSAs, which we refer to as a *genome*. In such a genome, all positions for the states, and all positions for transitions and end-states were fixed. Translations from a matrix to a genome and back are unambiguous.

The task now was to find one or more good FSAs encoded by such genomes. We first started with a set of random genomes, representing a *population*, where each individual encoded an FSA with six states; the number of states was evaluated in further experiments; data not shown. Initially, most of the individual members of a population would encode a valid, but certainly a "bad performing" matrix. We optimized this population on the training sample using a genetic algorithm, see Plake et al. [2005]. For this approach, we could use either precision, recall, or f-measure as a *fitness function* to measure the performance of every individual FSA in the population. Choosing the best 25% of a population to form the basis for a new generation, we filled the missing 75% using *recombinations*, *cross-overs*, and *mutations* of the preceding generation. Ultimately, this led to an FSA (the "best" genome in the population) covering at least some positive examples from the training sample (a positive example contains at least one interaction). As soon as no further improvements could be achieved, we removed all training sentences covered by this FSA. On the remaining sample, we started over with a completely new random population. We followed this *divide-and-conquer strategy*, until no positive sentences were left in the training sample, or a certain number of finite state automata was found.

Figure 4.14.: Finite state automata with word gaps. (A) shows the original transitions between two states, with a protein tag in the middle. In (B), the FSA is extended with an additional transition from the left state, using any POS tag, to the same state. This transition could be used multiple times. (C) shows an extension for a maximum of two, instead of multiple, optional POS tags.



POS tag	new state	end-state	possible transition
DT	000	0	"no transition possible"
IN	011	0	"possible to state 3"
PTN	100	1	"to state 4 $\Rightarrow$ end-state"
INT	000	0	"no transition possible"
POS	000	0	"no transition possible"

Table 4.1.: Binary representation of state 3 from the matrix in Figure 4.5. The table shows all possible transitions, and to which new state each transition leads. Using 'PTN', the transition from 3 to 4 would lead to an end-state.

5. Applications and Evaluation

As we have shown in the previous chapter, language patterns can be applied to information extraction tasks. A pattern encoded to capture a particular kind of information —protein–protein interactions, gene–disease associations, etc.— can be compared with an arbitrary sentence. If it matches, one may assume that the sentence contains the same kind of information — the meaning can be transferred from the pattern to the sentence. The patterns we described in the last chapter contain information on the interactors, their dependency, the (sub)type of the interaction, plus the exact evidence and positions of all objects. Thus, comparing pattern to sentence allows to extract the same information from a matching sentence. In this chapter, we will describe and evaluate our approaches for generating sets of patterns from previously unlabeled data and matching them against new text. We focus on the application of identifying protein–protein interactions, and discuss results of our and related approaches for this task.

5.1. Collecting a large pattern sample

As we have mentioned in the previous chapter, the quality of a pattern set is highly dependent on the size of the initial sample. Patterns generated from manually collected sets of sentences describing interactions, for example, presented by Hao et al. [2005] and Berleant et al. [2003], yield low recall values when applied to external texts [Hakenberg et al., 2005b]. Blaschke and Valencia [2002] showed that the same is true for manually defined patterns. In most cases, patterns are over-fitted towards the small sample they originate from, and are not general enough to recognize previously unseen examples (see our results on the LLL corpus, Section 5.3). Therefore, we developed a new strategy, in which we start with a large set of precise initial patterns. As it is infeasible to create such large sets manually, we show how to collect such sets automatically in Sections 5.1.1 and 5.1.2.

5.1.1. Curated data bases to find examples

One idea we pursue to automatically collect large samples for extracting patterns is based on information contained in hand-curated databases. Databases that store precise information on relations between objects can be exploited for this approach; for instance, protein-protein interactions are stored in IntAct and DIP [Hermjakob et al., 2004, Xenarios et al., 2001]; gene ontology annotations of proteins are contained in GOA [Camon et al., 2004]. All such databases map objects to unique entries (in the same or other databases): IntAct has UniProt IDs for all proteins, GOA has UniProt IDs and GO term IDs. Our assumption is that when we take a large number of such pairs and search for their occurrences in a large enough text corpus, we will come up with a large sample of sentences, each of which are likely to explain the relation between the objects. This approach relies on a properly solved named entity normalization (EMN; see Chapter 3) to successfully identify objects in texts. A similar strategy would be to search for all pairs of synonyms of proteins known to interact, where we would essentially skip the disambiguation component. Even if EMN produces false positives (single objects), it is less likely that a pair of false positives predicted in a single sentence also maps to a pair in the database. To increase the probability that an extracted sentence indeed describes a relation of the requested type, one can improve the search with further constraints. For each relation, one could require the occurrence of specific words typically used to describe such relations. For protein–protein interactions, this would be words indicating interactions, as shown in the previous chapter. However, it is clear that restricting the sample in this way might result in an over–specification of the resulting patterns.

5.1.2. Protein–protein interaction patterns using IntAct

The IntAct database [Hermjakob et al., 2004] provides data and analysis tools for protein interactions. Data are derived manually from literature or come from user submissions. IntAct contains 68,269 interactions identified in 3585 distinct experiments (as of early 2006). Most data come from large-scale data sets for few organisms, for instance *Saccharomyces cerevisiae* [Ho et al., 2002], *Drosophila melanogaster* [Giot et al., 2003], and *Homo sapiens* [Bouwmeester et al., 2004] (containing ≈30,000, 20,000, and 1700 interactions, respectively). More than 75% of interaction identifications stem from two hybrid methods (two hybrid, array, pooling, fragment pooling)[1], which need not necessarily occur in vivo (also applies to other identification methods).

From IntAct, we extracted all pairs of interacting proteins. Using the dictionary-based named entity recognition (and partial identification) described in Chapter 3, we scanned all of MEDLINE for any co–occurrences of each pair in a single sentence. This NER with partial identification assigned UniProt IDs to each recognized protein name. We treated protein names with multiple candidate IDs (same protein shared, for instance, among different species) as hits, not further disambiguating the recognized name. Whenever a sentence contained a pair of interacting proteins, we considered it for further use.

We restricted all resulting sentences further by requiring an interaction–indicating word (see Appendix A.3). From each sentence, we extracted the core–phrase that mentions both proteins and the interaction word, plus a parameterizable boundary to the left and right. This restriction should reduce the phrase to the actual statement (description of a protein–protein interaction).

The generation of a pattern set from these initial phrases followed the strategy we discussed in Section 4.3. All initial phrases were clustered using their pairwise sentence alignment score as a similarity function. On each cluster, we performed a multi-

[1]See http://www.ebi.ac.uk/intact/statisticView/do/statistics

ple sentence alignment to identify representative consensus patterns. We distinguished between initial phrases that were restricted using interaction–indicating words depending on their respective general POS category: nouns, verbs, adverbs, and adjectives.

5.2. Assigning attributes to relations

The assignment of attributes to predicted relations provides additional important information to the user. Attributes refer to quantifications and qualifications of each relation, conditions under which relations were observed, and so on. For protein-protein interactions, it is interesting to know about the cellular location where an interaction took place, the experimental methods used to identify the interaction, the organism used in the experiment, and the duration (transient or permanent) of spatial bindings.

To extract attributes for every relation, there are mainly two possibilities. Patterns can include additional information themselves, or rule-based approaches are used. Encoded in a pattern, the occurrence of an attribute could be exactly specified (position in the pattern), or it could be given relatively to certain other fixed terms. In case of protein-protein interactions, relative positions could refer to the interaction type: an adjective occurring before an interaction noun specifies this type further ("strong bond"); adjective preceding a protein specify this protein further, for example when mentioning the organism ("human protein Cdk3").

Rule-based approaches, on the other hand, encode information on where to look for additional information if a pattern fits the current text. We observed that the organism for which a certain relation was studied often is mentioned in the title of a publication, or the very first sentence of its abstract [Schmeier, 2005].

5.2.1. Predicting directed relations

For many relations, it is possible to identify the direction of an interaction, that is, the exact agent-target dependency. For protein-protein interactions, this is not always implicitly given, like for drug-target associations. For PPI, we use a small set of rules that have been compiled manually and can be applied to other tasks. For some interaction types, we assign a bi-directed relation: bonds, attachments, interactions, formations of complexes, and so on. For others, like phosphorylations, inhibitions, or cleavages, we determine the relation using information on conjugation of the verb and possible prepositions. For example, a phrase like

`'(PROTEIN1) (is) (VERB: passive) (by) (PROTEIN2)'`

suggests 'PROTEIN2' as agent, while

'(PROTEIN1) (VERB: 3^{rd} pers. sing.) (PROTEIN2)'

suggests 'PROTEIN1'.

5.3. Mining protein-protein interactions

For the task of protein–protein interaction extraction from text, we evaluated the performance of three approaches: hand–crafted regular-expression patterns, optimized regular-expression patterns, and alignment-based patterns learned from large, unannotated samples. For this purpose, we used six benchmark set:

- SPIES, consisting of 893 sentences [Hao et al., 2005];
- 1000 sentences derived from the BioCreative I task 1A data; this set contains annotations for gene/protein names, and we added annotations for protein–protein interactions [Plake et al., 2005];
- the LLL'05 challenge corpus, which contains sentences describing genic interactions, i.e., gene–gene, gene–protein, and protein–protein interactions [Nédellec, 2005];
- IEPA, a set of 400 abstracts [Berleant et al., 2003];
- a set of abstracts and full texts (where available) that we derived from known interactions in the DIP database;
- the BioCreative 2 challenge corpus, consisting of full text articles [Krallinger et al., 2007].

Note that for the first three benchmark sets, exact positions of the interacting proteins/genes are known. The other three sets annotate interactions per text, that is, although the (standardized) names of all involved proteins are known, the evidence for the interaction can be anywhere in the text. Appendix A.1 shows detailed on each corpus. We will present results for each of the three approaches in the following.

5.3.1. Hand–crafted patterns

Our first approach to extract protein-protein interactions from text was based on manually designed patterns, as described in Section 4.2.1 and 4.4. Table 5.1 shows our initial set of 22 hand–crafted patterns. These were derived from example sentences occurring in the BioCreative I task 1A data set and are partly based on Blaschke and Valencia [2002]. Performances of this pattern set, taken as–is or after optimization (see next section) are given in Table 5.2.

5.3.2. Optimized patterns

Optimizing a pattern set

We optimized hand-crafted patterns using a genetic algorithm on Mealy finite state automata. All potential parameters were encoded as a genome (as described in Section 4.5 on page 85). Table 5.1 shows all these parameters for 22 patterns in more detail (up to six variables per pattern). As fitness functions, we chose either the precision, recall, or F1-measure, as achieved on a test set. The changes of results for these functions depending on evolution are shown in Figure 5.1. We found that an optimization using genetic algorithms could improve the overall performance: precision by up to 16%, recall 5%, F1-measure 8%, respectively, depending on the optimization strategy.

(1) **Protein**$_A$ Word*$_{1.1}$ **IVerb** Word*$_{1.2}$ **Protein**$_B$

(2) **Protein**$_A$ Word*$_{2.1}$ **IVerb** Word*$_{2.2}$ `by` Word*$_{2.3}$ **Protein**$_B$

(3) **IVerb** `of` Word*$_{3.1}$ **Protein**$_A$ Word*$_{3.2}$ `by` Word*$_{3.3}$ **Protein**$_B$

(4) **IVerb** `of` Word*$_{4.1}$ **Protein**$_A$ Word*$_{4.2}$ `to` Word*$_{4.3}$ **Protein**$_B$

(5) **INoun** `of` Word*$_{5.1}$ **Protein**$_A$ Word*$_{5.2}$ `[by|through]` Word*$_{5.3}$ **Protein**$_B$

(6) **INoun** `of` Word*$_{6.1}$ **Protein**$_A$ Word*$_{6.2}$ `[with|to|on]` Word*$_{6.3}$ **Protein**$_B$

(7) **INoun** `between` Word*$_{7.1}$ **Protein**$_A$ Word*$_{7.2}$ `and` Word*$_{7.3}$ **Protein**$_B$

(8) **Protein**$_A$ Word*$_{8.1}$ **Protein**$_B$ Word*$_{8.2}$ `[`*Complex*`]`

(9) **Protein**$_A$ Word*$_{9.1}$ **Protein**$_B$ Word*$_{9.2}$ **INoun**

(10) `complex formed between` Word*$_{10.1}$ **Protein**$_A$ Word*$_{10.2}$ `and` Word*$_{10.3}$ **Protein**$_B$

(11) `[complex|complexes] of` Word*$_{11.1}$ **Protein**$_A$ Word*$_{11.2}$ `and` Word*$_{11.3}$ **Protein**$_B$

(12) **Protein**$_A$ Word*$_{12.1}$ `[form|forms]` Word*$_{12.2}$ `complex with` Word*$_{12.3}$ **Protein**$_B$

(13) `[complex|complexes] containing` Word*$_{13.1}$ **Protein**$_A$ Word*$_{12.2}$ `and` Word*$_{13.3}$ **Protein**$_B$

(14) **Protein**$_A$ Word*$_{14.1}$ **Protein**$_B$ Word*$_{14.2}$ `[form|formed]` Word*$_{14.3}$ **Protein**$_C$

(15) **Protein**$_A$ Word*$_{15.1}$ **Protein**$_B$ Word*$_{15.2}$ **IVerb** Word*$_{15.3}$ `with each other`

(16) **Protein**$_A$ Word*$_{16.1}$ `[inhibitor|repressor] of` Word*$_{16.2}$ **Protein**$_B$

(17) **Protein**$_A$ Word*$_{17.1}$ **IVerb** Word*$_{17.2}$ `but not` Word*$_{17.3}$ **Protein**$_B$

(18) **Protein**$_A$ Word*$_{18.1}$ `cannot` Word*$_{18.2}$ **IVerb** Word*$_{18.3}$ **Protein**$_B$

(19) **Protein**$_A$ Word*$_{19.1}$ `[does|did|was] not` Word*$_{19.2}$ **IVerb** Word*$_{19.3}$ **Protein**$_B$

(20) **Protein**$_A$ Word*$_{20.1}$ `not` Word*$_{20.2}$ **IVerb** Word*$_{20.3}$ `by` Word*$_{20.4}$ **Protein**$_B$

(21) **Protein**$_A$ Word*$_{21.1}$ `not required for` Word*$_{21.2}$ **IVerb** Word*$_{21.3}$ **Protein**$_B$

(22) **Protein**$_A$ Word*$_{22.1}$ `failed to` Word*$_{22.2}$ **IVerb** Word*$_{22.3}$ **Protein**$_B$

Table 5.1.: Set of hand-crafted patterns. Markup: **fixed anchor entities**, `fixed constraint words`. Word*$_{18.2}$ refers to the maximum length of the second word gap in pattern 18. `[`*Complex*`]` in pattern (8) is short for: `[complex|complexes|dimer|heterodimer|homodimer]`

All results are shown in Table 5.2. Additionally, the search for optimal parameters with genetic algorithms proved to be very fast on this small pattern set. Individual results for the BC1-PPI (BioCreAtIvE), DIP, and LLL'05 benchmarks are discussed in detail in the following paragraphs.

BioCreAtIvE — evaluation on single sentences

Results on the BioCreative 1 PPI corpus (BC1-PPI) are shown in Table 5.2, comparing hand-crafted to optimized patterns; results were obtained using a 10-fold cross-validation. The best averaged precision of 75% was achieved at a recall level of 21%, ranging from 63% to 84% in the 10 runs (recall: 16–32%). Compared to the manual refinement of patterns, the genetic algorithm was able to find a solution which yields about 16% higher precision. The best recall found was 51% (42–58%), about 5% better than a manual refined pattern set. For this configuration, the precision ranged from 37–52%, with an average of 43%. When the fitness was determined by F1–measure, the optimization yielded 52% in the ultimate parameter setting, a manual definition performs about 8% worse. The optimal pattern set could be found after 21–160 steps.

The evaluation on BioCreAtIvE set included the impact of named entity recognition. When we excluded the NER from the evaluation, we took the answer key as a 'perfect' NER system. We compared this approach to an ensemble including our own protein

Data set	Fitness function	P	R	F-m
BioCreAtIvE	optimization, f-measure	60%	46%	52%
perfect NER	optimization, precision	75%	21%	33%
	optimization, recall	43%	51%	47%
BioCreAtIvE	optimization, f-measure	43%	36%	39%
imperfect NER	optimization, precision	58%	19%	29%
	optimization, recall	30%	39%	34%
BioCreAtIvE	no optimization, word gaps: 0-2	59%	26%	36%
perfect NER	no optimization, word gaps: 0-5	48%	40%	44%
	no optimization, word gaps: 0-10	40%	46%	43%
	no optimization, word gaps: 0-15	37%	46%	41%
BioCreAtIvE	no optimization, word gaps: 0-5	33%	29%	30%
imperfect NER	no optimization, word gaps: 0-10	26%	33%	29%
DIP, all data	optimization, recall	n/a	50%	n/a
DIP, abstracts	optimization, recall	n/a	23%	n/a
DIP, all data	no optimization, word gaps: 0-2	n/a	40%	n/a
DIP, all data	no optimization, word gaps: 0-10	n/a	52%	n/a
DIP, all data	no optimization, word gaps: 0-15	n/a	53%	n/a
DIP, abstracts	no optimization, word gaps: 0-2	n/a	14%	n/a
DIP, abstracts	no optimization, word gaps: 0-10	n/a	25%	n/a
DIP, abstracts	no optimization, word gaps: 0-15	n/a	27%	n/a

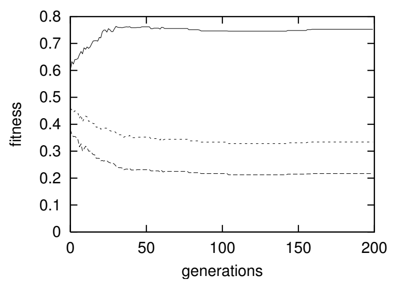
Table 5.2.: Results for hand-crafted patterns versus optimized patterns; for optimization, the fitness-function for the genetic algorithm is also given. Two benchmarks are used: BioCreative and DIP. Optimization on different fitness functions (f-measure, precision, recall) is also given. The best results achieved with and without optimization are underlined.

name recognition method (see Section 3.1). When the system included the recognition of protein names, the precision dropped about 17% (14% recall, 13% F1-measure).

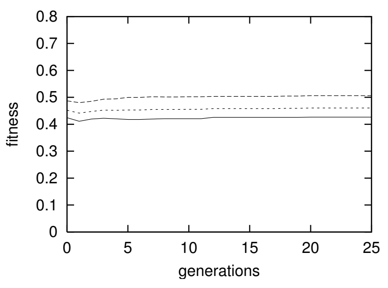
DIP — full texts versus abstracts

For the evaluation on the DIP corpus, we used the parameter settings evolved during optimization on the BioCreAtIvE corpus. In the DIP database, information for interactions are curated. These information contain references to publications that discuss these interactions. The references do not contain the exact textual evidence, however. Starting from one DIP interaction, we can determine only that a publication discusses this interaction somewhere. We cannot determine where exactly the interaction appears in the text, and we can not know about additional interactions that might be discussed somewhere else in the same text. Training on DIP full papers would thus include many interactions not marked as positive examples, and would lead to a very conservative model. Correspondingly, we aim at a high recall for this test scenario, and chose recall as the fitness function, because we cannot evaluate the precision.

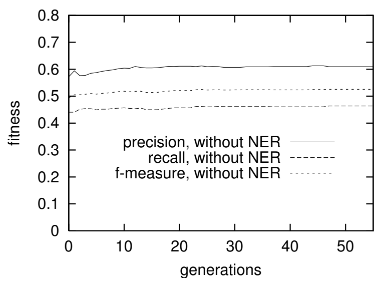
Results are shown in Figure 5.2, with and without optimization of hand-crafted patterns. The recall on full papers regarding all 297 interactions was 50%. When using



(a) Precision



(b) Recall



(c) F-measure

Figure 5.1.: Evolution progress for the three fitness-functions used: (a) precision, (b) recall, and (c) F1-measure, all without NER. Each figures show the other two corresponding measures as well, though these had no influence on the respective fitness.

manual refined patterns, we could achieve a recall of 53%. When we took only the PUBMED abstracts of references for the evaluation, only 23% of all 297 interactions could be detected. For one, more interactions are discussed somewhere in the full text than in the abstract of a paper. On the other hand, descriptions appearing in the abstract are repeated in the full text, sometimes more than once. If evidences in the abstract are not covered by patterns, the evidences in the full text –expressed in different ways– might be detected instead. The detection of one of these appearances is sufficient to extract an interaction correctly. We noticed that the genetic algorithm could not improve the performance in both cases, it yielded about 4% less recall.

LLL'05 corpus

For evaluation on the LLL'05 corpus, we chose different generation and matching strategies: sentence alignment and finite state automata with different parameters. The patterns were generated from two different corpora (or their combination): 1000 sentences derived from BioCreative I Task 1a and the training sample provided with the LLL'05 data set. In Table 5.3, these corpora are named C_{1000} (BioCreative), C_{55} (LLL training), and C_{86} (LLL test), respectively. The LLL'05 had two different type of annotations, one with, one without using co-references. All results result from training/testing without incorporating the co-references. In this category, our method proved best among three systems. Including annotated co-references, the best system achieved an F1-measure of 52.4%, which was 0.6% higher than our best result.

Method	Corpus	Precision	Recall	F1 (%)
Alignment	C_{1000} C_{55}	57.1	7.8	13.7
Alignment	C_{55} C_{86}	63.6	8.1	14.4
Mealy (2 FSAs)	C_{1000} C_{55}	64.3	17.5	27.5
Mealy (2 FSAs)	C_{1000} C_{86}	42.9	9.2	15.2
Mealy (4 FSAs)	C_{55} C_{86}	28.1	31.4	29.6
Mealy (5 FSAs)	$C_{1000+55}$ C_{86}	50.0	53.8	51.8

Table 5.3.: Results on the LLL corpus using alignment or regular expressions. Upper corpus in each row was taken for training, lower for test; C_{1000}: corpus of 1000 sentences; C_{55}: genic_interaction_data; C_{86}: basic_test_data.

Our method often predicted the interaction partners correctly, but an incorrect agent-target dependency. This could have been solved exploiting the provided co-references. Syntactic relations could help to generate more specific patterns that contain whole phrases instead of single tags ("expression of rsfA"). For this specific phrases, certain positions could even be restricted to particular tokens, instead of general POS tags. Combinations of verbs and prepositions, for instance, contain linguistic information clearly stating the exact role of a gene/protein appearing before or after this verb phrase ("was phosphorylated by"). Grouping particular interaction types together (for example, "phosphorylation" and "methylation" refer to the creation of bonds, while "reduction" and "repression" imply an inactivation) and restricting patterns to these particular types might further help to exploit nomenclature usages.

We saw that the performance of our approach is clearly dependent on the size of the training set. Our system was able to detect only interactions for which it encountered a quite similar example in the training data. Using only 55 sentences for generating patterns proved insufficient, as many relations in the test set did not resemble any of these. The training corpus of 1000 sentences from BioCreative I Task 1a contained 256 different protein-protein interactions, but these were in general quite dissimilar from the genic interactions in the LLL'05 data. For instance, our training sample did not include any descriptions of regulon family memberships. Most of our example sentences describe actions and bindings, and our system performed best in these categories (see Table 5.4).

	action	bind	regulon	nothing	all
TP	19	7	2	0	28
FN	17 (36)	5 (12)	2 (4)	0 (0)	24 (52)
FP	10 (29)	4 (11)	1 (3)	13 (13)	28 (56)

Table 5.4.: Performance on different types of interactions contained in the LLL'05 corpus. Numbers in brackets give the total number of existing interactions (row FN), and predicted interactions (row FP), respectively.

5.3.3. Patterns learned from large samples

Results for protein-protein interactions using the IntAct Sample

Information on the size of the corpus and pattern sets derived from MEDLINE and IntAct using the method described in Section 5.1 are presented in Table 5.7. Out of almost 120,000 sentences that contained at least one pair of interacting proteins according to IntAct, we filtered different sets of initial patterns, dependent on word boundaries and interaction types. Not all of the 120,000 sentences contained one of the required interaction types (see Appendix A.3). Some sentences contained multiple interactions, in most cases in different core-phrases, so that they were represented multiple times in the initial phrases. We show the support of initial phrases in Figure 5.2. Table 5.5 shows the initial phrases that have the highest support in the sample. We discuss some core-phrases with uncertain veracity in Table 5.6. We clustered all phrases that shared the same POS class of the interaction type (noun, verb, adjective), and computed consensus patterns on each subset. Ultimately, this lead to more than 9,500 consensus patterns.

SPIES corpus

We performed our initial experiments on the SPIES corpus [Hao et al., 2005], described in the Appendix A. Table 5.8 shows the performance of various methods in detail. The experiments were done with different sets of initial patterns: different word boundaries for the core-phrases, different guide-list cut-offs, and different maximal lengths of phrases in a cluster. All experiments included the full set of IntAct pairs and resulting phrases from MEDLINE. To tune the system towards a potential bias in the annotations in the SPIES corpus, we additionally ran experiments were we sampled 70% of the sentences for optimization and used the remaining 30% for the test (10-fold cross-validated). For the optimization –either towards precision or recall– we adjusted the thresholds of the minimum alignment scores using the training set.

BioCreative II IPS corpus

BioCreative (Critical Assessment of Information Extraction systems in Biology) is a community effort aimed at "evaluating text mining and information extraction sys-

Supp.	Initial phrase
246	*/*/PTN receptor/receptor/n:e */*/PTN
55	*/*/PTN -dependent/-dependent/adj:b */*/PTN
43	*/*/PTN binding/binding/n:e to/to/prep */*/PTN
40	*/*/PTN -mediated/-mediated/adj:b */*/PTN
37	*/*/PTN interacts/interact/v:G3e with/with/prep */*/PTN
31	*/*/PTN phosphorylation/phosphorylation/n:e and/and/cnj:K */*/PTN
31	*/*/PTN binds/bind/v:G3e to/to/prep */*/PTN
27	*/*/PTN bound/bind/v:P to/to/prep */*/PTN
27	*/*/PTN expression/expression/n:e on/on/prep */*/PTN
25	*/*/PTN associated/associate/v:P with/with/prep */*/PTN
23	*/*/PTN inhibitor/inhibitor/n:e */*/PTN
22	*/*/PTN expression/expression/n:e and/and/cnj:K */*/PTN
21	*/*/PTN receptor/receptor/n:e ,/,/pun */*/PTN
19	*/*/PTN activity/activity/n:e and/and/cnj:K */*/PTN
18	*/*/PTN binds/bind/v:G3e */*/PTN
18	*/*/PTN regulates/regulate/v:G3e */*/PTN
17	*/*/PTN associates/associate/n:m with/with/prep */*/PTN
15	*/*/PTN activation/activation/n:e and/and/cnj:K */*/PTN
14	*/*/PTN -induced/-induced/adj:b */*/PTN
14	*/*/PTN interacts/interact/v:G3e */*/adv:b with/with/prep */*/PTN
13	*/*/PTN enhances/enhance/v:G3e */*/PTN
13	*/*/PTN activates/activate/v:G3e */*/PTN
13	*/*/PTN is/be/v:G3e required/require/v:P for/for/prep */*/PTN
12	*/*/PTN induces/induce/v:G3e */*/PTN
11	*/*/PTN phosphorylation/phosphorylation/n:e of/of/prep */*/PTN
11	*/*/PTN association/association/n:e with/with/prep */*/PTN
11	*/*/PTN binding/binding/n:e ,/,/pun */*/PTN
11	*/*/PTN receptor/receptor/n:e for/for/prep */*/PTN
11	*/*/PTN interaction/interaction/n:e with/with/prep */*/PTN
11	*/*/PTN in/in/prep */*/det regulation/regulation/n:e of/of/prep */*/PTN
11	*/*/PTN activity/activity/n:e of/of/prep */*/PTN
11	*/*/PTN binding/binding/adj:b domain/domain/n:e of/of/prep */*/PTN
11	*/*/PTN -mediated/-mediate/v:V */*/PTN
11	*/*/PTN binding/binding/n:e site/site/n:e on/on/prep */*/PTN
10	*/*/PTN degradation/degradation/n:e ,/,/pun */*/PTN
10	*/*/PTN binding/binding/adj:b */*/PTN (/(/pul */*/PTN
10	*/*/PTN binding/binding/adj:b activity/activity/n:e and/and/cnj:K */*/adj:b */*/PTN
10	*/*/PTN inhibits/inhibit/v:G3e */*/PTN
10	*/*/PTN -induced/-induced/adj:b activation/activation/n:e of/of/prep */*/PTN
10	*/*/PTN activity/activity/n:e ,/,/pun */*/PTN
10	*/*/PTN)/)/pur receptor/receptor/n:e (/(/pul */*/PTN
10	*/*/PTN in/in/prep */*/n:e to/to/prep */*/PTN
10	*/*/PTN activated/activated/adj:b */*/PTN
10	*/*/PTN form/form/n:e of/of/prep */*/det */*/PTN

Table 5.5.: Initial phrases extracted from MEDLINE using IntAct data; shown are phrases with a support of at least the ten, that is, the number of sentences that result in the same initial phrase after replacing nouns, adjectives, adverbs, verbs (as long as they are none of the interaction words from Table A.3), modal verbs, determiners, numbers, punctuation, conjunctions, and protein names with *, applying to token and stem. Format of each word: token/stem/POS-tag.

Core phrase	Comment
PTN bind PTN PTN despite its ability to bind PTN	unusual phrasing (list of proteins on the left) interspersed qualification
PTN mutant inhibited PTN PTN) bound to PTN PTN / PTN blocks PTN	interspersed modifier interspersed abbreviation for the protein name interspersed synonym of the protein name, might also refer to a protein complex
PTN to PTN and then activate PTN	complex formation, A+B activate C
PTN binding sites for PTN PTN binding domain of PTN	synonymous expressions
PTN suppressed PTN PTN suppressed the effects of the PTN PTN also specifically binds to PTN	extended noun phrase extended verbal phrase
PTN –independent forms of PTN PTN were defective in binding PTN	suggesting that there *is* a dependent form → normally an effective interaction
PTN can affect PTN PTN is required in PTN	goes together with other modal verbs not necessarily a physical interaction, few FPs
PTN + T–lymphocytes expressing PTN PTN -negative cells expressing PTN	uncertain veracity as pattern → some FPs
PTN but did not affect PTN	refers to a protein left of the snippet, but the mentioned pair *might* still interact
PTN expression was reduced while PTN	suggests a study of a potential relationship between the two proteins → some FPs
PTN was not affected by the PTN	negation in contrast to IntAct data, different behavior under different conditions; "not" treated as adverb, thus matches "always" etc.

Table 5.6.: Sample of the core phrases extracted from MEDLINE. The top half shows typical examples, some with unusual phrasing, to demonstrate the high variability (might be missed during manual pattern engineering). Some core phrases that might lead to false positives (FPs) when used as patterns are shown in the bottom half. Protein names have been replaced with PTN.

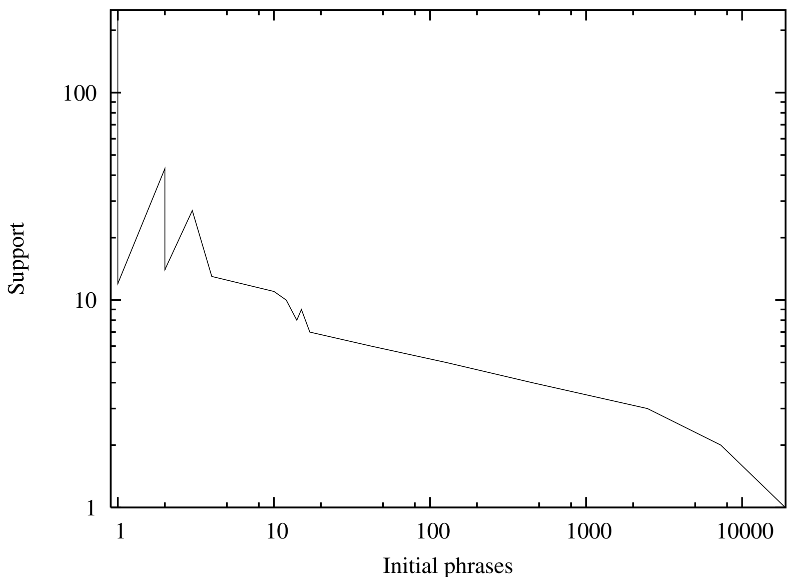


Figure 5.2.: Distribution of the support of initial phrases extracted from the IntAct data, being close to an expected Zipfian distribution.

proteins in IntAct	31.471
protein pairs from IntAct	41.748
sentences with at least one IntAct pair	67.870
IntAct pairs in the sentences	117.460
phrases with IntAct pair + interaction verb	
0/0 word boundary *	20.439
1/1 word boundary	23.573
phrases with IntAct pair + interaction noun	
0/0 word boundary	19.954
1/0 word boundary *	19.972
0/1 word boundary *	21.932
1/1 word boundary	24.284
2/0 word boundary	24.498
phrases with IntAct pair + interaction adjective	
0/0 word boundary *	2.413
phrases with IntAct pair + interaction verb, noun, or adjective	
0/0 word boundary	29.285
final set of consensus patterns from (*)	9.640

Table 5.7.: Statistics for MEDLINE sentences and interaction pairs from IntAct (May 2005). Duplicate sentences are removed. Boundaries refer to additional tokens to the left/right. The final set of patterns consists of the individual sets indicated by *.

Method	Precision	Recall	F1 (in %)
Initial pattern set	85.8	15.2	25.8
CP, single layer (POS tag incl. entity)	76.6	47.1	58.3
CP, multi layer (token, POS tag, stem, entity)	78.7	51.9	62.6
CP, optimized for precision	+1	-4	60.1
CP, optimized for recall	-5	+5	63.9
Current best system: Hao et al., 2005	79.8	59.5	68.1

Table 5.8.: Impact of different parameters for collection and refinement of patterns. For optimized sets, we use 70% of SPIES to adjust thresholds for each single pattern.

tems applied to the biological domain"[2]. Two rounds of this competition were held in 2003/04 and 2006/07, respectively, with a third starting in mid-2009. The first round focused on the tasks of protein named entity recognition and gene mention normalization (mouse, fruit fly, and yeast genes), as well as functional annotation of proteins using Gene Ontology terms[3]. The second round was concerned with protein NER, EMN of human genes, and four tasks concerning protein-protein interactions: classification of abstracts and sentences (each for containment of PPIs), extraction of protein interaction pairs, and extraction of protein interaction identification methods (such as yeast-2-hybrid). BioCreative is held as a competition-like task, where participating teams are first provided with training data to develop their respective systems, and later with test data on which predictions have to be made and then submitted to the organizers for evaluation (typically within a week upon release). The BioCreative task relevant to our analysis in this chapter is the interaction pair extraction subtask (IPS), aiming at the extraction of pairs of interacting proteins from full-text articles. Extraction includes mapping of all proteins to their respective UniProt identifiers. We described the recognition and identification of protein names in Chapter 3, and briefly summarize our approach to the IPS task in the following (details are provided earlier in this chapter and in Chapter 4).

We ran four kinds of experiments on the BioCreAtIvE II IPS corpus. For the relation mining component, we used consensus patterns identified using clustering and multiple sentence alignment (Section 4.3) on the aforementioned IntAct collection of initial phrases. As a variation of the test performed on SPIES, we did not generalize prepositions to arbitrary tokens, which should result in more precise consensus patterns. The experiments differed in five parameters, which should tune either precision or recall of the overall system, and influence the named entity identification or pattern matching.

1. For each protein interaction and publication, we set a minimum number of predicted interactions resulting from different evidences in the text. That means, an interaction that was predicted too few times was removed from the final predic-

[2]BioCreAtIvE: see http://biocreative.sourceforge.net/
[3]GeneOntology: see http://geneontology.org/

tion.

2. When clustering initial phrases from the IntAct sample, a cut-off specifies how far the computed guide list if pursued. This means, the cut-off defines the minimum similarity of initial phrases that occur in one cluster.
3. For clustering initial phrases, we also set a maximal length difference for phrases in the same cluster.
4. As named entity identification influences the recall (and precision) quite strongly, we also tested results for the top two predicted identifiers per protein, the top three, and so on.
5. To resolve proteins to species, we used different scopes. First, we analyzed the abstract, and compared the organism name(s) found therein with the species assigned to each remaining UniProt ID; if nothing was found or still multiple IDs remained, we took the most likely ID. Another scheme, resulting from frequency counts, was to first look in the abstract, but then take the best human, mouse, or yeast protein (in this order) before deciding on another species.

All results for different parameters settings are shown in Table 5.9, together with the currently best-performing system (Huang et al., 2007). Results of the other top-performing systems can be found in the section on Related Work, see in particular Table 5.10 on page 105. The parameters are abbreviated *m* (minimum of required evidences), *gl* (guide list cut-off), *ld* (maximum length difference), *ids* (submitted IDs per protein), and *sr* (order of species resolution), respectively. As BioCreative provided training data for the protein-interaction extraction task, and our patterns were extracted independently from PubMed abstracts, we could use the whole training set for parameter optimization. Given the set of patterns that performed best on SPIES, parameter tuning focused on the aforementioned, task-specific parameters. For one, those are parameters that are decoupled from the actual relationship extraction, but instead focus on the task-specific needs for mapping proteins to species, protein normalization, detection of novel interactions, and so on. Second, the guide list cut-off and maximum length difference of initial phrases were adapted to the BioCreative training set. All four parameters were carefully manually chosen, according to trial-and-error runs on the training data. Regarding the relationship-extraction parameters (guide list, max. length), note that these were specifically tuned to BioCreative data, which are full-text publications rather than abstracts. It can be expected that on different tasks, these parameters will have to be adjusted. Regarding the non-relationship extraction specific parameters (species, number of submitted interactions), the submission of multiple IDs per protein has a drastic impact on the precision of the overall system. Predicting two IDs per protein results in four potential pairs, three of which are certain false positives; this limits the maximal achievable precision to an upper bound of 25% (not including the precision of the EMN itself).

On the BioCreAtIvE II GN corpus, our system for gene name identification yielded a maximal F1-measure of 81% (precision 78.9% at 83.3% recall). This GN tasks included 30,000 human genes from EntrezGene only. For the IPS task, named entity identification included more than 325,000 protein names from UniProt. We can thus estimate an

Short description	Precision	Recall	F1 (in%)
m=2; *gl*=0.83; *ld*=3; *ids*=2; *sr*=ahmy*	0.2	43.7	0.3
m=1; *gl*=0.83; *ld*=3; *ids*=1; *sr*=a*	25.2	23.3	21.1
m=1; *gl*=0.75; *ld*=5; *ids*=1; *sr*=a*	22.8	21.6	19.7
Current best system: Huang et al., 2007	37.0	32.7	30.4

Table 5.9.: Results for different strategies using the IntAct pattern collection on the BioCreAtIvE II test set. *m*, minimum of identified interactions per pair and article required for a prediction; *gl*, guide list cut-off; *ld*, max. length difference; *ids*=number of submitted IDs per protein; *sr*=species resolution (see text for explanations).

upper bound for the precision for named entity identification alone; this bound is expected to be lower than $78.9^2\% = 62.3\%$ for any pair of two proteins. This bound holds for statistical independence, which might not be given for protein pairs (note, however, that interacting proteins do not always originate from the same species). Our maximum recall for recognizing single proteins on the IPS set was 69%, being the highest among all other participating systems.

When we required only one identified evidence pair, precision dropped significantly. One reason is the occurrence of non-important interactions in the article, which are not annotated in the corpus. Such interactions often occur in the introduction of an article, where it becomes clear that this particular protein-protein interaction is not one of the main conclusions of the current article, but rather a repetition of known facts. The bibliography of an article is also prone to contain interactions —in titles of cited publications—, which provide additional background information. All these are actual protein-protein interactions taken for themselves, but count as false positives according to the definition of the BC2 IPS task. Another, similar reason for low precision is the occurrence of protein–protein interactions in dangling text; that is, articles that end on the same page on which the current articles begins, or vice versa. Automated systems thus identify interactions in parts that do not belong to the current text.

5.4. Related Work

Methods for relation mining mostly fall into either of the four categories

- statistical methods, machine learning,
- rule-based approaches, linguistic frames,
- methods based on syntactic parsing, and
- hybrid approaches.

Statistical methods range from approaches calculating co–occurrence frequencies to classification-based approaches. The later apply machine learning to infer models from annotated data, using words, word order, part-of-speech tags, and information on linkages. Predictions range from relevance ranking of sentences and paragraphs to detecting the actual objects in a relation. Rule-based approaches mostly use a form of regular expressions/language patterns to detect relations, and often include 'basic' syntactic information such part-of-speech tags. Such rules are extracted from labeled data, and applied to new text. Depending on the approach, predictions can be either binary (e.g., pattern matches) or result in a score (e.g., edit distance). Parsing involves syntactic analysis of sentences to find verb and noun phrases, constituent trees, subject-object dependencies, and other linkage structures. Until recently, sufficiently well performing parsers have not been available for the biomedical domain, which made it difficult to assess approaches building on parsers (compare Daraselia et al., 2004, and Miyao et al., 2008) — as opposed to the other two categories, where implementations of machine learning algorithms, part-of-speech taggers, etc., are readily available. Hybrid approaches combine some of these techniques: including POS tags or parse tree information as features in machine learning, using linkages in patterns, or defining entire patterns over parse trees (instead of words and POS tags).

In this section, we will discuss the most prominent approaches, separated by category; we also present related work on sentence alignment and pattern–based relation extraction other than for protein–protein interactions. We compare different systems qualitatively and, where possible, quantitatively; in many publications, systems have been tested on in-house corpora only, making a meaningful comparison based on pure numbers impossible. An overview of approaches and quantitative results is shown in Table 5.10. This section ends with conclusions drawn from our and related work.

5.4.1. Evaluation corpora for relationship extraction

The evaluation presented in this section focuses on the extraction of protein-protein interactions. Several such corpora have been made available, refer to Section A for a list with statistics and references. Pyysalo et al. [2008] proposed to use a set of five corpora, consisting of AIMed, BioInfer, HPRD50, IEPA, and LLL'05, for evaluating systems for protein interaction extraction. They provided a unified format, together with conversion tools for each of these corpora, so experiments can be conducted without having to parse different formats. The corpus annotation guidelines and sizes differ vastly. AIMed, BioInfer, HRPD50, and IEPA are based on PubMed abstracts, and annotate only protein-protein interactions. LLL'05 contains annotations not only for PPIs, but also genic interactions such as genes occurring in the same operon structure. For both reasons, the type of text and annotation guidelines, which also differ between the first four corpora, results cannot be directly compared to each other. LLL'05 is the only corpus that has a fixed split of training and test data, where the actual test data is not publicly available. Instead, LLL'05 offers a web service, where predictions can be submitted and scores are returned. Thus, it is guaranteeed that no parameter tuning can be done on the test data (although by repeated querying, the actual gold standard for the test data could be reconstructed). Critique regarding the LLL'05 corpus concerns its size: there are only 52 interactions in the test set and 103 in the training set. That means that for each evaluation, a single additional true positive leads to an increase by 2% in precision (recall and f-score correspondingly). Thus, close results for differ-

Studied in	Underlying method	F1	Test collection
Daraselia et al. [2004]	syntactic parsing, mapped to frames	P:91	361 pairs
Blaschke and Valencia [2002]	31 hand-crafted patterns	42.3	300 abs.
Stapley and Benoit [2000]	co–occurrence	>R	2524 abs.
Marcotte et al. [2001]	text classification + co–occurrence	n/a	260 abs.
Friedman et al. [2001]	≈3000 patterns	P:95	
Saric et al. [2006]	hand–crafted rules	44	6640 abs.
Huang et al. [2004]	pattern set from annotated corpus	60.7	SPIES
Hao et al. [2005]	pattern set from annotated corpus + MDL	68.1	SPIES
*Hakenberg et al. [2005b]	refined hand–crafted patterns	51.8	LLL'05
Riedel and Klein [2005]	clauses based on discourse representation	52.6	LLL'05
Fundel et al. [2007]	parse tree–patterns	72	LLL'05
Kim et al. [2008a]	Tree kernel SVM	77.5	LLL'05
Katrenko and Adriaans [2008]	Local alignment kernel	80.5	LLL'05
	Local alignment kernel	78	BC1-PPI
*Plake et al. [2005]	patterns learned from annotated corpus	39	BC1-PPI
Jang et al. [2006]	parse tree patterns	56.0	BC1-PPI
Sun et al. [2007a]	pairwise classification	>80	BC1-PPI
Miyao et al. [2009]	Reranking parser, subset tree kernel	59.5	AIMed
*Plake et al. [2005]	patterns learned from annotated corpus	R:53	DIPPPI
Huang et al. [2007]	pattern set from annotated corpus + MDL	30.4	BC2 IPS
Rinaldi et al. [2007]	co-occurrence + lexico-syntactic filtering	23.5	BC2 IPS
Baumgartner et al. [2007]	semantic parsing with 67 rules	25.3	BC2 IPS
*Hakenberg et al. [2007]	consensus patterns from IntAct+PubMed	21.1	BC2 IPS

Table 5.10.: Performances reported for different systems for extraction of protein–protein interactions from text. BioCreAtIvE II IPS data (BC2 IPS) additionally required protein name normalization, so numbers are lower in general; BC2 numbers refer to the data set restricted to UniProt proteins. Note that evaluation methods and corpora differ by large, and most results are not comparable, but give only hints on the effectiveness of approaches. F1–measure in % if not indicated otherwise, P: precision, R: recall, >R means high recall, abs: abstracts, *: systems presented in this thesis, all other approaches are discussed in more detail in Section 5.4.

ent systems or parameter tuning within the same system have to be analyzed properly to exclude performances differing due to pure chance. For example, as we see in Table 5.10, the two systems ranked first during the actuall LLL competition 2005 differ only by 0.8% in f-score, so no conclusions can be drawn. It showed that our system (see previous section) had almost balanced precision and recall, and the recall in Riedel and Klein [2005] was 9% lower with a corresponding 9% higher precision. Therefore, conclusions on the performance of different approaches to PPI extraction are more reliable when drawn from larger corpora such as AIMed and BioInfer. The BioCreative IPS corpus is a large-scale and full-text corpus, with the training set consisting of 740 and the test set of 358 publications. However, the task for BioCreative IPS differs from the aforementioned corpora in two major ways: *a)* Evaluation is done on a per-document level, which means that only a single occurrence per PPI/document has to be found; all other corpora require the extraction of each individual interaction. *b)* The task also include the identification of proteins and the detection of physical and novel interactions for which a document provides experimental support; other corpora do not require to identify proteins (i.e., map them to UniProt identifiers), which in itself is not a solved task. Filtering all predictions by novelty and physical evidence poses an additional challenge, here related to information retrieval. Thus, performances as evaluated on BioCreative IPS are lower in general, because these sub-tasks and their respective individual performances yield a lower bound for the protein extraction task.

5.4.2. Co–occurrence–based approaches

The first attempts for relationship extraction from biomedical text used co–occurrences of entity names in a sentence or larger discourse. The granularity of discourses intuitively has a large impact on precision versus recall. Co–occurrences in headings, titles, or figure caption will be more reliable indication for an actual relation than co–occurrence in arbitrary phrases or sentences; these in turn yield higher precision than abstracts, or any other paragraph in an article. On the other hand, the larger the discourse, the higher the recall will be. In general, co–occurrence–based approaches achieve very high recall values, but low precision (compare Stapley and Benoit [2000], Marcotte et al. [2001], Stephens et al. [2001]. Ding et al. [2002] studied the impact of choosing the text unit for extracting interactions between biomedical terms (such as 'flavonoid' and 'cholesterol') based on co–occurrences; they focused on the question whether to consider phrases, sentence pairs, sentences, or abstracts. In that order, recall increased from 62 to 100%, while precision dropped from 74 to 57%. The highest f-measure could be achieved choosing abstracts or sentences (both 73%), phrases follow with 68%, and sentence pairs are worst with only 50% (thus improvement of performance on abstracts and sentences was significantly better, $p < 0.05$). Rinaldi et al. [2007] experimented with distances between two protein mentions, either limited to single sentences or allowing pairs to cross sentence boundaries. The former increased precision, recall, and f-measure (f-measure from 7.3 to 8% on BioCreative II IPS).

When large corpora are available, like the set of all 18 million PubMed abstracts, one can try to improve the reliability of co–occurrence–based predictions by aggregating over the entire collection. Another strategy is to measure whether a particular co–occurrence is more significant than simple chance. Bunescu et al. [2006] compare these two approaches, studying both different aggregation functions and different statistical distributions. Regrettably, the study lacks a direct comparison of baseline methods (simple co–occurrence, or using subsequence kernels) to aggregation and corpus–level statistics, and only report results for integrated models (subsequence kernel plus aggregation, or adding point-wise mutual information statistics). Gathering such statistics, however, requires to solve one fundamental task, namely named entity identification (discussed in 2.1.2), so that interacting pairs where authors used different names can be mapped together, or separated in case of ambiguous names.

Jenssen et al. [2001] built a gene–gene co–occurrence network ('PubGene') for 13,712 human genes names, from ca. 10 million PubMed abstracts (and enriched the network with available MeSH and Gene Ontology annotations). Taking into consideration the co–occurrence frequency of each pair, they found a precision of 71% for pairs occurring at least five times, and 60% for pairs occurring at least once; estimations were done by cross–checking 500 random pairs of each category. Note, however, that the task was not to find protein-protein interactions, but any kind of valid relations, including gene homology and expression correlation. A comparison to the Database of Interacting Proteins (DIP, Xenarios et al., 2001) that contained 169 human protein pairs revealed that the PubGene network covered 51% of those (DIP database as of 2001; pairs that had references to articles indexed by PubGene).

In the OntoGene system, plain protein co–occurrences are filtered further to identify actual interactions [Rinaldi et al., 2007]. To deal with the specific task of BioCreative II IPS, Rinaldi et al. add a 'novelty filter' constructed from training data to distinguish between interactions relevant to the curation task or not. This step increased the f-measure from 8 to 12.8%. Actual filters for interactions versus coincidental mentions of two proteins in OntoGene are based on hand-crafted lexical, morphological, and syntactic filters, for instance taking into account the (potential) syntactic dependencies between two proteins. Such filtering results in a final performance of 21.7% on the BioCreative II IPS test set for all proteins.

5.4.3. Classification

Sun et al. [2007a] use maximum entropy modeling (MEMM) to first judge sentences for relevance to discussing protein-protein interactions, and then classify each candidate pair of proteins in such sentences to find interacting pairs. The first step takes only the number of proteins and interaction-indicating words as features. Arguing that shallow lexical features contribute most to the performance, as compared to parse of chunk information (also see Xiao et al. [2005]), Sun et al. use the tokens in a window of two to the left and right of both protein names as the only features (despite the binary output of the first step). Using ca. 50% of the BC1-PPI corpus each for training and testing, the approach yields an accuracy of 81.9% when using both steps, and 81.2% when using the second step only. Maximum *per-class* precision and recall were 86.8%/80.1% for negative instances (non-interacting protein-pair) and 81.6/72.7 for positive instances, respectively, so the joint f-measure would be around 80%.

5.4.4. Pattern–based approaches

The probably most popular approach to find PPI is by matching the text against predefined patterns, that is, regular expressions over words or word stems, pre–recognized tokens, linguistic information, or combinations of those [Blaschke and Valencia, 2002, Hao et al., 2005, Hakenberg et al., 2005b, Saric et al., 2006]. Given the "right" set of patterns, pattern matching can yield both good precision and good recall. However, the major problem of pattern matching approaches is the creation of this "right" set of patterns. There are two options: Either a very large pattern base is created, covering many sentences, which is very time-consuming. Second, a smaller set of patterns is somehow generalized, which increases recall, but might lower precision.

We described our pattern–based approaches in Chapter 4 and previously in this chapter. Apart from ours, there are a number of other systems that use pattern matching. The SUISEKI system [Blaschke and Valencia, 2002] used a set of 31 hand–crafted patterns including eight negations and yielded 45% precision at a 40% recall level, evaluated on 300 abstracts. Patterns allowed for gaps between anchor positions and contained some fixed positions, for example, prepositions. The GENIES system can be considered as a mixture between pattern matching and natural language processing (NLP), as it used patterns over linguistic information [Friedman et al., 2001]. It reached precision of more than 95%, but required a very large set of hand–crafted patterns. Saric et al. [2006] created sets of rules for gene expression and (de)phosphorylation relations, achieving an accuracy of 83 and 95% for the two types of relations, respectively. In former work, we used manually created patterns that were optimized in a second step using genetic optimization [Plake et al., 2005, Hakenberg et al., 2005b]. Results for precision were encouraging, but, as for all systems based on manually created rules, recall was a problem.

The system that is most similar to our current approach is SPIES [Hao et al., 2005]. It is a semi–automatic system that generates patterns automatically from a manually annotated corpus. Patterns in Hao et al. [2005] use POS tags and word lists to describe observations at each position. From an annotated corpus, a pattern set is generated by removing redundant patterns. Redundancy is defined by the existence of a more general pattern that covers the same (or more) of the initial sentences. Hao et al. refine patterns using the minimum description length (MDL) method. MDL compresses groups of sentences by finding similar regions in the sentence and generalizing them (e.g., using regular expressions to describe the disjunction). MDL is given by the sum of the description lengths of all patterns and the number of false positives and false negatives the set produces. SPIES reduces a set of 192 initial patterns to only 14 merged patterns, increasing the f–measure from 60 to 68% on the SPIES corpus, which accompanies the system. Note, however, that this corpus contains only sentences that contain at least one interaction, which both helps in generating minimal patterns as well as during their application on the same kind of data. The approach we propose in Section 4.3 also starts with an initial set of patterns, which is by far larger because it originates from a vastly larger document collection (it is based on unannotated texts, an abundance of which exists). Instead of removing redundant patterns, we combine similar patterns into consensus patterns to express variability. Concerning alignment as a matching strategy, adapting the principle of Occam's razor [Thorburn, 1918], the best alignment is the one with the least edit operations. Conery [2007] use MDL to realign DNA sequences. They compare their results to a prediction by ClustalW, finding a disagreement of 38%, and conclude that only a comparison with known alignments would determine which was better.

Saric et al. [2006] hand–crafted two sets of rules, one for extracting gene expression relations, the other for phosphorylation events. Such rules are derived from argument structures of relevant predicates (e.g., 'phosphorylated' or 'participates'), where arguments can be filled by fixed words that also act as triggers. As an example, observing the noun 'phosphorylation' in a sentence triggers all rules that allow for this noun as an argument of their predicate. A phosphorylation event then requires two more arguments, namely two proteins, to be present in the same sentence. Saric et al. estimated performance on a set of 6640 PubMed abstracts that discuss yeast and mention two genes/proteins. For gene expression, the system achieves an accuracy of 83%, and for phosphorylation, 95%. They obtained similar performances for three other species. An f-measure of 44% was estimated by cross-checking 250 sentences.

Baumgartner et al. [2007] use a form of semantic parsing following the Direct Memory Access Parsing paradigm (DMAP; see Riesbeck, 1986). All major components computing to the extraction of relations are performed in a single process and thus are directly interdependent. The system builds on 67 rules that were manually created together with domain experts. These rules resemble regular expressions and include entity classes, fixed words, and conceptualized word categories. Such concepts refer to pre-defined lists of words observed for a given position. In general, interaction-indicating verbs are grouped by conjugation, the same holds for interaction nouns, and some basic categories such as prepositions etc. Certain positions in each rule can be expanded using a fixed grammar. For instance, single protein names in a pattern can be expanded to a list of proteins (separated by commas or other conjunctions). Each rule thus also captures sentences that include such enumerations. On the BioCreAtIvE II IPS test set, this system achieved an f–measure of about 28%. OpenDMAP is backed by concepts sorted in an ontology of default and problem specific word categories, and can easily be maintained by the ontology editor Protégé. In our approach using consensus pattern, we generate position-specific word lists, as defined by alignment; these lists differ from position to position, as well as from pattern to pattern. As a future perspective, we may try to combine word lists observed in different patterns (e.g., when they overlap to a certain extent in tokens, stems, and/or part-of-speech tags), and extend existing patterns with new word lists to make up for potential occurrences not observed even in a large corpus.

5.4.5. Sentence parsing and graph kernels

Until recently, systems using linguistic parsing suffered from insufficient results from underlying parses, as many sentences cannot be parsed unambiguously using current techniques [Daraselia et al., 2004, Yakushiji et al., 2001, Friedman et al., 2001]. The ARIZONA RELATIONAL PARSER was reported to reach very good precision and recall values, but it required the creation of 1778 grammar rules and was evaluated on a very small corpus only, without cross validation McDonald et al. [2004]. The PREBIND system reached an f–measure around 90% for the task of predicting whether or not a a sentence contains a protein-protein interaction, without actually extracting it [Donaldson et al., 2003]. Fundel et al. [2007] use dependency parse trees to extract PPIs from text, resolving local dependencies of recognized objects. This leads to an increase in precision (exact dependency resolved, independent objects skipped) and also recall (dependency of objects that are 'far' away in a sentence). They use three rules to extract matching paths in the dependency tree; these rules reflect constructs commonly used to expression PPIs: "effector–relation–effectee", "relation–of–effectee–by–effector", and "relation–between–effector–and–effectee". Fundel et al. [2007] evaluated the system on the LLL'05 data set and achieved an f–measure of 72%.

Daraselia et al. [2004] employ a context-free grammar to parse sentences in their MedScan system. As the syntactic parser yields a set of possible structures, they further analyze each candidate and transform it into a semantic tree. This transformation uses the identified predicate structure and prepositional patterns. MedScan then maps arguments to an ontology, trying to fill slots in a predefined concept frame for protein interactions. One set of frames corresponding to biomedical objects (proteins, cells), the other to protein functions (regulation). Daraselia et al. thus searched for triplets linking two proteins by a function. They extracted 2976 human protein-protein interactions from a collection of ca. 3.5 million sentences (Medline after 1988); manual cross-checking revealed a high precision of 91% at 21% recall. They also compared their predictions to the DIP and BIND databases, and found that 96% of the data could be considered new, as it appeared in neither one.

AKANE [Sætre et al., 2007] parses sentences with the Enju HPSG parser [Yakushiji, 2006]. It recognizes pairs of proteins and finds the smallest connected parse tree that includes two proteins. The system builds on such partial trees and extracts them as patterns from a training set (AImed or any other corpus). These patterns are then compared to parse trees of arbitrary sentences, which are also reduced to smallest connected parse trees covering two recognized proteins. On the AImed corpus, AKANE achieved an f–measure of 42% (70% precision at 30% recall). On the BioCreAtIvE II IPS test set, the f–score was 15.8% (equal precision and recall).

Jang et al. [2006] use the full sentence parse and analyze the path between (protein) leaf nodes. In most of the cases, two proteins are connected via a noun phrase and verb phrase (NP+VP: the first protein is in the noun phrase, the second protein occurs as an object in the verb phrase), a noun phrase and a prepositional phrase (NP+PP), or a conjugating conjunction (NP+CC+NP). Jang et al. search for these three patterns in each sentence parse, and apply two additional templates to find interactions described as is-a relations ("JAB is a regulator of JAK2 phosphorylation") and adjectival phrases ("CD38-associated Lck"), which are not covered with the three patterns. Using pre-defined lists of interaction-indicating verbs and nouns, the system decides on interactions when these occur along the path connecting two proteins. To obtain the syntactic structure of a sentence, Jang et al. use the Stanford parser with part-of-speech tags from a Brill tagger trained on GENIA. On the BC1-PPI corpus, they achieve a precision of 81.3% at 42.7% recall (F1: 56.0%). Their system extracts 44% of the interactions in the DIP-PPI corpus. Jang et al. also found that simplification of sentences by substituting named entities and noun phrases with single nouns, and removing bracketed expressions boost the initial parsing performance by 12%.

Instead of full sentence parsing, Park et al. [2001] use shallow parse information in their system to find functionally related proteins. Such parsers identify phrasal components, finding noun and verbal phrases, and dependencies between adjacent phrases. Starting with a search for interaction-indicating verbs, the system tries to identify verbal phrases that include two proteins as noun phrases. Parsing is done with a combinatory categorial grammar parse (CCG) in both directions starting from the verb. Out of 250,000 abstracts discussing "cytokines", the system identified 492 that have relevant keywords, and extracted 182 relations from these sentences. On this set, manual assessment showed a precision of 80% at 48% recall.

Another category of algorithms, only recently introduced, are convolution kernels, also graph kernels. Convolution kernels essentially calculate the similarity between parse trees. Tsochantaridis et al. [2005] presented one of the first implementations, SVM^{struct}, that is able to predict structured output, such as trees or sequences[4]. For predicting sequence in named entity recognition, Tsochantaridis et al. [2005] were able to demonstrate errors rates decreasing from 9.17% (HMM) and 5.17 (CRF) to 5.08 (structured SVM). For predicting parse trees, their structured SVM outperformed a probabilistic context–free grammar parser (PCFG) by 2.5%, achieving an f–score of 88.5%. However, such methods have not yet been largely studied for information extraction tasks. Kim et al. [2008a] studied four different formulations of tree kernels. The best performing kernel takes as features "walks" connecting two named entities in a parse tree, using positive and negative examples. They achieve 77.5% on the LLL'05 data and show that more complex formulations —such as a dependency kernel that takes into account information on the types of linkages connecting two entities, which at first looked more promising—, where outperformed by up to 17%.

Katrenko and Adriaans [2008] presented a local alignment kernel, based on the Smith-Waterman algorithm Smith and Waterman [1981]. The later defines the distance between two sequences as the local alignment score resulting from their best alignment; by summing up over the contributions of each possible alignment, Katrenko and Adriaans could derive a valid kernel. Of particular interest then is the definition of a substitution matrix. Katrenko and Adriaans compare different distributional smiliarity measures, namely Dice, Cosine, and Euclidean (also cmp. Section 2.2.2 on page 26). Experiments included the BC1-PPI and LLL'05 corpora (see Appendix A); the first was parsed using the Stanford parsers, for the second, parse information comes with the data set (Link parser). On LLL'05 data, the local alignment kernel using Dice or Cosine achieves an F-measure of 80.5%, as compared to a baseline of 56%. Best results on the BC1-PPI were achieved with the Euclidean distance metric, slightly outperforming the other two metrices (78% F-measure, 10-fold cross-validation).

Miyao et al. [2009] compare various dependency and phrase structure parsers (MST, KSDEP; Stanford, Berkeley, etc.) used as input for subset tree kernels Sætre et al. [2008], Moschitti [2006]. They also evaluate the contributions of different representations, such as predicate-argument-structure (PAS), Penn-treebank (PTB), or Co-NLL-X. On AIMed data (protein-protein interactions), a hybrid approach combining Charniak & Johnson's reranking parser Charniak and Johnson [2005] with a PAS representation performed best, with an F-score of 59.5%, as compared to a bag-of-words-baseline at 51.1% and also outperforming approaches previously evaluated on the same data set. Another interesting conclusion drawn by Miyao et al. [2009] is that a 1% improvement in parser accuracy corresponded to a .25% increase in relationship extraction performance.

[4]See http://svmlight.joachims.org/svm_struct.html

5.4.6. Sentence alignment

Conery (2007; also see discussion of pattern-based approaches) propose a sequence alignment strategy that divides (pairs of) sequences into variable regions, aligning amino acids within these regions, but not across regions. Such a region can have variable length and content in either of the sequences. The alignment at first is a sequence of regions with high similarity instead of single letters. Conery uses the minimum description length (MDL) principle on the encoding of blocks, where blocks with similar letters require shorter encodings. In a single experiment, Conery compared their predictions on aligning two sequences to ClustalW. He showed a discrepancy of 38% between both predictions; further analysis would require a study involving a set of known alignments.

Och and Ney [2000] compare different statistical alignment models for statistical machine translation. Among two fertility models [Brown et al., 1993] and four hidden Markov models, a fertility model that models a jump distance for target words performs best. These model the alignment probabilities $p(i|i-1, I)$ and translation probability $p(f|e)$ for target position i, target current target sequence I, and source and target strings f and e, respectively. They find the Viterbi alignment (the alignment for the maximum translation probability) using traditional dynamic programming. Och and Ney also modeled the distance relative to the diagonal in the alignment matrix in a diagonal-oriented HMM model, outperforming all other HMM models. In machine translation, where source and target sentences usually are of equal length, this refers to the main diagonal. In our approach of aligning (typically shorter) patterns with (typically longer) sentences, this technique may be adapted to allow for the diagonal to start at any position within the longer of the two. Och and Ney conclude that using word groups instead of single words could improve their method; for pattern/sentence alignment, this would mainly refer to noun phrases and other chunks (e.g., from a shallow parse) mapping to single word nouns etc. In addition, Och and Ney argue that making use of cognates could result in better alignments. We identify such groups of related terms during the computation of consensus patterns in the token and word stem layers, although they include only single word terms, and make use of these groups during application.

5.4.7. Learning patterns for information extraction

Hearst [1992] used six manually defined patterns to extract hypernym–hyponym pairs from arbitrary text. These pattern consisted of fixed words, word lists, optional words, punctuation, and shallow parse information (noun phrases). An example is the pattern

"NP_0 such as {NP_1, NP_2, ..., (and|or)} NP_n",

where the NP_i depict noun phrases, and the part in curly brackets is optional. A matching sentence will have the hypernym as NP_0, with its hyponyms NP_1 through NP_n. Hearst also proposed a scheme to iteratively extend the sets of known patterns and hypernyms/hyponyms. Starting from a small set, instances are extracted from a large document collection. Other sentences that contain the same instances could then be analyzed to extract new patterns; these could be applied to find more and different instances, and so on. Hearst concluded that the only underdetermined part was to find commonalities between new sentences, in order to extract precise enough patterns (and exclude sentences that discuss instances coincidentally).

In a related approach, McCrae and Collier [2008] generated patterns to find domain-specific synsets from literature, focusing on terms related to disease outbreaks. From seed sets of high coverage patterns, they subsequently identify more precise patterns; each predicted synonym relation is classified further using feature vectors depicting matches with individual patterns and a score inherent to the pattern. Predicted synsets were compared to a manually created epidemiology ontology (for infectious diseases and outbreaks). The system achieves 73.2% precision at 29.7% recall for synsets that map to concepts in this ontology, giving an estimate for the task of finding synsets.

5.5. Conclusions

Four main strategies for collecting patterns for information extraction exist. Apart from manual pattern engineering (e.g., Hunter et al., 2008), automated methods have been proposed that use manually labeled training samples (Hao et al.), bootstrapping from very few labeled examples (e.g., Hearst, 1992), or do not require any labeled data (the method proposed in this thesis). While no evaluation was done on the problem of protein-protein interaction extraction for bootstrapping, experiments using the other three methods suggest that their results are comparable to each other. It might also be possible to use automatically generated patterns sets as input for manual refinement. Subsequently relaxing the constraints for clustering and multiple sentence alignment will result in different sets of concordances for positions across consensus patterns); this helps linguists to grasp the underlying language and its variability (sentence structure, interjections, cognates) to generalize/restrict the initially generated patterns further.

Approaches that include parse tree information, such as Kim et al. [2008b], currently seem to outperform pattern-based approaches, although the later were the top systems presented for BioCreative 2. This particular data set relies on input from previous tasks such as NER and EMN, however, so the influence of relation extraction itself is hard to estimate. Kabiljo et al. [2009] showed that excluding the problem of NER results an performance increase of ca. 20%, independent of the underlying corpus (protein-protein interactions). Regarding sentence parsing at input to information extraction, Miyao et al. [2008] showed that most recent parser perform on par; and that an increase of 1% in parsing performance corresponds to a 0.25% increase in relation extraction performance (again for protein-protein interactions). A major advantage of parse trees and other kinds of linkage structures is information on dependencies between objects. In most kinds of relations, there exists a subject-object dependency conveyed in a sentence, which is included in parse tree-based models, but which is missing in all current pattern-based methods. Such dependencies mostly influence the precision of a method, as it may discard objects that do not correlate to each other in an agent–target relationship. Note that some protein-protein interactions, such as complex formation, cannot be split into active and passive component. Methods using convolution kernels to compare sentences based on their parse trees look particularly promising (but have not been extensively studied), because they combine precisely structured information with the advantage of machine learning. A disadvantage of parsing methods results from the long time required for parsing large corpora. Approximately 20% of the sentences still have incorrect parse trees [Clegg and Shepherd, 2007].

Experiments show that our current implementation of aligning patterns with sentences is not suited for online computations, where time is essential. Processing the IPS test data (70,000 sentences) takes in the order of 100h. After translating the set of initial patterns into OpenDMAP grammar, parsing the same data with these patterns requires less than 10 minutes. One reason for this difference is the quadratic complexity just to obtain the alignment score; another the sequential processing of patterns. Parsing with OpenDMAP, in contrast, is on average linear in the length of the sentences (but has a quadratic worst-case complexity), plus it searches for all patterns in parallel, as they are encoded in a finite state automaton, as discussed in Section 4.2.1. Runtime may be reduced by pre-selection of patterns for each sentence or vice versa, for instance by indexing patterns and sentences by the order of selected word categories. Just using POS tags for proteins and interaction-indicators, sentences that follow the structure "PTN IVBD PTN" do not have to be compared to patterns with different structures, like "INN PTN PTN". However, as alignment is an inexact matching strategy, such indexing may reduce the recall rate of the overall pattern set. An improved implementation of the method for alignment, which also includes parallel processing of distinct portions of the data to be annotated, as well as an indexing strategy, resulted in a speed-up of around 10,000 [Palaga and Leser, unpublished].

So far, none of the techniques based on alignment of sentences has been reported to support groupings of words to be aligned, but work on single words only. Such groupings make sense in natural language processing, as they might refer to noun phrases, reflected by few words in one and a single corresponding word in the other sentence. Other examples are intermediate clauses, relative sentences, bracketed expressions, etc. in one sentence that might not have any corresponding part in the other sentence. One way to address this problem is to introduce dedicated functions for gap costs, making gaps less cost-intensive when aligning two noun phrases of different length or aligning a whole bracketed expression with a gap, and expensive when aligning only a part of a bracketed expression with a gap. We have not addressed this problem in our work.

It is often enough to detect one occurrence of a fact in a text. Especially the basic facts and main contributions of a publication get repeated quite often, and with different levels of complexity. A repetition of a once established fact often occurs in a much simpler form than the initial statement. This can in particular be observed by looking at the complexity of such sentences when they occur in the abstract of a paper, the main thrust, and the discussion with conclusions drawn. In the introductory parts of a publications, long established facts get repeated in a much simpler and more condensed way than in the original publication first to describe this fact.

6. Visualization of networks — AliBaba

In the previous chapters and sections, we discussed methods to pre-process text, recognize named entities, and finally find relationships between entities. We have implemented all these components and combined them to act together as an overall system for information extraction. Especially, we are interested in steps towards collection wide analyses (also see Chapter 2, page 6). A typical application for a text mining system in the biomedical domain would be to process the entire set of PubMed abstracts. A component being able to map all extracted information, by means of linking together identical entities referred to with different names, then becomes mandatory.

With ALIBABA[1], we developed a tool to visualize information extracted from subsets of PubMed [Plake et al., 2006]. In ALIBABA, users select a set of abstracts by keyword queries. We use PubMed's own information retrieval system, based on query expansion, automated term mapping, and including Medical Subject Headings (MeSH) annotations [Nahin, 2008], so the set of abstracts is identical to the one a user would receive from the Entrez PubMed interface. Citations in PubMed are semi-automatically indexed with ca. 40,000 MeSH terms, which is organized in a structured vocabulary; note that indexing includes full-texts, so citations are also indexed with terms not necessarily found in the freely available abstract. On this set of abstracts, ALIBABA, by means described in Section 3.2, searches for entities of the classes

- protein/gene,
- disease,
- drug,
- species,
- tissue,
- cell type/cell line,and optionally
- enzyme,
- compound, and
- nutrient.

The dictionary-based approach to named entity recognition allows to get a list of candidate identifiers per entity mention, which helps mapping entities to a single instance when authors used different names/spelling variations across texts. All IDs refer to entries in databases such as UniProt (proteins), NCBI Taxonomy (species), DrugBank (drugs), etc. ALIBABA then extracts relationships of the following kinds:

- protein-protein interactions,
- (sub)cellular location of proteins,

[1]Available at http://alibaba.informatik.hu–berlin.de

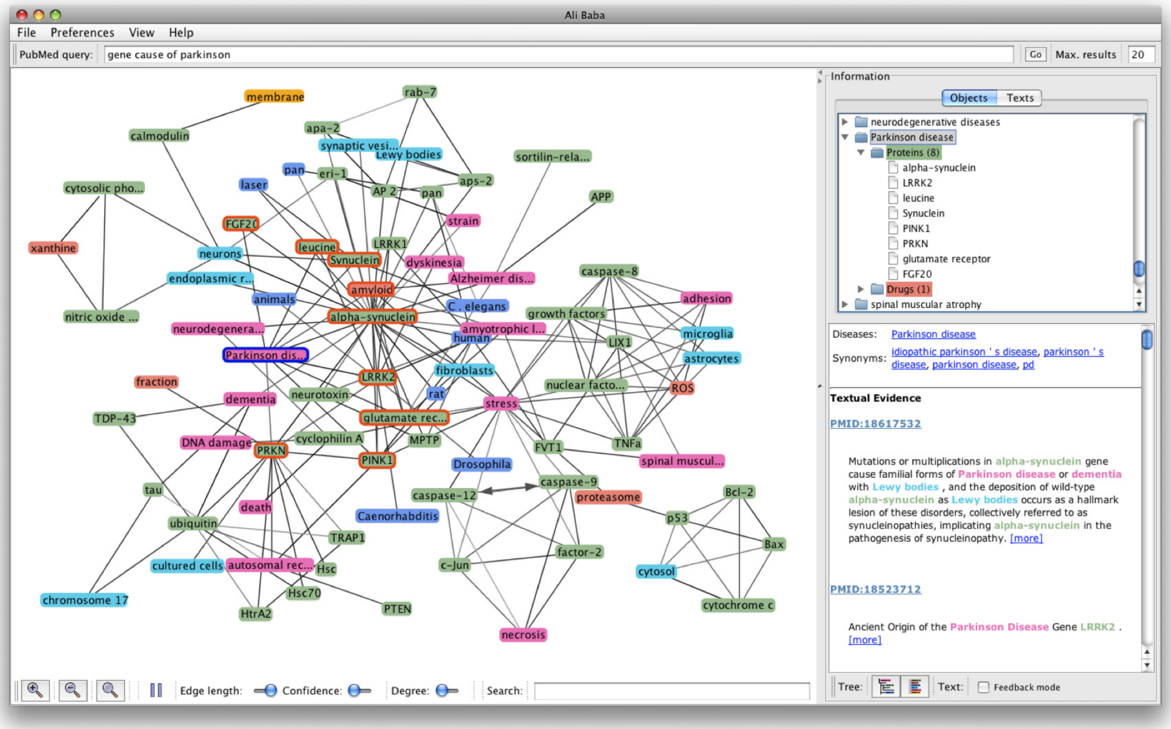
- proteins/genes discussed with drugs,
- proteins/genes discussed with species,
- proteins/genes discussed with tissues,
- relations between drugs and diseases,
- proteins/genes and diseases, and
- nutrients associated with proteins, diseases, or tissues.

For the first two kinds, protein-protein interactions and sub-cellular locations of proteins, we employ a pattern-based approach, discussed in Chapter 4, using alignment as matching strategy. We use the pattern set as described in the previous chapter for protein-protein interactions, and a hand-crafted set of 22 patterns for sub-cellular locations. For all other kinds, we rely on co-occurrence of entities in sentences; relations of the first two kinds are also 'enriched' using co-occurrence. We observed that for most relations, co-occurrence gives a sufficiently high precision, as opposed to protein-protein interactions, where many proteins are coincidentally mentioned. Each relation gets assigned a confidence score; for alignment, this refers to the alignment quality, as defined in Definition 4.2 on page 81; for co-occurrences, we measure the number of words between two entities, and normalize it with the number of entities of the same kind in between. Users can thus decide to view only the most confident relations.

ALIBABA displays the extracted information in three ways. A graph consisting of all entities as nodes and (binary) relations as edges summarizes the information found in the whole collection. A sorted list of entities allows for quick searches. These two are backed with a display of the original texts, with inline markup of entities and relations. Figure 6.1 shows an example summarizing 20 abstracts returned for the query "gene cause of parkinson".

As ALIBABA caches the results for each query (abstracts with annotations, recognized entities and relations), it becomes possible to extend PubMed's information retrieval. A first step is to allow queries for entity IDs instead of entity names. Users thus do not have to write large queries that contain known synonyms for the entities they want to search for. ALIBABA currently supports queries that include UniProt IDs; each such ID gets *i)* looked up in the cache to pre-select corresponding abstracts, and *ii)* expanded according to our own synonym list to relieve the user of this task and then sent to PubMed to retrieve additional abstracts. Note that for queries that include keywords and UniProt IDs, ALIBABA currently retrieves abstracts from PubMed only, rewriting the query by replacing UniProt IDs with lists of synonyms. As an example, the query "Q14790" gets expanded to ("Caspase-8 precursor" OR "EC 3.4.22.61" OR CASP-8 OR MACH OR "FADD-like ICE" OR FLICE OR "Apoptotic protease Mch-5" OR CAP4 OR CASP8 OR MCH5), where originally PubMed users themselves had to provide the extended synonym list, resulting in a more complete set of abstracts discussing this protein.

The overall system architecture of ALIBABA is shown in Figure 6.2. The major components are the user interface (upper left), sending queries to the server (lower right). The server routes the query to PubMed (upper right) to obtain PubMed IDs. If the corresponding abstracts are not yet in the local repository (lower left), they are fetched



neurodegenerative diseases
Parkinson disease
Proteins (8)
alpha-synuclein
LRRK2
leucine
Synuclein
PINK1
PRKN
glutamate receptor
FGF20
Drugs (1)
spinal muscular atrophy

Diseases: Parkinson disease
Synonyms: idiopathic parkinson ' s disease, parkinson ' s disease, parkinson disease, pd

Textual Evidence

PMID:18617532

Mutations or multiplications in alpha-synuclein gene cause familial forms of Parkinson disease or dementia with Lewy bodies , and the deposition of wild-type alpha-synuclein as Lewy bodies occurs as a hallmark lesion of these disorders, collectively referred to as synucleinopathies, implicating alpha-synuclein in the pathogenesis of synucleinopathy. [more]

PMID:18523712

Ancient Origin of the Parkinson Disease Gene LRRK2 . [more]

Figure 6.1.: Query for "gene cause of parkinson" in ALIBABA. Shown is only the core graph that includes the node for Parkinson's disease (blue border). Nodes with a red border are directly connected to Parkinson's disease, all others have at least one intermediary step. The upper right shows a list of all proteins connected to Parkinson's, according to the information found in 20 abstracts. The lower right shows the evidence for each relations, with links to the original texts and a link from each entity to a corresponding database for detailed information (middle right).

from PubMed. The server also handles the processing of text if annotations are not yet in the local cache. The processing falls into 'basic' natural language processing, named entity recognition, and relation extraction. NER includes word sense disambiguation (using SVM models, see Schiemann et al. [2008]). The server sends all annotated abstracts back to the client, which handles the parsing of annotations from XML and puts together the resulting graph.

6.1. Case studies

In this section we demonstrate how to use ALI BABA to find answers for questions typical to biomedical and related research. Such questions may refer to the formation of protein complexes, the regulation of gene expression, signaling cascades, metabolic pathways, or genetic linkage of diseases. Clinicians might seek for explanations of

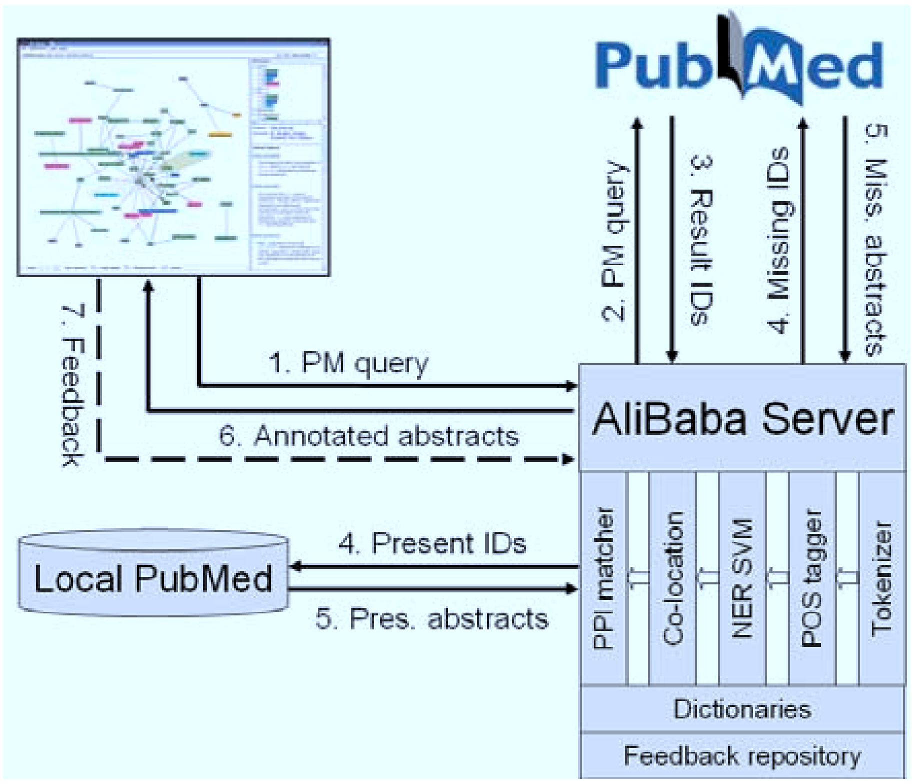


Figure 6.2.: System architecture and workflow in ALIBABA.

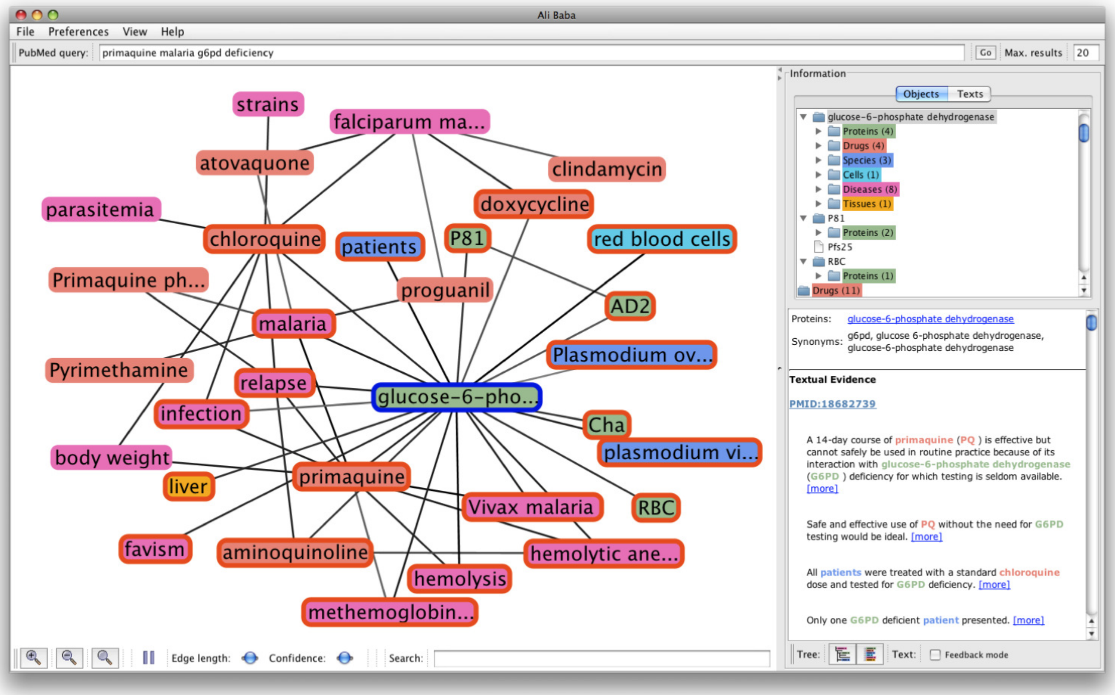
symptoms or for new therapies and drugs for diseases.

We chose the following case studies and show how to handle them ALI BABA:

- What are the risk factors of treating G6PD-deficient malaria patients with primaquine?
- Which proteins are involved in the activation or recognition mechanism for PmrD?
- Given the Wnt signaling pathway from KEGG, how can it be extended using the latest findings on the relevant protein *dickkopf* discussed in the literature?

What are the risk factors of treating G6PD-deficient malaria patients with primaquine?

Figure 6.3 shows the resulting graph for the PubMed query "primaquine malaria g6pd deficiency". The graph shows a connection between *G6PD* (deficiency), *vivax malaria* (patients), and their treatment with *primaquine*. The risk factors *hemolytic anemia/hemolysis* and *methemoglobinemia* are directly connected to *G6PD* and *primaquine*. Other nodes in the graph present more information on studied populations, the viral origin, infected cells, and information on the disease. Each object recognized by ALI BABA is linked to an appropriate external resource or database. In the upper part of the `Text view`, objects selected in either the graph or tree are shown by name and by all synonyms



PubMed query: primaquine malaria g6pd deficiency

Go Max. results 20

Proteins: glucose-6-phosphate dehydrogenase

Synonyms: g6pd, glucose 6-phosphate dehydrogenase, glucose-6-phosphate dehydrogenase

Textual Evidence

PMID:18682739

A 14-day course of primaquine (PQ) is effective but cannot safely be used in routine practice because of its interaction with glucose-6-phosphate dehydrogenase (G6PD) deficiency for which testing is seldom available. [more]

Safe and effective use of PQ without the need for G6PD testing would be ideal. [more]

All patients were treated with a standard chloroquine dose and tested for G6PD deficiency. [more]

Only one G6PD deficient patient presented. [more]

Figure 6.3.: AliBaba graph for query "primaquine malaria g6pd deficiency".

encountered in the abstracts. A mouse click on one of the names opens the database entry page for this particular object in the web browser. We select the node *primaquine* and click on its name, appearing as a link in the `Text view`, which brings us to its entry page in DrugBank, a comprehensive database that combines drug data and drug target information. We follow the same procedure for the protein *G6PD* and the disease *hemolytic anemia*, which opens the UniProt/SwissProt page and the entry page from the Medical Subject Headings of the National Library of Medicine, respectively. The three pages together with the network and annotated texts in ALI BABA offer a lot of background information on this particular question. ALI BABA can thus serve as a starting point for information retrieval in biology and biomedicine.

Which proteins are involved in the activation or recognition mechanism for PmrD?

To find an answer we first have to translate this question into the PubMed query language. In this case, we formulate the query as "PmrD (activation OR recognition)" and enter it into ALI BABA's query field located at the top of the application window. This results in a graph of nodes (representing entities) in different colors connected by edges (representing interactions and other associations) displayed on the large main panel – the `Graph view` (Figure 6.4). Pressing the `F9` key or the `Play/Pause` button on the bottom of the application window toggles the animation of the graph on or off.

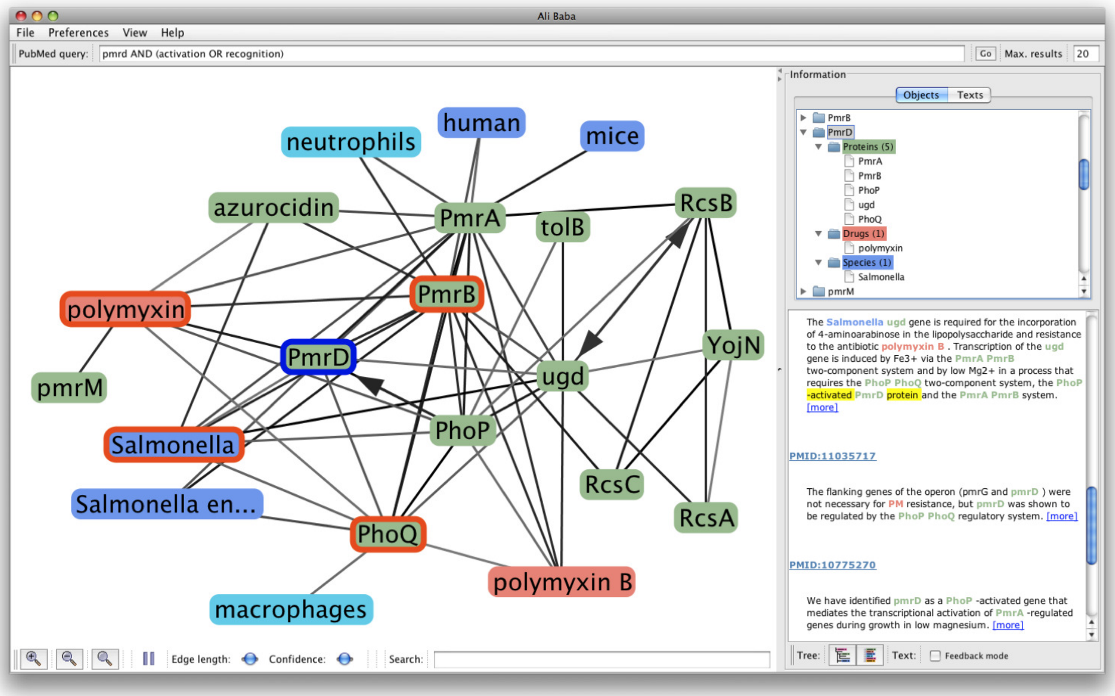
In the upper right hand panel (`Objects`) all entities are shown as a tree, sorted by their categories (protein, drug); each category has its unique color that is used throughout the ALI BABA application – proteins in green, diseases in pink, cells in light blue, tissues in orange, drugs in brown, and species are colored in blue. The number in brackets next to each category name tells us how many instances of this category were found in the PubMed result. A left click with the mouse on the node for *PmrD* in the graph also opens this protein in the tree and shows the different categories its direct neighbors in the graph belong to: proteins, drugs, species etc. These categories contain objects that are related to *PmrD*. In the lower right panel, all sentences from the PubMed abstracts that mention protein *PmrD* are shown in the `Text view`. Opening a category below *PmrD* in the tree and clicking on one of the objects listed shows all sentences from the abstracts that are evidence for a relation between this object and *PmrD*. The same effect can be seen when clicking on the edge between the two objects in the graph.

Proteins related to *PmrD* are *PhoP*, *PhoQ*, *ugd*, *PmrA*, and *PmrB*[2]. Reading the text snippets we can quickly verify that each of the proteins is indeed involved in either activation or recognition of *PmrD* or closely related to it. Figure 6.4 shows all proteins found in the texts. The selected node *PmrD* is highlighted with a blue and its direct neighbor nodes with a red border showing the answer to the question also graphically.

Extending the Wnt signaling pathway

The Wnt signaling pathway is a complex network of proteins well known for their roles in embryogenesis and cancer, but also involved in normal physiological processes in adult animals Lie et al. [2005]. To load the pathway for the first time, we open the `File menu` and select `Update KEGG files`. After the update process is complete we go to the File menu again and select `Open KEGG file`. In the next dialog, all available pathways are shown. The list is downloaded directly from the KEGG server and is therefore always up-to-date. Note that the list needs to be downloaded only once or when significant changes KEGG are expected. Selecting, for instance, the Wnt signaling pathway, will display the corresponding graph. KEGG pathways have a distinct layout that helps to get a better overview on the network. To view the pathway in its original presentation as known from KEGG, we open the `View menu` and choose the `Coordinates layout`.

Figure 6.5 shows the resulting Wnt signaling pathway of protein interactions. One of the proteins involved is *Dickkopf*, which functions as an inhibitor of the Wnt signaling pathway. The literature contains many references to this protein. From January to March 2008 about 30 articles have been published already that mention *Dickkopf* or one of its synonyms. To append a graph for *Dickkopf* from the literature to the Wnt signaling pathway from KEGG, we enable the option `Append queries` from the `Preferences menu` and then enter the query "dickkopf". The result is shown in Figure 6.6. The appended graph does not come with distinct layout and is thus just scrambled around the upper left of the pathway, but we can see clearly many new relations for protein *Dickkopf* and other proteins from the pathway. Furthermore, textual evidences are added



The Salmonella ugd gene is required for the incorporation of 4-aminoarabinose in the lipopolysaccharide and resistance to the antibiotic polymyxin B . Transcription of the ugd gene is induced by Fe3+ via the PmrA PmrB two-component system and by low Mg2+ in a process that requires the PhoP PhoQ two-component system, the PhoP -activated PmrD protein and the PmrA PmrB system. [more]

PMID:11035717

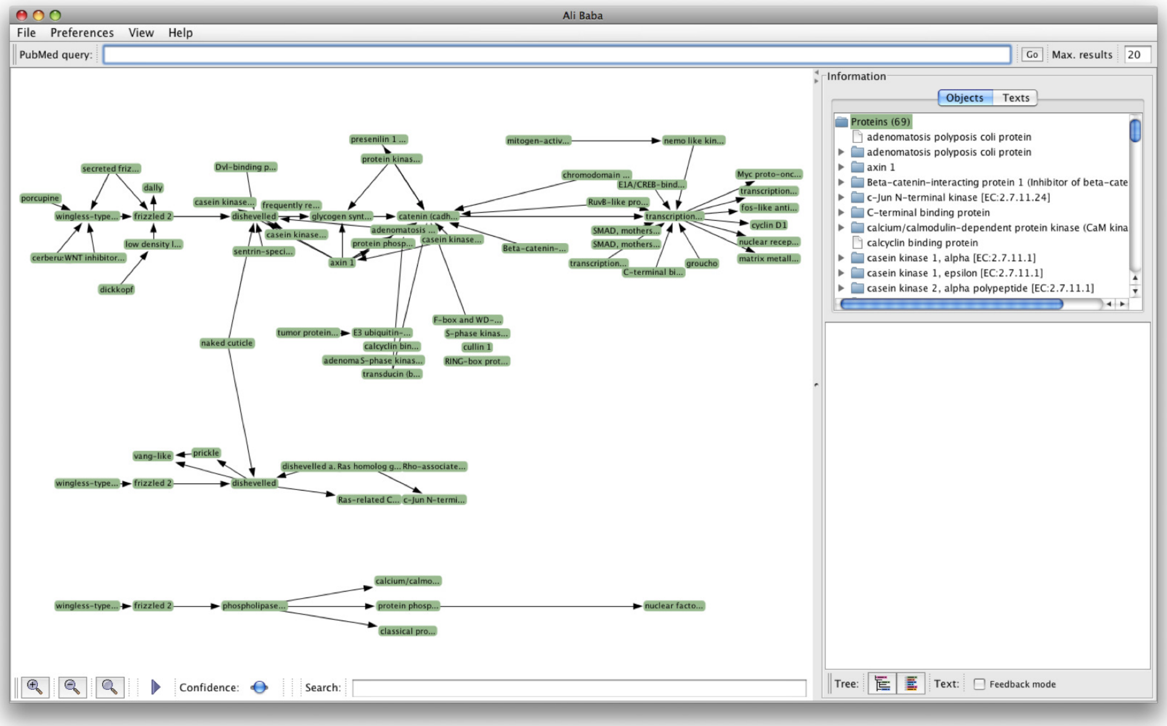
The flanking genes of the operon (pmrG and pmrD) were not necessary for PM resistance, but pmrD was shown to be regulated by the PhoP PhoQ regulatory system. [more]

PMID:10775270

We have identified pmrD as a PhoP -activated gene that mediates the transcriptional activation of PmrA -regulated genes during growth in low magnesium. [more]

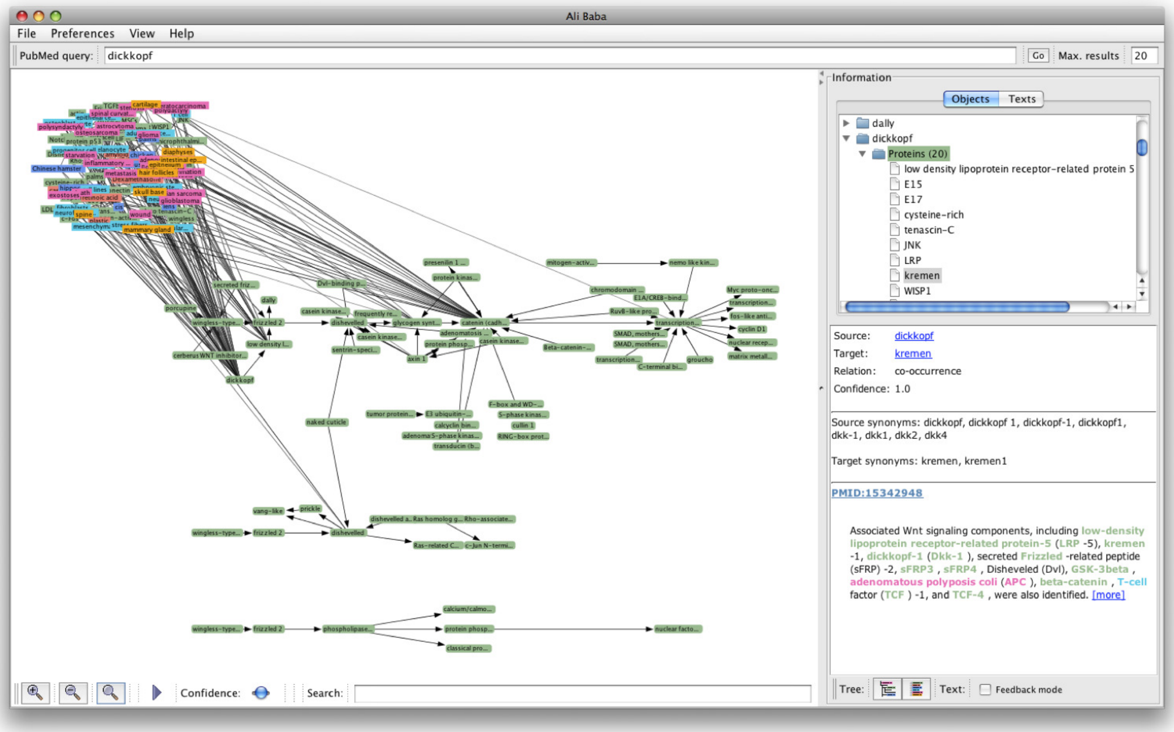
Figure 6.4.: Result for the PubMed query "PmrD (activation OR recognition)" as shown by Ali Baba. The selected node *PmrD* is framed in blue and all its direct neighbors in red.

to some of the pathway proteins and relations as they were found in the abstracts. This can be repeated with other queries to PubMed, though the resulting network will quickly become too complex to oversee. For such cases, users may either use the various `Display filter` options, or simply go to the `View` menu and switch to `Tree View only`, which hides the graph and shows only the tree (that contains the same information as the graph) and the texts.



Proteins (69)
adenomatosis polyposis coli protein
adenomatosis polyposis coli protein
axin 1
Beta-catenin-interacting protein 1 (Inhibitor of beta-cate
c-Jun N-terminal kinase [EC:2.7.11.24]
C-terminal binding protein
calcium/calmodulin-dependent protein kinase (CaM kina
calcyclin binding protein
casein kinase 1, alpha [EC:2.7.11.1]
casein kinase 1, epsilon [EC:2.7.11.1]
casein kinase 2, alpha polypeptide [EC:2.7.11.1]

Figure 6.5.: Wnt signaling pathway loaded into ALI BABA and presented using the original KEGG layout.



Source: dickkopf
Target: kremen
Relation: co-occurrence
Confidence: 1.0

Source synonyms: dickkopf, dickkopf 1, dickkopf-1, dickkopf1, dkk-1, dkk1, dkk2, dkk4

Target synonyms: kremen, kremen1

PMID:15342948

Associated Wnt signaling components, including low-density lipoprotein receptor-related protein-5 (LRP -5), kremen -1, dickkopf-1 (Dkk-1), secreted Frizzled -related peptide (sFRP) -2, sFRP3 , sFRP4 , Disheveled (Dvl), GSK-3beta , adenomatous polyposis coli (APC), beta-catenin , T-cell factor (TCF) -1, and TCF-4 , were also identified. [more]

Figure 6.6.: Graph from the literature query "dickkopf" appended to the Wnt signaling pathway. Most of these appended texts discuss entities (upper left) related to *dickkopf* and *catenin beta-1*, as indicated by the large number of edges towards either protein. Textual evidence for such relations can be viewed in the right panel, after selecting an entity node from the graph.

7. Summary and Outlook

7.1. Summary

In this thesis, we presented our approaches to a number of tasks in biomedical text mining, with the ultimate goal to extract relationships between entities from text. Biomedical text mining is concerned with searching for pieces of information in large text collections, summarizing them, allowing for collection-wide analyses, and help extract facts for further analysis. The numbers of entities, relationships, and (sub-)networks present in the literature dwarf information currently present in databases: compare the size of IntAct (May 2008), with less than 30,000 protein-protein interactions verified in small-scale experiments, to the number of almost 270,000 interactions contained in PubMed abstracts estimated by Donaldson et al. [2003], where interactions in abstracts mostly refer to small-scale experiments (but also include duplicate mentions, which decreases the total amount and increases reliability). This motivates the need to make them accessible despite their unstructured natural language representation, to researchers and database curators.

We presented goals, typical approaches, and challenges for each of the main building blocks in biomedical text mining. The building blocks in our overall approach fall into four categories: *i)* 'Basic' natural language processing components that handle tasks such as sentence splitting and part-of-speech tagging. *ii)* Named entity recognition finds mentions of biomedical entities (proteins, diseases) in text; focusing on recognition of protein names. *iii)* Extraction of relations between entities, focusing on finding protein-protein interactions. *iv)* We combined the approaches into a tool for visualization of networks extracted from the literature, AliBaba, as a first step in collection-wide analysis.

Both NER and relation extraction draw from output of the first component, namely POS tags. We use the TnT part-of-speech tagger to analyze text [Brants, 2000], and implemented methods for sentence splitting and tokenization. We developed two approaches for NER. The first transforms the *NER task into a classification problem*, deriving features from tokens and context windows; we use an SVM as classifier and add a post-expansion phase to find multi-word terms [Hakenberg et al., 2005a]. Our second approach to *NER builds on dictionaries*; starting from initial lexica drawn from databases specific to each particular class of entity, we developed a method for term expansion and *generation of spelling variations*. These variations are then encoded in a finite state automaton, provided by Monq [Kirsch et al., 2005]. We use the later approach for the nine entity types recognized by AliBaba, as the main advantage of dictionary-based approaches is their ability to instantly provide candidate identifiers for each mention found. For relation mining, our approaches build on pattern-matching; we developed two main methods, the first building on *patterns generated from a small training set and optimized using a genetic algorithm*. The second approach is split into two phases. We make use of information on relations in a database (in this case, IntAct for protein-protein interactions) and map them to a large text collection (PubMed abstracts). This step requires NER and entity mention normalization (EMN) and finds *candidate sentences that potentially discuss the sought relation* [Hakenberg et al., 2006]. These automatically labeled data can feed into subsequent methods to generate patterns. After filtering likely false positive sentences (mere coincidental mention of entities), we build a pairwise alignment library (PAL) to identify similar sentences. We transform this PAL into a guide list for clustering sentences by similarity. On each such cluster, we perform *multiple sentence alignment*, which identifies the commonalities of the sentences, represented in *consensus patterns*. Finally, we align these consensus pattern with new text to recognize instances of the relations encoded in the patterns [Hakenberg et al., 2008b,a].

We compare our approaches for NER and relation mining with methods presented by others. It shows that the most successful methods for named entity recognition currently build on sequential learning, conditional random fields the most prominent representative. Hsu et al. [2008] currently achieve an f–measure of 88.3% on the BioCreative 2 GM data, while our dictionary-based approach yields 73.1%. However, our approach implicitly handles the mapping of mention to identifiers, which allows us to use it as a building block in automatically collecting large training samples. While other approaches (using pattern-matching) for relation mining build on hand-crafted patterns or learn patterns from manually labeled data, we combine the collection of a training sample with an approach to learn patterns. On the BioCreative 2 IPS data, the current best system achieves 30.4% in f–measure, and our own approach yields 24.2%. On the SPIES corpus, these two systems result in of 68.2% and 63.9% f–measure, respectively. We see advantages for our own methodology when training samples do not exist, which is still the case for most kinds of relations of biomedical entities. However, it remains to be seen if methods like the one presented by Hsu et al. or Hunter et al. [2008] can draw from the collection of training samples we propose.

7.2. Contribution

The focus of this thesis is the extraction of interactions between proteins from text. As our main contribution, we present a method to automatically extract and refine sets of language patterns from unannotated data. We introduce the idea of multiple sentence alignment to find commonalities between similar patterns and express these in consensus patterns. As alignment strategy, we propose multi–layer alignment that takes into account not only part–of–speech tags (as other systems do), but also includes tokens and word stems to ensure precision.

As a prerequisite for subsequent components, we first introduce two methods for named entity recognition, focusing on protein and gene names (Chapter 3.) One is based on a model learned from training data and solves the general problem; the other works with a pre–defined dictionary and is useful when lists of named entities are known beforehand. The later approach also implicitly helps in solving the problem of mapping recognized entities to database identifiers, which is necessary for our subsequent components. We presented an approach for gene mention normalization, which is currently the best performing on the four BioCreative benchmarks, and extended it to the problem of inter-species gene mention normalization, a task for which no other methods have been proposed.

We apply sentence alignment as an inexact matching strategy to compute the similarity between two sentences. Most work shown previously transforms patterns into regular expressions for matching, resulting in binary decisions rather than quantitative measures.

We show a method to extract language patterns from unannotated data. Systems proposed previously were based on hand–crafted pattern sets or needed annotated data to extract patterns. As these initial patterns do not generalize well, we present a method to refine sets of language patterns to allow for larger coverage. We therefore introduce a clustering method based on sentence similarity. We introduce multiple sentence alignment (MSA) to find commonalities between groups of similar sentences. Such commonalities can then be expressed in consensus patterns. Using MSA, we are able to determine the fourth step in Hearst's (1992) methodology of pattern induction by repeatedly searching for instances of relations and generating new patterns from these.

To improve both ability to generalize and render more precise a set of patterns, we present an adapted implementation of alignment. We take into account not only a single type of information (such as amino acids for protein sequences, or part–of–speech tags for sentences), but incorporate multiple annotation layers (these are, part–of–speech tags, tokens, and word stems). On one hand, this method together with MSA increases the recall of patterns by drawing from multiple observations in a training sample. Precision is increased by restricting matches, as opposed to wild-cards or too general annotations such as POS tags only, but allowing for partial deviations, for instance in the token layer. Alignment as an inexact matching strategy allows —by setting thresholds on positional and overall scores and influencing the weight of individual layers— to tune our method to specific tasks.

7.3. Future directions

As natural language parsers become more and more reliable, a logical consequence is building a system based on sentence parsing rather than mere part-of-speech tagging. For initial parsing to get the phrase structure of a sentence, systems presented by Lease and Charniak [2005] and Bikel [2002] currently yield the best results (80.2 and 79.4% f–score for constituent trees), as evaluated by Clegg and Shepherd [2007] on the GENIA treebank. de Marneffe et al. [2006] proposed a method to generate dependency graphs from sentence parses. Both combinations yield an f–score for dependency graphs of 77.0% [Clegg and Shepherd, 2007]. Sætre et al. [2007] parse sentences with the Enju HPSG Yakushiji [2006] and finds the smallest connected parse tree that includes two interacting proteins. The system builds on patterns extracted from a training set, which has to be manually annotated for PPIs. Our method for collecting such sets might function as an input to a similar approach.

Tsochantaridis et al. [2005] and Kim et al. [2008a] presented kernel methods for structured input and output. While Tsochantaridis et al. apply their method to parse sentences and achieve and improvement over a probabilistic context-free grammar parser, Kim et al. predict relationships based on features drawn from parse tree information. The later system slightly outperforms an approach that builds on patterns over parse trees [Fundel et al., 2007]. Research in this direction is still ongoing, and more theoretical foundations and applications can be expected in the near future.

A useful extension of our method might be to compute profile hidden Markov models (Profile HMMs) to explain patterns in multiple sequence alignments [Eddy, 1998]. Such a method complements standard pairwise comparison methods, providing another approach for inexact matching by transforming the problem of matching into a sequence classification. In addition, profile HMMs output the probability of a sequence to match the given model (an MSA).

A logical extension of our approach to increase the speed of pattern alignment is the indexing of patterns to pre-select patterns. While the computation of a pattern set can occur offline, alignment as matching strategy is too complex to handle large sets of patterns on large text collections at runtime. Indexing may map patterns (and sentences) to a generalized sequence, for instance, restricting a pattern to fixed data types, such as POS tags for interaction-indicating words, named entities, and prepositions. An ongoing study led to an improvement in time performance by a factor of >10,000 (Palaga and Leser, data not shown). Sun et al. [2007b] pre-filter sentences by classification, grouping them into sentences that likely discuss a protein-protein interaction or not. When large enough training samples are available, an extension of this method could treat each pattern (or groups of patterns) as a category of its own, using the support of a pattern in the training sample, and classifying new sentences using these multiple labels.

Future text mining systems may also draw from efforts such as the BioCreative MetaService (BCMS; Leitner et al., 2008). BCMS is a framework to combine methods for the various sub-tasks in text mining, such as NER or EMN. Currently, BCMS governs twelve originally independent systems, all presented and evaluated in BioCreative 2007, and joins the predictions made by either tool into a single representation[1]. While in the past, building a text mining system for sophisticated analyses meant to first build each of the necessary sub-components from scratch (although some off-the-shelf solutions exist), it is now possible to use such services —taking advantage of the joint and thus oftentimes better predictions— for feeding into larger projects.

We are currently working on adopting our methodology to the extraction of other (protein-centered) types of associations. The overall approach already allows to swap the input we use to collect a labeled sample from protein-protein interaction databases to databases storing any other kinds of tuples. Many of these databases provide references for each entry to PubMed articles, potentially improving the accuracy of the

[1]See http://bcms.bioinfo.cnio.es

method to find initial sentences, as such referenced articles are more likely not to contain mere coincidental mentions. On the other hand, exploiting references narrows down the size of the input significantly. Initial studies showed that entity mention normalization becomes easier for some other types of entities (especially diseases, drugs and species). The first adaptation of our approach thus works on data from the Genetic Association Database (GAD, Becker et al. [2004]) to find associations between genes, drugs, and diseases (data not shown). Another source for the same kinds of relations is the Therapeutic Target Database (TTD, Chen et al. [2002]). Both include references to the literature and/or paraphrased summaries that can be used to extract initial patterns and, more importantly, function as a gold standard for evaluation.

Now that sufficiently well performing tools for extracting relations from literature become available, next tasks include the extraction of attributes and background information on each such relation [Krallinger et al., 2007]. For protein-protein interactions, users are interested in experimental conditions (which species, which cell line) and laboratory methods used for detection (Y2H, CoIP), see, for instance, Schmeier [2005].

As the analysis of large collections such as PubMed —roughly 10 million abstracts and 2.5 million open–access full–texts— leads to an abundance of extracted information, it becomes more and more important to find means to point users to significant facts. Reporting an association between the tumor suppressor protein "p53" and any type of neoplasms will hardly surprise any researcher. Still, text mining systems cannot afford ignore such highly repeated facts, because users will grow suspicious when such "basic facts" are missing. Golden nuggets might hide in hardly noted publications, or where information retrieval using keyword queries fails. Yet, information discussed in only a few abstracts might not always be correct, despite peer-review processes, and such errors might accumulate in significant amounts, as Krauthammer et al. [2002] have observed. Approaches for prioritization [of gene lists] have been discussed recently [Yu et al., 2008, Gonzalez et al., 2007], which may be used to rank extracted knowledge. In addition to verification, knowledge integration presents another challenge resulting from the availability of large amounts of automatically extracted data.

A. Data sets

A.1. Corpora

In this section, we provide information on the corpora used in our studies to evaluate the performance of named entity recognition and relationship extraction, as well as related corpora. A comparative analysis of five often-used corpora was presented by Pyysalo et al. [2008].

BioCreative 1, Task 1A — NER of gene names
Size: 15,000 sentences in three fixed parts for training, testing during development, and evaluation;
Split: training set contains 7500, development test set 2500, evaluation set 5000 sentences;
Annotations: ca. 55% sentences that contain at least one gene name; inline as POS-tags;
Note: Training and development test set used for the GM task in BioCreative 2 as well;
Reference: Yeh et al. [2005];
URL: http://biocreative.sourceforge.net

BioCreative 2, IPS — protein-protein interaction extraction
Size/split: 740 full text articles for training, 358 for test;
Annotations: stand-off on a per-document level, given as UniProt ID pairs;
Note: task includes NER, EMN, relevance ranking;
Reference: Krallinger et al. [2007];
URL: http://biocreative.sourceforge.net

SPIES — protein-protein interaction extraction
Size: 893 sentences, no fixed split into training/test;
Annotations: 1550 protein-protein interactions, no negative sentences, stand-off;
Reference: Hao et al. [2005];
URL: http://spies.cs.tsinghua.edu.cn — offline as of May 2008

LLL'05 — genic interaction extraction
Size/split: 55 sentences as training data, 87 as test;
Annotations: 103/52 interactions in training/test; stand-off; includes coreference information;
Reference: Nédellec [2005]
URL: http://genome.jouy.inra.fr/texte/LLLchallenge/

IEPA — protein-protein interactions
Size: 400 abstracts;

Annotations: stand-off, at least one interaction per abstract;
Reference: Berleant et al. [2003]
URL: http://class.ee.iastate.edu/berleant/home/me/cv/papers/corpuspropertiesstart.htm

DIPPPI — protein-protein interaction extraction
Size: 170 full-text articles in ASCII format
Annotations: 297 interactions for 165 articles; stand-off;
URL: http://www2.informatik.hu-berlin.de/~hakenber/corpora

BC1-PPI — protein-protein interactions
Size: 1000 sentences;
Annotations: 255 interactions in 173 sentences; stand-off;
Note: derived from BioCreative 1 task 1A data, additional annotations for PPIs
Reference: Plake et al. [2005]
URL: http://www2.informatik.hu-berlin.de/~hakenber/corpora

BioInfer — protein-protein interactions
Size: 836 abstracts, no fixed split into training/test;
Annotations: 2531 interactions in 1100 sentences;
Reference: Pyysalo et al. [2007]
URL: http://mars.cs.utu.fi/BioInfer

CGDCP — Cancer Gene Data Curation Project Size: various;
Entity types: genes, diseases, drug compound;
Relations: gene-disease and gene-drug compound; in two different data sets;
Annotations: plain text, stand-off (same file), negations
Note: comes in multiple phases, starting with a set of 1000 genes that have a moderate citation count (less than 1000 sentences in Medline), and ranging to a set that covers 4658 genes total;
URL: http://ncicb.nci.nih.gov/NCICB/projects/cgdcp

REmerge — Merging Relation Extraction resources This project aims at providing a common format for annotating entities and relations; multiple corpora have already been converted or are about to be converted into this format (LLL'05, HPRD50, BC1PPI, AIMed, BioInfer, GeneReg, GENIA, IEPA);
Annotations: currently focused on molecular interactions; XML, stand-off;
URL: http://remerge.wikidot.com

A.2. Databases and terminologies

In the following, we provide information on databases (such as IntAct for protein-protein interactions) and terminologies (such as the Gene Ontology) that were used in our experiments.

DIP — Database of Interacting Proteins
As a member of the IMEX Consortium (International Molecular Exchange), DIP adheres to standards shared by other similar databases; also, DIP cooperates with other protein-protein interaction databases (IntAct, MINT, etc.) to not curate redundant data.
Size: 57,683 interactions involving 20,728 proteins, taken from 65k experiments (Oct 2009);
Reference: Xenarios et al. [2001]
URL: http://dip.doe-mbi.ucla.edu

Entrez Gene — Database of genes from RefSeq genomes
Entrez Gene curates information on genes, including sequence, taxa, names and synonyms, chromosomal location, Gene Ontology annotations, gene products, functional descriptions found in PubMed abstracts (GeneRIFs), interactions, pathways, etc.
Size: 5,784,479 genes from 6584 taxa (Oct 2009); among those, 45,301 are human genes and loci;
URL: http://www.ncbi.nlm.nih.gov/sites/entrez/?db=gene

Gene Ontology — Standardization of gene and gene product annotations
The Gene Ontology (GO) is a collaborative effort to standardize descriptions of genes and gene products. GO is organized as an ontology, divided into the major categories: *biological process*, *molecular function*, and *cellular component*. Each concept contains a specific term, synonyms, definition, and further description, as well as links to related concepts. The actual annotation of genes and gene products is available via the AmiGO! project, and annotations are also curated in gene- and protein-centered databases such as EntrezGene and UniProt.
Size: GO contains 28,594 terms (17.5k for biological process, 8.6k for molecular function, 2.5k for cellular component; Oct 2009); currently, 376,451 gene products are annotated with GO terms;
Reference: Ashburner et al. [2000]
URL: http://www.geneontology.org

IntAct — Protein-protein interaction database
As a member of the IMEX Consortium (International Molecular Exchange), IntAct adheres to standards shared by other similar databases; also, IntAct cooperates with other protein-protein interaction databases (MINT, DIP, etc.) to not curate redundant data.
Size: 201,652 binary interactions involving 60k proteins, taken from 11k experiments (Oct 2009).
Reference: Hermjakob et al. [2004]

URL: http://www.ebi.ac.uk/intact

NCBI Taxonomy — Species organized in a taxonomic structure

The NCBI Taxonomy organizes taxa in a structured way, depicting the lineage from kingdoms to higher taxa, genus, species, and lower taxa. It also stores information on names and synonyms, and names used in common English, other languages, and colloquially.
Size: in total, there are 346,615 taxa organized in the taxonomy, relating to 212k species (Oct 2009);
URL: http://www.ncbi.nlm.nih.gov/sites/entrez?db=taxonomy

UniProt — Protein sequence and functional information

UniProt provides a wealth of sequence and functional annotations of proteins. Two parts of the UniProt knowledge base are available: Swiss-Prot, maintained by UniProt personnel, containing manually curated data; and TrEMBL, a public submission stage for automatic curation, which contains proteins and annotations that yet have to be verfied, curated, and possibly merged with SwissProt data. Thus, SwissProt can be considered the high-quality part in the UniProtKB. Annotations in UniProt include sequence and sequence features, isoforms, names and synonyms, Gene Ontology annotations, interactions, SNPs, domains, homology, associations with diseases, tissue specificity, catalytic activity, enzymatic reactions, encoding genes, among many others.
Size: UniProt/Swiss-Prot has 510,076 sequence entries; TrEMBL has 9,501,907 sequence entries (Oct 2009).
Reference: Bairoch et al. [2005]
URL: http://www.uniprot.org

A.3. Interaction-indicating words

In the language patterns and alignment algorithms, we use markup for certain words that indicate interactions between proteins. Note that the same words also might refer to interactions between all other types of molecules. These lists, presented in Table A.1 through 3, originated from a collection by Temkin and Gilder [2003], as well as a set used by Huang et al. [2004]. During our studies, we expanded them to 95 nouns, 129 verbs, and eight adjectives. The overall lists contain all conjugations for verbs, numbers for nouns, and adjectives with or without a trailing hyphen ("-mediated"). Hatzivassiloglou and Weng [2002] showed how to learn anchor verbs for biological interactions from scratch.

abolishment	catalyzator	down-regulator	mono-ubiquitination	replacement
abrogation	cluster	effect	multi-ubiquitination	repression
acceleration	complex	elevation	mutation	repressor
accelerator	complexes	expansion	obstruction	requirement
acceptor	conjugation	exposion	participation	sequestration
accumulation	control	formation	phosphorylation	stabilization
acetylation	conversion	inactivation	poly-ubiquitination	stimulation
activation	deconjugation	increase	precipitation	stimulator
activator	deconjugator	induction	production	substitution
activity	degradation	influence	promotion	sumoylation
addition	demethylation	inhibition	promotor	suppression
affection	dephosphorylation	inhibitor	proteolysis	suppressor
amplification	depletion	interaction	reaction	synthesis
assembly	destabilization	interactor	receptor	transacetylation
association	destruction	mediation	recognition	transactivation
attachment	detachment	methylation	recruitment	transcription
augmentation	disassembly	modification	reduction	ubiquitination
binding	disruption	modifier	regulation	up-regulation
bond	down-regulation	modulation	regulator	up-regulator

Table A.1.: Nouns that typically indicate interactions between proteins (and/or other molecules); shown are only the singular forms.

-dependent	-mediated	inhibitory	regulatory
-derived	-induced	-reactive	stimulatory

Table A.2.: Adjectives that typically indicate interactions between proteins (and/or other molecules).

abolish	comprise	encompass	link	repress
abrogate	conjugate	enhance	localize	require
absorb	consist	evoke	mediate	respond
accelerate	contact	exhibit	methylate	restrict
accept	contain	express	modify	severe
accumulate	control	form	modulate	stabilize
acetylate	deacetylate	fuse	mono-ubiquitinate	stain
activate	deconjugate	hasten	multi-ubiquitinate	stimulate
add	decrease	immunoblot	overexpress	substitute
affect	degrade	immunoprecipitate	participate	sumoylate
amplify	demethylate	immunoreact	phosphorylate	suppress
anchor	depend	impair	poly-ubiquitinate	synthesize
antagonize	dephosphorylate	inactivate	potentiate	target
arrest	deplete	incite	precipitate	tether
assemble	derive	include	prevent	transacetylate
assist	destabilize	increase	produce	transactivate
associate	destruct	induce	promote	transcribe
attach	detach	infect	react	transduce
augment	dimerize	influence	recognize	transfer
bind	direct	inhibit	recruit	transform
block	disassemble	initiate	reduce	treat
break	discharge	inject	regulate	trigger
catalyze	disrupt	interact	relate	ubiquitinate
cleave	down-regulate	interfere	remove	up-regulate
co-immunoprecipitate	elevate	interplay	repair	yield
coinfect	encode	ligate	replace	

Table A.3.: Verbs that typically indicate interactions between proteins (and/or other molecules); shown are infinitive forms only.

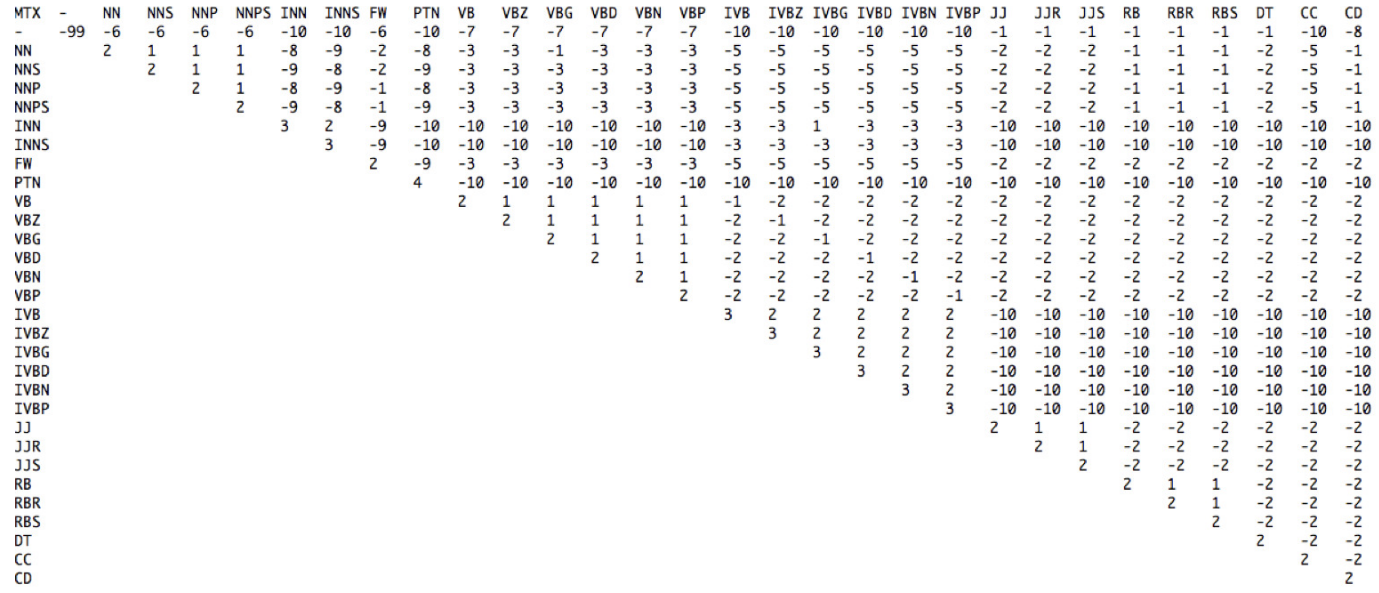
A.4. Part-of-speech tags

Brown tag	CIS tag	Description	Example
CC	cnj:K	coordinating conjunction	and
CD	adj:4	cardinal number	1, third
DT	det	determiner	the
FW	xinc	foreign word	d'hoevre
IN	prep	preposition/subordinating conjunction	in, of, like
JJ	adj:b	adjective	green
JJR	adj:c	adjective, comparative	greener
JJS	adj:s	adjective, superlative	greenest
MD	v:D:h	modal	could, will
NN	n:e	noun, singular or mass	table
NNS	n:m	noun plural	tables
NNP	en	proper noun, singular	John
NNPS	en:m	proper noun, plural	Vikings
PDT	cnj	predeterminer	both the boys
PRP	pron:q	personal pronoun	I, he, it
PRP$	det:o	possessive pronoun	my, his
RB	adv	adverb	however, usually, here, good
RBR	adv:c	adverb, comparative	better
RBS	adv:s	adverb, superlative	best
TO	infp	to	to go, to him
UH	intj	interjection	uhhuhhuhh
VB	v:I	verb, base form	take
VBD	v:V	verb, past tense	took
VBG	v:C	verb, gerund/present participle	taking
VBN	v:P	verb, past participle	taken
VBP	v:G	verb, sing. present, non-3d	take
VBZ	v:G3e	verb, 3rd person sing. present	takes
WDT	det:w	wh-determiner	which
WP	pron:w	wh-pronoun	who, what
WP$	pron:w	possessive wh-pronoun	whose
WRB	adv:w	wh-abverb	where, when
PTN		protein name	FADD, tumor necrosis factor α
IVB*, INN*, IJJ*		interaction-indicating words	binds, attachment, -mediated

Table A.4.: List of part-of-speech tags, in Brown notation as used throughout this thesis. In some related projects, we use the CIS notation (in which abbreviations originate from German words, thus 'n:m' refers to nouns in plural ['Mehrzahl']).

A.5. Substitution matrix for part-of-speech tags

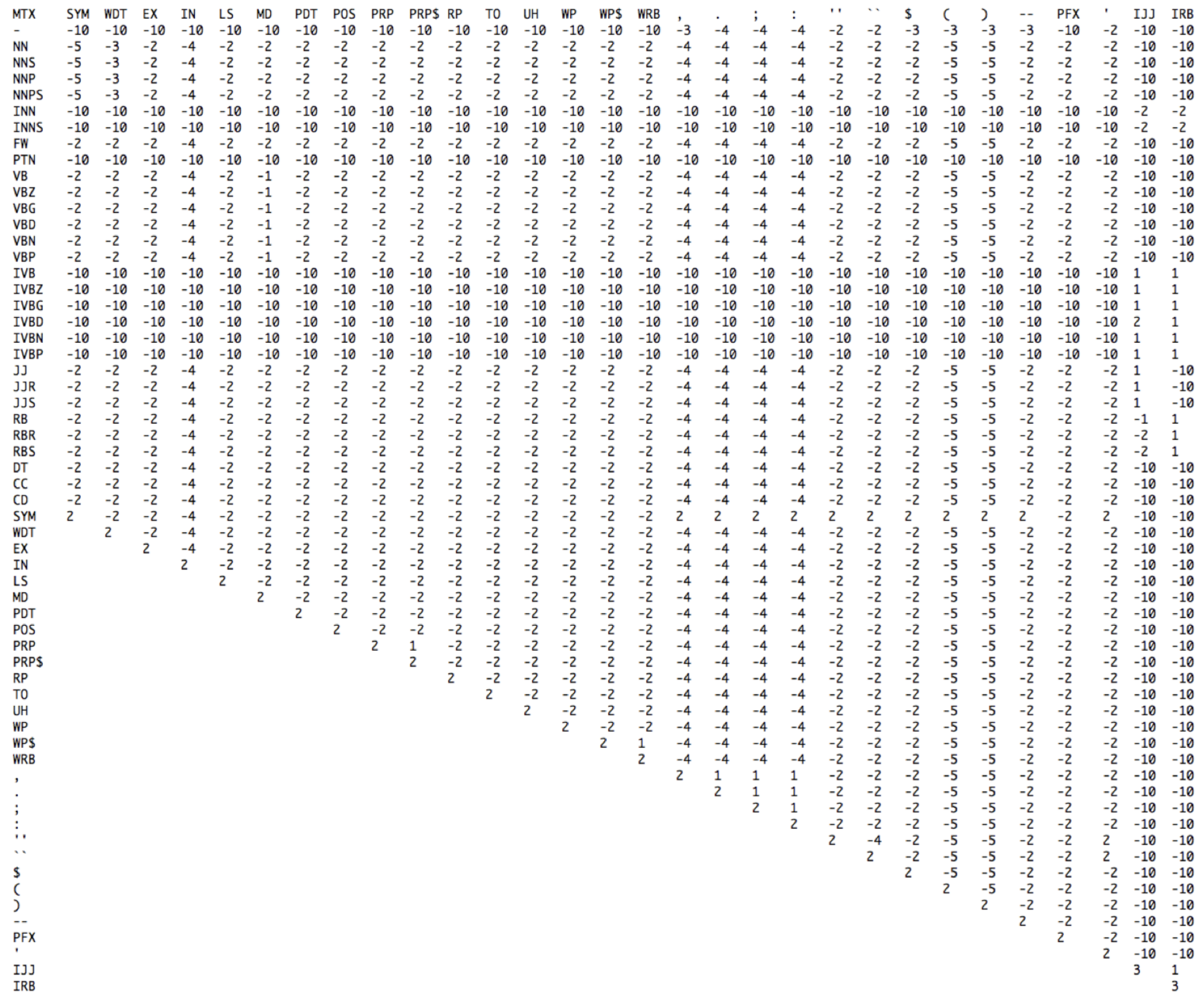
Figures A.1 and A.2 show the substitution matrix used in our experiments, split into two parts for readability. The cost of aligning a determiner (DT) with a gap ('-') is very small, as determiners can be removed/inserted without affected the meaning of a sentence much. Skipping a protein, however, often completely changes the meaning, so this assigned a high penalty (**-10**). Shifts in verb tense (VBZ↔VBD) slightly reduce the score (**1** instead of **2**). In the same manner, placement of prepositions is not arbitrary. Certain prepositions make sense (this also means, keeping the meaning of a pattern) only at certain positions within patterns. While the preposition in the pattern '`PROTEIN` binds to `PROTEIN`' can also be replaced by 'with', the preposition in 'activation of A by B' cannot be exchanged with 'and'. Usually, altering or skipping of prepositions has an effect of the meaning of a sentence (e.g., in passive forms: 'A is phosphorylated by B' versus 'A phosphorylated B' — note that here, the auxiliary verb is also missing).



MTX	-	NN	NNS	NNP	NNPS	INN	INNS	FW	PTN	VB	VBZ	VBG	VBD	VBN	VBP
-	-99	-6	-6	-6	-6	-10	-10	-6	-10	-7	-7	-7	-7	-7	-7
NN		2	1	1	1	-8	-9	-2	-8	-3	-3	-1	-3	-3	-3
NNS			2	1	1	-9	-8	-2	-9	-3	-3	-3	-3	-3	-3
NNP				2	1	-8	-9	-1	-8	-3	-3	-3	-3	-3	-3
NNPS					2	-9	-8	-1	-9	-3	-3	-3	-3	-3	-3
INN						3	2	-9	-10	-10	-10	-10	-10	-10	-10
INNS							3	-9	-10	-10	-10	-10	-10	-10	-10
FW								2	-9	-3	-3	-3	-3	-3	-3
PTN									4	-10	-10	-10	-10	-10	-10
VB										2	1	1	1	1	1
VBZ											2	1	1	1	1
VBG												2	1	1	1
VBD													2	1	1
VBN														2	1
VBP															2
IVB															
IVBZ															
IVBG															
IVBD															
IVBN															
IVBP															
JJ															
JJR															
JJS															
RB															
RBR															
RBS															
DT															
CC															
CD															

MTX	IVB	IVBZ	IVBG	IVBD	IVBN	IVBP	JJ	JJR	JJS	RB	RBR	RBS	DT	CC	CD
-	-10	-10	-10	-10	-10	-10	-1	-1	-1	-1	-1	-1	-1	-10	-8
NN	-5	-5	-5	-5	-5	-5	-2	-2	-2	-1	-1	-1	-2	-5	-1
NNS	-5	-5	-5	-5	-5	-5	-2	-2	-2	-1	-1	-1	-2	-5	-1
NNP	-5	-5	-5	-5	-5	-5	-2	-2	-2	-1	-1	-1	-2	-5	-1
NNPS	-5	-5	-5	-5	-5	-5	-2	-2	-2	-1	-1	-1	-2	-5	-1
INN	-3	-3	1	-3	-3	-3	-10	-10	-10	-10	-10	-10	-10	-10	-10
INNS	-3	-3	-3	-3	-3	-3	-10	-10	-10	-10	-10	-10	-10	-10	-10
FW	-5	-5	-5	-5	-5	-5	-2	-2	-2	-2	-2	-2	-2	-2	-2
PTN	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
VB	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
VBZ	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
VBG	-2	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
VBD	-2	-2	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
VBN	-2	-2	-2	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
VBP	-2	-2	-2	-2	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2
IVB	3	2	2	2	2	2	-10	-10	-10	-10	-10	-10	-10	-10	-10
IVBZ		3	2	2	2	2	-10	-10	-10	-10	-10	-10	-10	-10	-10
IVBG			3	2	2	2	-10	-10	-10	-10	-10	-10	-10	-10	-10
IVBD				3	2	2	-10	-10	-10	-10	-10	-10	-10	-10	-10
IVBN					3	2	-10	-10	-10	-10	-10	-10	-10	-10	-10
IVBP						3	-10	-10	-10	-10	-10	-10	-10	-10	-10
JJ							2	1	1	-2	-2	-2	-2	-2	-2
JJR								2	1	-2	-2	-2	-2	-2	-2
JJS									2	-2	-2	-2	-2	-2	-2
RB										2	1	1	-2	-2	-2
RBR											2	1	-2	-2	-2
RBS												2	-2	-2	-2
DT													2	-2	-2
CC														2	-2
CD															2

Figure A.1.: Substitution matrix used in our experiments for substituting POS tags during alignment; first half.



MTX	SYM	WDT	EX	IN	LS	MD	PDT	POS	PRP	PRP$	RP	TO	UH	WP	WP$	WRB	,	.	;	:	''	``	$	(	)	--	PFX	'	IJJ	IRB
-	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-3	-4	-4	-4	-2	-2	-3	-3	-3	-3	-10	-2	-10	-10
NN	-5	-3	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
NNS	-5	-3	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
NNP	-5	-3	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
NNPS	-5	-3	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
INN	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-2	-2
INNS	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-2	-2
FW	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
PTN	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10
VB	-2	-2	-2	-4	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
VBZ	-2	-2	-2	-4	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
VBG	-2	-2	-2	-4	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
VBD	-2	-2	-2	-4	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
VBN	-2	-2	-2	-4	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
VBP	-2	-2	-2	-4	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
IVB	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	1	1
IVBZ	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	1	1
IVBG	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	1	1
IVBD	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	2	1
IVBN	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	1	1
IVBP	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	1	1
JJ	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	1	-10
JJR	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	1	-10
JJS	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	1	-10
RB	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-1	1
RBR	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-2	1
RBS	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-2	1
DT	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
CC	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
CD	-2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
SYM	2	-2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	2	2	2	2	2	2	2	2	2	2	-2	2	-10	-10
WDT		2	-2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
EX			2	-4	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
IN				2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
LS					2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
MD						2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
PDT							2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
POS								2	-2	-2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
PRP									2	1	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
PRP$										2	-2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
RP											2	-2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
TO												2	-2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
UH													2	-2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
WP														2	-2	-2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
WP$															2	1	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
WRB																2	-4	-4	-4	-4	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
,																	2	1	1	1	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
.																		2	1	1	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
;																			2	1	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
:																				2	-2	-2	-2	-5	-5	-2	-2	-2	-10	-10
''																					2	-4	-2	-5	-5	-2	-2	2	-10	-10
``																						2	-2	-5	-5	-2	-2	2	-10	-10
$																							2	-5	-5	-2	-2	-2	-10	-10
(																								2	-5	-2	-2	-2	-10	-10
)																									2	-2	-2	-2	-10	-10
--																										2	-2	-2	-10	-10
PFX																											2	-2	-10	-10
'																												2	-10	-10
IJJ																													3	1
IRB																														3

Figure A.2.: Substitution matrix used in our experiments for substituting POS tags during alignment; second half.

B. Algorithms

B.1. Hidden Markov Models

Hidden Markov Models (HMMs) were introduced in the section on sequence learning, see page 29. Here, we present algorithms to solve the scoring, decoding, and estimation problems, respectively. First order hidden Markov models are fully described using:

- a finite set of states: $Q = \{q_i\}$,
- an output alphabet: $V = \{v_i\}$,
- start probabilities of being in state q_i at time $t=0$: $\Pi = \{\pi_i\}$,
- transition probabilities: $A = \{a_{ij}\}$

$$A = \{a_{ij} \mid a_{ij} = P(q_{t+1} = j | q_t = i)\},$$

- emission probabilities: $B = \{b_j(k)\}$

$$B = \{b_j(k) \mid b_j(k) = P(v_k \text{ at } t | q_t = j)\}$$

There are three key problems for hidden Markov models that need to be solved. In the *scoring problem*, the probability of the observed sequence given a model (also called *evaluation problem*) is computed. To compute the probability of an observed sequence, O, being emitted by a model M, we have to sum over all possible state sequences:

$$P(O|M) = \sum_{q_1,..,q_l \in Q^l} \prod_{t=1}^{l+1} a_{t,t+1} \cdot b_j(O_k), \tag{B.1}$$

where $q_1, .., q_l$ is the state sequence from 1 to l, and q_{l+1} is the end state. This problem is solved using the Forward-Backward algorithm (see later in this section).

The *decoding problem* involves the computation of a state sequence that best explains an observed sequence given a model (also called *alignment problem*). To recover the state sequence S that has the highest probability to produce the observed output, we have to solve

$$V(O|M) = \underset{q_1,..,q_l \in Q^l}{argmax} \prod_{t=1}^{l+1} a_{t,t+1} \cdot b_j(O_k), \tag{B.2}$$

This is computed using the Viterbi algorithm (see later).

Finally, the *estimation problem* deals with adjusting the parameters in a model to maximize the probability for the observed sequence (also called *training problem*). Expectation maximization algorithms, Baum-Welch, or Viterbi solve this problem.

Forward-backward and Baum-Welch

The direct approach for calculating the probability of an observation sequence given a model would be a simple enumeration of all state sequences having the same length. This would require, however, $O(N * T^N)$ (where $T = |O|$ is the number of observations l and $N = |Q|$ the number of states in the model). The Forward-backward algorithm solves this problem in $O(N^2 * T)$ using a dynamic programming approach [Rabiner, 1989].

Forward-backward introduces a forward variable, $\alpha_t(i)$, that stores the probability of observing the partial sequence $O_1, .., O_t$ and being in state i at step t ($t \leq T$). Recursively, Forward-backward computes the $\alpha_t(j)$ for every partial observation sequence and all states j.

$\boxed{\text{Forward-Backward}}$

1. Initialization:$\alpha_1(i) = \pi_i b_i(O_1), \;\; 1 \leq i \leq N$

2. Recursion:for $t = 1,..,T–1, \;\; 1 \leq j \leq N$:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\alpha_{ij} \right] b_j(O_{t+1})$$

3. Termination:$P(O|M) = \sum_{i=1}^{N} \alpha_T(i)$

Based on the Forward-backward procedure, Baum-Welch re-estimates the HMM parameter values Π, A, and B in an additional step [Baum et al., 1970]. It calculates the frequency of each transition/emission pair. Divided by the overall probability of the given sequences, this results in new expected values for each transition/emission pair. The probability to be in state q_i at time t is

$$\gamma_t(i) = P(i_t = q_i | O, M) \tag{B.3}$$
$$= \alpha_t(i)\beta_t(i) \;/\; P(O|M), \tag{B.4}$$

and the probability to be in state q_i and make a transition to state q_j from time t to time $t+1$ is

$$\xi_t(i,j) = P(i_t = q_i, \; i_{t+1} = q_j) \tag{B.5}$$
$$= \alpha_t(i) a_{ij} \beta_{t+1}(j) b_j(O_{t+1}) \;/\; P(O|M) \tag{B.6}$$

To re-estimate Π, A, and B, we use the Baum-Welch formulas

$$\Pi' = \{\pi_i = \gamma_1(i),\ 1 \le i \le N\} \tag{B.7}$$

$$A' = \{a_{i,j} = \sum_{t=1}^{T-1} \xi_t(i,j) \ / \ \sum_{t=1}^{T-1} \gamma_t(i)\} \tag{B.8}$$

$$B' = \{b_j(k) = \sum_{t=1, O_t=k}^{T} \gamma_t(j) \ / \ \sum_{t=1}^{T} \gamma_t(j)\} \tag{B.9}$$

As an example, let us consider a set of multiple sequences, for which we search the best model (parameters) to explain all sequences. We can see the evaluation problem also as a scoring problem. Given one model, we score each sequence by computing its respective probability to have been generated by the model. This results in an overall likelihood for all sequences to originate from the given model. Among all scorings, we pick the model that best explains all sequences.

Expectation maximization

In Expectation maximization (EM) is a general framework for a group of related algorithms. In a parameterized probabilistic model, EM solves the problem of finding maximum likelihood estimates for all parameters. This is achieved by alternating between two steps. (i) The expectation step (E) estimates the respective expected values of the latent variables. (ii) The maximization step (M) calculates the maximum likelihood estimates of all parameters (according to some training data), satisfying the expected values of the latent variables. The next E step then uses the parameters from the M step, which again computes an expectation of the likelihoods.

For a probability model p, the parameters are given in a vector $\Theta : p(y, x|\Theta)$. Expectation maximization starts with an initial estimate for Θ, Θ_0. Each new estimation is derived using

$$\Theta_{n+1} = \underset{\Theta}{argmax}\, Q(\Theta), \tag{B.10}$$

where the expectation $Q(\Theta)$ is

$$Q(\Theta) = \sum_x p(x, y|\Theta_n) \log p(y, x|\Theta). \tag{B.11}$$

Apart from estimating parameters for hidden Markov models, EM is applied to solve a variety of other problems, for example unsupervised learning (clustering) [Mak et al., 2004] or fitting mixture density models [Aitkin and Rubin, 1985]. The Baum-Welch algorithm is one special case of expectation maximization.

Viterbi

In contrast to Forward-Backward, the Viterbi algorithm calculates the most likely path in a model, together with its probability [Viterbi, 1967]. That is, is determines the alignment of a sequence with the model.

$\boxed{\text{Viterbi}}$

$$
\begin{aligned}
\textbf{1. Initialization}:&\delta_1(i) = \pi_i b_i(O_1),\ \ 1 \le i \le N\\
&\psi_1(i) = 0\\
\textbf{2. Recursion}:&\text{for } t=2,..,T,\ 1 \le j \le N:\\
&\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(O_t)\\
&\psi_t(j) = \underset{1 \le i \le N}{argmax} \left[\delta_{t-1}(i) a_{ij}\right]\\
\textbf{3. Termination}:&P(O|M) = \sum_{i=1}^{N} \alpha_T(i)\\
\textbf{4. Backtracking}:&P(O|M) = \sum_{i=1}^{N} \alpha_T(i)
\end{aligned}
$$

Viterbi typically is used to solve the training problem, that is, to estimate the parameters of a model so that it best explains a set of given sequences, called training data. Solving this problem is important for most applications of hidden Markov models.

The algorithmic complexity for training and applying hidden Markov models is the same for both Baum-Welch and Viterbi algorithms. The worst-case time complexity is $O(NM^2)$, and storage complexity is $O(NM)$ (where N is the length of the sequence and M is the number of states in the HMM). More storage efficient implementations of HMMs have been proposed that need only $O(M)$ or $O(M^{1.5})$, respectively [Hughey and Krogh, 1996, Tarnas and Hughey, 1998].

Abbreviations

ASO alternating structure optimization
BC BioCreAtIvE
CCG combinatory categorial grammar
CP consensus pattern
CRF conditional random field
CWA collection–wide analysis
EM expectation maximization
EMN entity mention normalization
FSA finite state automaton
HMM hidden Markov model
IDF inverse document frequency
IE information extraction
IR information retrieval
KD knowledge discovery
kNN k-nearest neighbor
MSA multiple sequence/sentence alignment
NEI named entity identification
NER named entity recognition
NLP natural language processing
POS part–of–speech (tag)
PPI protein–protein interaction
RBF radial basis function
SMO sequential minimal optimization
SVM support vector machine
TF term frequency
UMLS Unified Medical Language System
UPGMA unweighted pair–group methods using the arithmetic average
WSD word sense disambiguation
WSJ Wall Street Journal (˜corpus)
Y2H yeast two-hybrid (˜screen,˜experiment)

Bibliography

M. Aitkin and D. B. Rubin. Estimation and hypothesis testing in finite mixture models. *Journal of the Royal Statistical Society B*, 47(1):67–75, 1985.

M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25: 821–837, 1964.

Russ B. Altman and Teri E. Klein. Challenges for biomedical informatics and pharmacogenomics. *Annual Review of Pharmacology and Toxicology*, 42:113–133, 2002. doi: 10.1146/annurev.pharmtox.42.082401.140850.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–10, 1990.

S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.*, 25:3389–3402, 1997.

Rie Kubota Ando. Biocreative ii gene mention tagging system at ibm watson. In *Proc 2nd BioCreative Challenge Evaluation Workshop*, pages 101–103, 2007.

Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25: 25–29, 2000.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

Amos Bairoch, Rolf Apweiler, Cathy H. Wu, Winona C. Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, Maria J. Martin, Darren A. Natale, Claire O'Donovan, Nicole Redaschi, and Lai-Su L. Yeh. The universal protein resource (uniprot). *Nucleic Acids Res.*, 33 (Database issue):D154–D159, Jan 2005.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of the Mathematical Statistics*, 41:164–171, 1970.

William A. Baumgartner, Zhiyong Lu, Helen L. Johnson, J. Gregory Caporaso, Jesse Paquette, Anna Lindemann, Elizabeth K. White, Olga Medvedeva, K. Bretonnel Cohen, and Lawrence Hunter. An integrated approach to concept recognition in biomedical text. In *Proc 2nd BioCreative Challenge Evaluation Workshop*, pages 257–271, 2007.

Kevin G. Becker, Kathleen C. Barnes, Tiffany J. Bright, and Alex Wang. The genetic association database. *Nature Genetics*, 36:431–432, 2004.

D. A. Benson, M. S. Boguski, D. J. Lipman, J. Ostell, and B. F. Ouellette. Genbank. *Nucleic Acids Res.*, 26:1–7, 1998.

Daniel Berleant, Jing Ding, and Andy W. Fulmer. Corpus properties of protein interaction descriptions in medline. unpublished, 2003.

Daniel M. Bikel. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proc Human Language Technology Conference 2002 (HLT2002)*, San Diego, 2002.

Mikhail V. Blagosklonny and Arthur B. Pardee. Unearthing the gems. *Nature*, 416:373, 2002.

Christian Blaschke and Alfonso Valencia. The frame-based module of the suiseki information extraction system. *IEEE Intelligent Systems*, 17:14–20, 2002.

Christian Blaschke, Eduardo Andres Leon, Martin Krallinger, and Alfonso Valencia. Evaluation of biocreative assessment of task 2. *BMC Bioinformatics*, 6(Suppl. 1):S16, 2005.

Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology, ISMB*, Detroit, USA, 2005.

Tewis Bouwmeester, Angela Bauch, Heinz Ruffner, Pierre-Olivier Angrand, Giovanna Bergamini, Karen Croughton, Cristina Cruciat, Dirk Eberhard, Julien Gagneur, Sonja Ghidelli, Carsten Hopf, Bettina Huhse, Raffaella Mangano, Anne-Marie Michon, Markus Schirle, Judith Schlegl, Markus Schwab, Martin A. Stein, Andreas Bauer, Georg Casari, Gerard Drewes, Anne-Claude Gavin, David B. Jackson, Gerard Joberty, Gitte Neubauer, Jens Rick, Bernhard Kuster, and Giulio Superti-Furga. A physical and functional map of the human tnf-alpha/nf-kappab signal transduction pathway. *Nat Cell Biol*, 6(2):97–105, 2004.

Thorsten Brants. Tnt – a statistical part-of-speech tagger. In *Proc. Conf. on Applied Natural Language Processing, ANLP*, Seattle, USA, 2000.

Eric Brill. A simple rule-based part of speech tagger. In *Proc. Conf. on Applied Natural Language Processing, ANLP*, pages 152–155, Trento, Italy, 1992.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Razvan Bunescu, Raymond Mooney, Arun Ramani, and Edward Marcotte. Integrating co-occurrence statistics with information extraction for robust retrieval of protein interactions from medline. In *ACL Workshop*, 2006.

Evelyn Camon, Michele Magrane, Daniel Barrell, Vivian Lee, Emily Dimmer, John Maslen, David Binns, Nicola Harte, Rodrigo Lopez, and Rolf Apweiler. The gene ontology annotation (goa) database: sharing knowledge in uniprot with gene ontology. *Nucleic Acids Res.*, 32(Database issue):D262–D266, Jan 1 2004.

Evelyn B. Camon, Daniel G. Barrell, Emily C. Dimmer, Vivian Lee, Michele Magranea, John Maslen, David Binns, and Rolf Apweiler. An evaluation of go annotation retrieval for biocreative and goa. *BMC Bioinformatics*, 6(Suppl 1):S17, 2005.

Jeffrey T. Chang, Hinrich Schütze, and Russ B. Altman. Gapscore: finding gene and protein names one word at a time. *Bioinformatics*, 20(2):216–225, 2004.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc ACL*, pages 173–180, Ann Arbor, USA, June 2005.

Andrew Chatr-aryamontri, Arnaud Ceol, Luisa Montecchi Palazzi, Maria Victoria Schneider Giuliano Nardelli, Luisa Castagnoli, and Gianni Cesareni. Mint: the molecular interaction database. *Nucleic Acids Research*, 35(Database issue):D572–D574, 2007. doi: 10.1093/nar/gkl950.

J. Chen, R. Chau, and C. Yeh. Discovering parallel text from the world wide web. In *Proc ACM Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, pages 157–161, 2004.

Kuang-Hua Chen and Hsin-Hsi Chen. A probabilistic chunker. In *ROCLING-93*, 1993.

Lifeng Chen, Hongfang Liu, and Carol Friedman. Gene name ambiguity of eukaryotic nomenclatures. *Bioinformatics*, 21:248–256, 2005.

Xin Chen, Zhiliang Ji, and Yuzong Z. Chen. Ttd: Therapeutic target database. *Nucleic Acids Res.*, 30(1):412–415, 2002.

Yizong Cheng and George M. Church. Biclustering of expression data. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology, ISMB*, pages San Diego, USA, 2000.

J. M. Cherry. Genetic nomenclature guide. saccharomyces cerevisiae. *Trends in Genetics*, pages 11–12, 1995.

Andrew B. Clegg and Adrian J. Shepherd. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8:24, 2007. doi: 10.1186/1471-2105-8-24.

Michael Collins. Discriminative reranking for natural language parsing. In *ICML*, pages 175–182, Standford, USA, 2000. Morgan Kaufmann, San Francisco, CA.

John S. Conery. Aligning sequences by minimum description length. *EURASIP Journal on Bioinformatics and Systems Biology*, 2007. doi: 10.1155/2007/72936.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

Nikolai Daraselia, Anton Yuryev, Sergej Egorov, Svetlana Novichkova, Alexander Nikitin, and Ilya Mazo. Extracting human protein interactions from medline using a full-sentence parser. *Bioinformatics*, 20(5):604–611, 2004.

Marie-Catherine de Marneffe, Bill MacCartney, , and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proc LREC*, pages 449–454, Genoa, Italy, 2006.

Charlotte M. Deane, Łukasz Salwinski, Ioannis Xenarios, and David Eisenberg. Protein interactions. *Molecular & Cellular Proteomics*, 1:349–356, 2002.

Jing Ding, Daniel Berleant, Dan Nettleton, and Eve S. Wurtele. Mining medline: Abstracts, sentences, or phrases? In *Proc. Pac. Symp. Biocomput., PSB*, pages 326–337, Kaua'i, Hawaii, USA, Jan. 3-7 2002.

Ian Donaldson, Joel Martin, Berry de Bruijn, Cheryl Wolting, Vicki Lay, Brigitte Tuekam, Shudong Zhang, Berivan Baskin, Gary D. Bader, Katerina Michalickova, Tony Pawson, and Christopher W.V. Hogue. Prebind and textomy – mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, 4:11, 2003.

Sean R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

J.T. Eppig, C.J. Bult, J.A. Kadin, J.E. Richardson, J.A. Blake, A. Anagnostopoulos, R.M. Baldarelli, M. Baya, J.S. Beal, S.M. Bello, W.J. Boddy, D.W. Bradt, D.L. Burkart, N.E. Butler, J. Campbell, M.A. Cassell, L.E. Corbani, S.L. Cousins, D.J. Dahmen, H. Dene, A.D. Diehl, H.J. Drabkin, K.S. Frazer, P. Frost, L.H. Glass, C.W. Goldsmith, P.L. Grant, M. Lennon-Pierce, J. Lewis, I. Lu, L.J. Maltais, M. McAndrews-Hill, L. McClellan, D.B. Miers, L.A. Miller, L. Ni, J.E. Ormsby, D. Qi, T.B. Reddy, D.J. Reed, B. Richards-Smith, D.R. Shaw, R. Sinclair, C.L. Smith, P. Szauter, M.B. Walker, D.O. Walton, L.L. Washburn, T.T. Witham, and Y. Zhu; Mouse Genome Database Group. The mouse genome database (mgd): from genes to mice–a community resource for mouse biology. *Nucleic Acids Res.*, 33(Database issue):D471–D475, 2005.

T.A. Eyre, F. Ducluzeau, T.P. Sneddon, S. Povey, E.A. Bruford, and M.J. Lush. The hugo gene nomenclature database, 2006 updates. *Nucleic Acids Res.*, 34(Database issue): D319–D321, 2006.

Jenny Finkel, Shipra Dingare, Huy Nguyen, Malvina Nissim, Gail Sinclair, and Christopher D. Manning. Exploiting context for biomedical named entity recognition: From syntax to the web. In *Proc. Joint Workshop on Natural Language Processing in Biomedicine and its Applications, JNLPBA*, pages 88–91, 2004.

W. Nelson Francis. A standard sample of present-day english for use with digital computers. *Report to the U.S. Office of Education on Cooperative Research Project*, No. E-007, 1964.

Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Lidén, and Joakim Cöster. Protein names and how to find them. *International Journal of Medical Informatics*, 67(1-3):49–61, Dec 2002.

Carol Friedman, Pauline Kra, Hong Yu, Michael Krauthammer, and Andrey Rzhetsky. Genies: A natural-language processing system for the extraction of molecular pathways from biomedical texts. *Bioinformatics*, 17(Suppl. 1):S74–82, 2001.

Christoph M. Friedrich, Thomas Revillion, Martin Hofmann, and Juliane Fluck. Biomedical and chemical named entity recognition with conditional random fields: The advantage of dictionary features. In *Proc. Int. Symp. on Semantic Mining in Biomedicine, SMBM*, pages 85–88, Jena, Germany, April 9-12 2006.

Ken'ichiro Fukuda, Tatsuhiko Tsunoda, Ayuchi Tamura, and Toshihisa Takagi. Toward information extraction: Identifying protein names from biological papers. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 705–716, 1998.

Katrin Fundel and Ralf Zimmer. Gene and protein nomenclature in public databases. *BMC Bioinformatics*, 7(7):372, 2006. doi: 10.1186/1471-2105-7-372.

Katrin Fundel, Daniel Güttler, Ralf Zimmer, and Joannis Apostolakis. A simple approach for protein name identification: prospects and limits. *BMC Bioinformatics*, 6 (Suppl 1):S15, 2005. 10.1186/1471-2105-6-S1-S15.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007. doi: 10.1093/bioinformatics/btl616.

R. Gaizauskas, G. Demetriou, P. J. Artymiuk, and P. Willett. Protein structures and information extraction from biological texts: The pasta system. *Bioinformatics*, 19(1): 135–143, 2003.

W. Gale, K. Church, and D. Yarowsky. One sense per discourse. In *Proc. DARPA Speech and Natural Language Workshop*, pages 233–237, 1992.

Sylvain Gaudan, Harald Kirsch, and Dietrich Rebholz-Schuhmann. Resolving abbreviations to their senses in medline. *Bioinformatics*, 21(18):3658–3664, 2005.

Gene Ontology Consortium. The gene ontology (go) project in 2006. *Nucleic Acids Res.*, 34:D322–D326, 2006.

L. Giot, J.S. Bader, C. Brouwer, A. Chaudhuri, B. Kuang, Y. Li, Y.L. Hao, C.E. Ooi, B. Godwin, E. Vitols, G. Vijayadamodar, P. Pochart, H. Machineni, M. Welsh, Y. Kong, B. Zerhusen, R. Malcolm, Z. Varrone, A. Collis, M. Minto, S. Burgess, L. McDaniel,

E. Stimpson, F. Spriggs, J. Williams, K. Neurath, N. Ioime, M. Agee, E. Voss, K. Furtak, R. Renzulli, N. Aanensen, S. Carrolla, E. Bickelhaupt, Y. Lazovatsky, A. DaSilva, J. Zhong, C.A. Stanyon, Jr. R.L. Finley, K.P. White, M. Braverman, T. Jarvie, S. Gold, M. Leach, J. Knight, R.A. Shimkets, M.P. McKenna, J. Chant, and J.M. Rothberg. A protein interaction map of drosophila melanogaster. *Science*, 302(5651):1727–1736, Dec 3 2003.

Graciela Gonzalez, Juan C Uribe, Luis Tari, Colleen Brophy, and Chitta Baral. Mining gene-disease relationships from biomedical literature: weighting protein-protein interactions and connectivity measures. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 28–39, 2007. ISSN 1793-5091. doi: 17992743. PMID: 17992743.

R. Gould. Multiple correspondence. *Mechanical Translation*, 4(1/2):14–27, 1957.

B.B. Greene and G.M. Rubin. Automatic grammatical tagging of english. Technical report, Department of Linguistics, Brown University, Providence, RI, 1981.

Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir N. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46 (1-3):389–422, 2002.

Udo Hahn and Joachim Wermter. Levels of natural language processing for text mining. In Sophia Ananiadou and John McNaught, editors, *Text Mining in Biology*. Artech House Books, 2006.

Jörg Hakenberg, Steffen Bickel, Conrad Plake, Ulf Brefeld, Hagen Zahn, Lukas Faulstich, Ulf Leser, and Tobias Scheffer. Systematic feature evaluation for gene name recognition. *BMC Bioinformatics*, 6(Suppl 1):S9, 2005a.

Jörg Hakenberg, Conrad Plake, Ulf Leser, Harald Kirsch, and Dietrich Rebholz-Schuhmann. Lll'05 challenge: Genic interaction extraction with alignments and finite state automata. In *Proc Learning Language in Logic Workshop (LLL05) at the 22nd Int Conf on Machine Learning*, Bonn, Germany, August 2005b.

Jörg Hakenberg, Ulf Leser, Harald Kirsch, and Dietrich Rebholz-Schuhmann. Collecting a large corpus from all of medline. In *Proc. Int. Symp. on Semantic Mining in Biomedicine, SMBM*, pages 89–92, Jena, Germany, April 9-12 2006.

Jörg Hakenberg, Loic Royer, Conrad Plake, Hendrik Strobelt, and Michael Schroeder. Me and my friends: gene mention normalization with background knowledge. In Lynette Hirschman, Martin Krallinger, and Alfonso Valencia, editors, *Proc 2nd BioCreative Challenge Evaluation Workshop*, pages 141–144, Madrid, Spain, April 23-25 2007.

Jörg Hakenberg, Conrad Plake, Robert Leaman, Michael Schroeder, and Graciela Gonzalez. Inter–species normalization of gene mentions with gnat. *Bioinformatics*, 24, September 2008a. doi: 10.1093/bioinformatics/btn299. To appear.

Jörg Hakenberg, Conrad Plake, Loic Royer, Hendrik Strobelt, Ulf Leser, and Michael Schroeder. Gene mention normalization and interaction extraction with context models and sentence motifs. *Genome Biology*, 9(S2):S14, 2008b.

Daniel Hanisch, Juliane Fluck, Heinz Theodor Mevissen, and Ralf Zimmer. Playing biology's namen game: Identifying protein names in scientific text. In Russ B. Altman, A. K. Dunker, Lawrence Hunter, T. A. Jung, and T. E. Klein, editors, *Proc Pac Symp Biocomput*, pages 403–414, 2003.

Daniel Hanisch, Katrin Fundel, Heinz Theodor Mevissen, Ralf Zimmer, and Juliane Fluck. Prominer: Organism-specific protein name detection using approximate string matching. In *BioCreAtIvE Workshop*, Granada, Spain, March 2004.

Daniel Hanisch, Katrin Fundel, Heinz Theodor Mevissen, Ralf Zimmer, and Juliane Fluck. Prominer: rule-based protein and gene entity recognition. *BMC Bioinformatics*, 6(Suppl 1):S14, 2005.

Yu Hao, Xiaoyan Zhu, Minlie Huang, and Ming Li. Discovering patterns to extract protein-protein interactions from the literature: Part ii. *Bioinformatics*, 21(15):3294–3300, 2005.

Kenneth E. Harper. Semantic ambiguity. *Mechanical Translation*, 4(3):68–69, 1957a.

Kenneth E. Harper. Contextual analysis. *Mechanical Translation*, 4(3):70–75, 1957b.

J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

Vasileios Hatzivassiloglou and Wubin Weng. Learning anchor verbs for biological interaction patterns from published text articles. *Int. J. Med. Inform.*, 67:19–32, 2002.

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. COLING*, pages 539–545, Nantes, August 23–28 1992.

S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci., PNAS*, 89:10915–10919, 1992.

Henning Hermjakob, Luisa Montecchi-Palazzi, Chris Lewington, Sugath Mudali, Samuel Kerrien, Sandra Orchard, Martin Vingron, Bernd Roechert, Peter Roepstorff, Alfonso Valencia, Hanah Margalit, John Armstrong, Amos Bairoch, Gianni Cesareni, David Sherman, and Rolf Apweiler. Intact: an open source molecular interaction database. *Nucleic Acids Res.*, 32(Database issue):D452–D455, 2004.

M. Hirosawa, Y. Totoki, M. Hoshida, and M. Ishikawa. Comprehensive study on iterative algorithms of multiple sequence alignment. *Comput Appl Biosci*, 11:13–18, 1995.

Lynette Hirschman, Marc Colosimo, Alexander Morgan, and Alexander Yeh. Overview of biocreative task 1b: normalized gene lists. *BMC Bioinformatics*, 6(Suppl 1):S11, 2005. doi: 10.1186/1471-2105-6-S1-S11.

Yuen Ho, Albrecht Gruhler, Adrian Heilbut, Gary D. Bader, Lynda Moore, Sally-Lin Adams, Anna Millar, Paul Taylor, Keiryn Bennett, Kelly Boutilier, Lingyun Yang, Cheryl Wolting, Ian Donaldson, Søren Schandorff, Juanita Shewnarane, Mai Vo, Joanne Taggart, Marilyn Goudreault, Brenda Muskat, Cris Alfarano, Danielle Dewar, Zhen Lin, Katerina Michalickova, Andrew R. Willems, Holly Sassi, Peter A. Nielsen, Karina J. Rasmussen, Jens R. Andersen, Lene E. Johansen, Lykke H. Hansen, Hans Jespersen, Alexandre Podtelejnikov, Eva Nielsen, Janne Crawford, Vibeke Poulsen, Birgitte D. Sørensen, Jesper Matthiesen, Ronald C. Hendrickson, Frank Gleeson, Tony Pawson, Michael F. Moran, Daniel Durocher, Matthias Mann, Christopher W.V. Hogue, Daniel Figeys, and Mike Tyers. Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, 415:180–183, 2002.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Mass., 2 edition, 2001.

Chun-Nan Hsu, Yu-Ming Chang, Cheng-Ju Kuo, Yu-Shi Lin, Han-Shen Huang, and I-Fang Chuang. Integrating high dimensional bi-directional parsing models for gene mention tagging. In *Proc ISMB*, Toronto, Canada, 2008.

Minlie Huang. Note on learning substitution matrices, 2004. Personal communication.

Minlie Huang, Xiaoyan Zhu, Yu Hao, Donald G. Payan, Kunbin Qu, and Ming Li. Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, 20(18):3604–3612, 2004.

Minlie Huang, Shilin Ding, Hongning Wang, and Xiaoyan Zhu. Mining physical protein-protein interactions by exploiting abundant features. In Lynette Hirschman, Martin Krallinger, and Alfonso Valencia, editors, *Proc 2nd BioCreative Challenge Evaluation Workshop*, pages 237–245, 2007.

R. Hughey and A. Krogh. Hidden markov models for sequence analysis: Extension and analysis of the basic method. *Comput. Applic. Biosci.*, 12:95–107, 1996.

Lawrence Hunter, Zhiyong Lu, James Firby, William A. Baumgartner Jr, Helen L. Johnson, Philip V. Ogren, and K. Bretonnel Cohen. Opendmap: An open source, ontology-driven concept analysis engine, with applications to capturing knowledge regarding protein transport, protein interactions and cell-type-specific gene expression. *BMC Bioinformatics*, 9:78, 2008. doi: 10.1186/1471-2105-9-78.

L. Issel-Tarver, K.R. Christie, K. Dolinski, R. Andrada, R. Balakrishnan, C.A. Ball, G. Binkley, S. Dong, S.S. Dwight, D.G. Fisk, M. Harris, M. Schroeder, A. Sethuraman, K. Tse, S. Weng, D. Botstein, and J.M. Cherry. Saccharomyces genome database. *Methods Enzymol*, 350:329–346, 2002.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

Hyunchul Jang, Jaesoo Lim, Joon-Ho Lim, Soo-Jun Park, Kyu-Chul Lee, and Seon-Hee Park. Finding the evidence for protein-protein interactions from pubmed abstracts. *Bioinformatics*, 22(14):e220–e226, 2006. doi: 10.1093/bioinformatics/btl203.

Edwin T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

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):21–28, May 2001. doi: 10.1038/88213.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. ECML*. Springer, 1998.

Thorsten Joachims. Learning to align sequences: A maximum-margin approach. Technical report, Department of Computer Science Cornell University, August 28 2003.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11—21, 1972.

Susan Jones and Janet M. Thornton. Principles of protein-protein interactions. *Proc Natl Acad Sci*, 93(1):13–20, January 9 1996.

Renata Kabiljo, Andrew B. Clegg, and Adrian J. Shepherd. A realistic assessment of methods for extracting gene/protein interactions from free text. *BMC Bioinformatics*, 2009. To appear.

Kyo Kageura. *The Dynamics of Terminology – A descriptive theory of term formation and terminological growth*. John Benjamins Publishing Company, Amsterdam, 2002.

Thomas Kappeler, Kaarel Kaljurand, and Fabio Rinaldi. Tx task: Automatic detection of focus organisms in biomedical publications. In *Proceedings of the BioNLP 2009 Workshop*, pages 80–88, Boulder, Colorado, June 2009. Association for Computational Linguistics.

Sophia Katrenko and Pieter Adriaans. A local alignment kernel in the context of nlp. In *Proc COLING*, 2008.

Samuel Kerrien. Intact — serving the text–mining community with high quality molecular interaction data. In *2nd BioCreative Challenge Evaluation Workshop*, 2007.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. Genia corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl. 1):i180–i182, 2003.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity task at jnlpba. In *Proc. Joint Workshop on Natural Language Processing in Biomedicine and its Applications, JNLPBA*, pages 70–75, 2004.

Seonho Kim, Juntae Yoon, and Jihoon Yang. Kernel approaches for genic interaction extraction. *Bioinformatics*, 24(1):118–126, 2008a. doi: 10.1093/bioinformatics/btm544.

Sun Kim, Soo-Yong Shin, In-Hee Lee, Soo-Jin Kim, Ram Sriram, and Byoung-Tak Zhang. Pie: an online prediction system for protein-protein interactions from text. *Nucleic Acids Res*, 2008b. doi: 10.1093/nar/gkn281.

Shuhei Kinoshita, K. Bretonnel Cohen, Philip V. Ogren, and Lawrence Hunter. Biocreative task1a: entity identification with a stochastic tagger. *BMC Bioinformatics*, 6 (Suppl 1):S4, 2005.

Harald Kirsch and Dietrich Rebholz-Schuhmann. Method used for biocreative task 1a. In *Proc. BioCreAtIvE Workshop*, Granada, Spain, March 2004.

Harald Kirsch, Sylvain Gaudan, and Dietrich Rebholz-Schuhmann. Distributed modules for text annotation and ie applied to the biomedical domain. *Int. J. Med. Inform.*, 75:496–500, 2005.

Kiroaki Kitano. Systems biology: A brief overview. *Science*, 295:1662–1664, 2002.

J. Kivinen, M.K. Warmuth, and P. Auer. The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. In *Proc. Conf. on Computational Learning Theory*, pages 289–296, Santa Cruz, USA, 1995.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proc ACL*, pages 423–430, July 2003.

Roman Klinger, Christoph M. Friedrich, Juliane Fluck, and Martin Hofmann-Apitius. Named entity recognition with combinations of conditional random fields. In *Proc 2nd BioCreative Challenge Evaluation Workshop*, pages 89–91, 2007.

Martin Krallinger, Florian Leitner, and Alfonso Valencia. Assessment of the second biocreative ppi task: Automatic extraction of protein–protein interactions. In *Proc 2nd BioCreative Challenge Evaluation Workshop*, pages 41–54, Madrid, Spain, 2007.

M. Krauthammer, P. Kra, I. Iossifov, S.M. Gomez, G. Hripcsak anfd V. Hatzivassiloglou, C. Friedman, and A. Rzhetsky. Of truth and pathways: chasing bits of information through myriads of articles. *Bioinformatics*, 18(Suppl 1):S249–S257, 2002.

Michael Krauthammer, Andrey Rzhetsky, Pavel Morozov, and Carol Friedman. Using blast for identifying gene and protein names in journal articles. *Gene*, 259:245–252, 2000.

Cheng-Ju Kuo, Yu-Ming Chang, Han-Sheng Huang, Kuan-Ting Lin, Bo-Hou Yang, Yu-Shi Lin, Chun-Nan Hsu, and I-Fang Chung. Rich feature set, unification of bidirectional parsing and dictionary filtering for high f-score gene mention tagging. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, pages 105–107, Madrid, Spain, April 23–25 2007.

Bob Leaman and Graciela Gonzalez. Banner: An executable survey of advances in biomedical named entity recognition. In *Proc Pac Symp Biocomput*, 2008.

Matthew Lease and Eugene Charniak. Parsing biomedical literature. In R. Dale et al., editor, *IJCNLP 2005*, LNAI 3651, pages 58–69. Springer-Verlag, 2005.

Florian Leitner, Martin Krallinger, Carlos Rodriguez-Penagos, Jörg Hakenberg, Conrad Plake, Cheng-Ju Kuo, Chun-Nan Hsu, Richard Tzong-Han Tasi, Hsi-Chuan Hung, William W. Lau, Calvin A. Johnson, Rune Sætre, Kazuhiro Yoshida, Yan Hua Chen, Sun Kim, Soo-Yong Shin, Byoung-Tak Zhang, William A. Baumgartner, Lawrence Hunter, Barry Haddow, Michael Matthew, Xinglong Wang, Patrick Ruch, Frédéric Ehrler, Arzucan Özgür, Günes Erkan, Dragomir R. Radev, Michael Krauthammer, ThaiBinh Luong, Robert Hoffmann, Chris Sander, and Alfonso Valencia. Introducing meta-services for biomedical information extraction. *Genome Biology*, 9(S2):S6, 2008.

Christina Leslie and Rui Kuang. Fast string kernels using inexact matching for protein sequences. *J. Mach. Learn. Res.*, 5:1435–1455, 2004.

Christina Leslie, Eleazar Eskin, , and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Proc. Pac. Symp. Biocomput., PSB*, volume 7, pages 566–575, 2002.

James Lewis, Stephan Ossowski, Justin Hicks, Mounir Errami, and Harold R. Garner. Text similarity: an alternative way to search medline. *Bioinformatics*, 22(18):2298–2304, 2006. doi: 10.1093/bioinformatics/btl388.

Yanpeng Li, Hongfei Lin, and Zhihao Yang. Incorporating rich background knowledge for gene named entity classification and recognition. *BMC Bioinformatics*, 2009. To appear.

Dieter-Chichung Lie, Sophia A. Colamarino, Hong-Jun Song, Laurent Dèsirè, Helena Mira, Antonella Consiglio, Edward S. Lein, Sebastian Jessberger, Heather Lansford, Alejandro R. Dearie, and Fred H. Gage. Wnt signalling regulates adult hippocampal neurogenesis. *Nature*, 437(7063):1370–1375, 2005.

M.W. Mak, S.Y. Kung, and S.H. Lin. *Expectation-Maximization Theory*, chapter 3. Prentice Hall, 2004.

Edward M. Marcotte, Ioannis Xenarios, and David Eisenberg. Mining literature for protein interactions. *Bioinformatics*, 17(4):359–363, April 2001.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19:313–330, 1993.

John McCrae and Nigel Collier. Synonym set extraction from the biomedical literature by lexical pattern discovery. *BMC Bioinformatics*, 9:159, 2008. doi: 10.1186/1471-2105-9-159.

D.M. McDonald, H. Chen, H. Su, and B.B. Marshall. Extracting gene pathway relations using a hybrid grammar: the arizona relation parser. *Bioinformatics*, 20(18):3370–3378, 2004.

Ryan McDonald and Fernando Pereira. Identifying gene and protein mentions in text using conditional random fields. In *BioCreAtIvE Workshop*, Granada, Spain, March 2004.

Ryan McDonald and Fernando Pereira. Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics*, 6(Suppl 1):S6, 2005. doi: 10.1186/1471-2105-6-S1-S6. 10.1186/1471-2105-6-S1-S6.

Ryan McDonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *European Association for Computational Linguistics*, 2006.

Sven Mika and Burkhard Rost. Protein names precisely peeled off free text. *Bioinformatics*, 20(Suppl 1):i241–i247, 2004.

Andrei Mikheev. Feature lattices for maximum entropy modelling. In *Proc. COLING-ACL*, pages 848–854, 1998.

Boris Mirkin. *Mathematical Classication and Clustering*. Kluwer Academic Publishers, 1996.

Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

Tomohiro Mitsumori, Sevrani Fation, Masaki Murata, Kouichi Doi, and Hirohumi Doi. Gene/protein name recognition using support vector machine after dictionary matching. In *BioCreAtIvE Workshop*, Granada, Spain, March 2004.

Tomohiro Mitsumori, Sevrani Fation, Masaki Murata, Kouichi Doi, and Hirohumi Doi. Gene/protein name recognition based on support vector machine using dictionary as features. *BMC Bioinformatics*, 6(Suppl 1):S8, 2005.

Yusuke Miyao and Jun'ichi Tsujii. Feature forest models for probabilistic hpsg parsing. *Computational Linguistics*, 34:35–80, 2008.

Yusuke Miyao, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun'ichi Tsujii. Evaluating contributions of natural language parsers to protein–protein interaction extraction. *Bioinformatics*, 25(3):394–400, 2008. doi: 10.1093/bioinformatics/btn631.

Yusuke Miyao, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun'ichi Tsujii. Evaluating contributions of natural language parsers to protein–protein interaction extraction. *Bioinformatics*, 25(3):394–400, 2009. doi: 10.1093/bioinformatics/btn631.

Alexander A. Morgan, Zhiyong Lu, Xinglong Wang, Aaron A. Cohen, Juliane Fluck, Patrick Ruch, Anna Divoli, Katrin Fundel, Robert Leaman, Jörg Hakenberg, Chenjie Sun, Heng-Hui Liu, Rafael Torres, Michael Krauthammer, William W. Lau, Hongfang Liu, Chun-Nan Hsu, Martijn Schuemie, and Lynette Hirschman. Overview of biocreative ii gene normalization. *Genome Biology*, 9(S2):S3, 2008. doi: 10.1186/gb-2008-9-s2-s3.

Alessandro Moschitti. Making tree kernels practical for natural language processing. In *Proc EACL*, pages 113–120, Trento, Italy, 2006.

Annette M. Nahin. Boost for pubmed search results: New atm & citation sensor introduced. *NLM Tech Bull*, 362(e10), May-June 2008.

Claire Nédellec. Learning language in logic – genic interaction extraction challenge. In *Proc 4th Learning Language in Logic Workshop (LLL05) at the 22nd Int Conf on Machine Learning*, Bonn, Germany, August 2005.

Goran Nenadic, Simon Rice, Irena Spasic, Sophia Ananiadou, and Benjamin J. Stapley. Selecting text features for gene name classification: from documents to terms. In *Proc. ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 121–128, Sapporo, Japan, 2003.

Chikashi Nobata, Nigel Collier, and Jun'ichi Tsujii. Automatic term identification and classification in biology texts. In *Proceedings of the Natural Language Pacific Rim Symposium*, pages 369–374, 1999.

Mohamed A. F. Noor, Katherine J. Zimmerman, and Katherine C. Teeter. Data sharing: How much doesn't get submitted to genbank? *PLoS Biology*, 4(7):e228, 2006.

Franz Josef Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *Proc. Int. Conf. on Computational Linguistics, COLING*, pages 1086–1090, Saarbrücken, Germany, August 2000.

Tomoko Ohta, Yuka Tateisi, Hideki Mima, and Jun'ichi Tsujii. Genia corpus: an annotated research abstract corpus in molecular biology domain. In *Proc. of the Human Language Technology Conference (HLT)*, pages 73–77, 2002.

OMIM. Online mendelian inheritance in man, omim^{TM}. McKusick-Nathans Institute for Genetic Medicine, Johns Hopkins University (Baltimore, MD) and National Center for Biotechnology Information, National Library of Medicine (Bethesda, MD), 2000.

D. Panov. La traduction mécanique et l'humanité. *Impact*, 10(1):17–25, 1960.

Jong C. Park, Hyon Sook Kim, and Jung Jae Kim. Bidirectional incremental parsing for automatic pathway identification with combinatory categorical grammar. In *Proc Pac Symp Biocomput*, volume 6, pages 396–407, 2001.

Conrad Plake, Jörg Hakenberg, and Ulf Leser. Optimizing syntax patterns for discovering protein-protein interactions. In *Proc ACM Symposium for Applied Computing (ACM SAC), Bioinformatics track*, pages 195–201, Santa Fe, USA, March 2005. doi: 10.1145/1066677.1066722.

Conrad Plake, Torsten Schiemann, Marcus Pankalla, Jörg Hakenberg, and Ulf Leser. ALIBABA: Pubmed as a graph. *Bioinformatics*, 22(19):2444–2445, 2006. doi: 10.1093/bioinformatics/btl408.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, USA, 1998a. MIT Press.

J. Platt. Sequential minimal optimization: A fast algorithm for training of support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998b.

Martin F. Porter. An algorithm for suffix stripping. *Program*, pages 130–137, 1980.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Bjorne, Jorma Boberg, Jouni Jarvinen, and Tapio Salakoski. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50, 2007. doi: 10.1186/1471-2105-8-50.

Sampo Pyysalo, Antti Airola, Juho Heimonen, Jari Bjorne, Filip Ginter, and Tapio Salakoski. Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics*, 9 Suppl 3:S6, 2008. doi: 10.1186/1471-2105-9-S3-S6.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.

Dietrich Rebholz-Schuhmann, Harald Kirsch, and Francisco Couto. Facts from text–is text mining ready to deliver? *PLoS Biology*, 3(2):e65, 2005.

Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the 5th conference on Applied Natural Language Processing*, pages 16–19, Washington DC, USA, 1997.

Stefan Riedel and Ewan Klein. Genic interaction extraction with semantic and syntactic chains. In Claire Nédellec, editor, *Proc Learning Language in Logic Workshop (LLL05) at the 22nd Int Conf on Machine Learning*, pages 69–74, 2005.

C.K. Riesbeck. From conceptual analyzer to direct memory access parsing: An overview. In N.E. Sharkey, editor, *Advances in Cognitive Sciences*, pages 237–258. Ellis Horwood Limited, 1986.

Ellen Riloff and William Phillips. An introduction to the sundance and autoslog systems. Technical Report UUCS-04-015, University of Utah, November 8 2004.

Fabio Rinaldi, Gerold Schneider, Kaarel Kaljurand, Michael Hess, Christos Andronis, Andreas Persidis, and Ourania Konstanti. Relation mining over a corpus of scientific literature. In *Proc. AIME*, pages 550–559. Springer Verlag, LNAI 3581, July 2005.

Fabio Rinaldi, Gerold Schneider, Kaarel Kaljurand, Michael Hess, and Martin Romacker. An environment for relation mining over richly annotated corpora: the case of genia. In *Proc. Int. Symp. on Semantic Mining in Biomedicine, SMBM*, pages 68–75, April 9-12 2006.

Fabio Rinaldi, Thomas Kappeler, Kaarel Kaljurand, Gerold Schneider, Manfred Klenner, Michael Hess, Jean-Marc von Allmen, Martin Romacker, and Therese Vachon. Ontogene in biocreative ii. In *Proc. 2nd BioCreative Challenge Evaluation Workshop*, pages 193–198, Madrid, Spain, April 23-25 2007.

Barbara Rosario and Marti A. Hearst. Classifying semantic relations in bioscience texts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, ACL*, Barcelona, Spain, July 2004.

Rune Sætre, Kazuhiro Yoshida, Akane Yakushiji, Yusuke Miyao, Yunichiro Matsubayashi, and Tomoko Ohta. Akane system: Protein-protein interaction pairs in the biocreative 2 challenge, ppi-ips subtask. In Lynette Hirschman, Martin Krallinger, and Alfonso Valencia, editors, *Proc 2nd BioCreative Challenge Evaluation Workshop*, pages 209–211, Madrid, Spain, April 23-25 2007.

Rune Sætre, Kenji Sagae, and Jun'ichi Tsujii. Syntactic features for protein-protein interaction extraction. In Christopher J.O. Baker and Su Jian, editors, *Proc Int Symp Languages in Biology and Medicine (LBM 2007)*, pages 6.1–6.14, Singapore, 2008.

Kai Salomaa and Sheng Yu. Nfa to dfa transformation for finite languages. In *Automata Implementation*, volume 1260/1997 of *Lecture Notes in Computer Science*, pages 149–158. Springer Berlin/Heidelberg, 1997. doi: 10.1007/3-540-63174-7_12.

Jasmin Saric, Lars Juhl Jensen, Rossitza Ouzounova, Isabel Rojas, and Peer Bork. Extraction of regulatory gene/protein networks from medline. *Bioinformatics*, 22(6): 645–650, 2006.

Torsten Schiemann, Ulf Leser, and Jörg Hakenberg. Word sense disambiguation in biomedical applications: A machine learning approach. In Violaine Prince and Mathieu Roche, editors, *Information Retrieval in Biomedicine: Natural Language Processing for Knowledge Integration*, pages 141–162. IGI Global, 2008.

Andreas Schlicker, Francisco S. Domingues, Jörg Rahnenführer, and Thomas Lengauer. A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinformatics*, 7:302, 2006.

Andreas Schlicker, Carola Huthmacher, Fidel Ramirez, Thomas Lengauer, and Mario Albrecht. Functional evaluation of domain–domain interactions and human protein interaction networks. *Bioinformatics*, 23(7):859–865, 2007. doi: 10.1093/bioinformatics/btm012.

Sebastian Schmeier. Automated recognition and extraction of entities related to enzyme kinetics from text. Master's thesis, Freie Universität Berlin, October 31 2005.

Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proc. Int. Conf. New Methods in Language Processing*, pages 44–49, Manchester, UK, 1994.

Helmut Schmid. Unsupervised learning of period disambiguation for tokenisation. Internal report, IMS, University of Stuttgart, May 2000.

Bernhard Schölkopf and Alex J. Smola. Support vector machines and kernel algorithms. In *Encyclopedia of Biostatistics*. John Wiley and Sons, 2003.

Ariel S. Schwartz and Marti A. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proc. Pac. Symp. Biocomput., PSB*, pages 451–462, 2003.

Kazuhiro Seki and Javed Mostafa. A probabilistic model for identifying protein names and their name boundaries. In *Proc. Computational Systems Bioinformatics Conference (CSB)*, 2003.

Burr Settles. Abner: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005. doi: 10.1093/bioinformatics/bti475.

Benjamin A. Shoemaker and Anna R. Panchenko. Deciphering protein-protein interactions. part i. experimental techniques and databases. *PLoS Computational Biology*, 3 (3):e42, March 2007. doi: 10.1371/journal.pcbi.0030042.

Daniel Sleator and Davy Temperley. Parsing english with a link grammar. In *Third International Workshop on Parsing Technologies*, Tilburg, NL, and Durbuy, B, 10-13 August 1993.

L Smith, T Rindflesch, and W J Wilbur. Medpost: a part-of-speech tagger for biomedical text. *Bioinformatics*, 20(14):2320–2321, Sep 2004. doi: 10.1093/bioinformatics/bth227.

Larry Smith, Lorraine K. Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M. Friedrich, Kuzman Ganchev, Manabu Torii, Hongfang Liu, Barry Haddow, Craig A. Struble, Richard J. Povinelli, Andreas Vlachos, William A. Baumgartner, Lawrence Hunter, Bob Carpenter, Richard Tzong-Han Tsai, Hong-Jie Dai, Feng Liu, Yifei Chen, Chengjie Sun, Sophia Katrenko, Pieter Adriaans, Christian Blaschke, Rafael Torres, Mariana Neves, Preslav Nakov, Anna Divoli, Manuel Mana-Lopez, Jacinto Mata-Vazquez, and W. John Wilbur. Overview of biocreative ii gene mention recognition. *Genome Biology*, 8(Suppl. 2):S2, 2008.

T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147(1):195–197, 1981.

P.H.A. Sneath and R.R. Sokal. *Numerical Taxonomy*. Freeman, San Francisco, 1973.

Yu Song, Eunju Kim, Gary Geunbae Lee, and Byoung kee Yi. Posbiotm-ner in the shared task of bionlp/nlpba 2004. In *Proc. Joint Workshop on Natural Language Processing in Biomedicine and its Applications, JNLPBA*, 2004a.

Yu Song, Eunji Yi, Eunju Kim, and Gary Geunbae Lee. Posbiotm-ner: A machine learning approach. In *Proc. BioCreAtIvE Workshop*, Granada, Spain, March 2004b.

Benjamin J. Stapley and Gerry Benoit. Biobibliometrics: information retrieval and visualization from co-occurrences of gene names in Medline abstracts. In *Proc. Pac. Symp. Biocomput., PSB*, pages 526–537, Hawaii, USA, 2000.

Mathew J. Stephens, Mathew J. Palakal, Snehasis Mukhopadhyay, Rajeev Raje, and Javed Mostafa. Detecting gene relations from medline abstracts. In *Proc Pac Symp Biocomput*, volume 6, pages 483–496, 2001.

Chengjie Sun, Lei Lin, Xiaolong Wang, and Yi Guan. Using maximum entropy model to extract protein-protein interaction information from biomedical literature. In *Advanced Intelligent Computing Theories and Applications*, LNCS, pages 730–737. Springer, 2007a. doi: 10.1007/978-3-540-74171-8_72.

Jingchun Sun, Yan Sun, Guohui Ding, Qi Liu, Chuan Wang, Youyu He, Tieliu Shi, Yixue Li, and Zhongming Zhao. Inpreppi: an integrated evaluation method based on genomic context for predicting protein-protein interactions in prokaryotic genomes. *BMC Bioinformatics*, 8:414, 2007b. doi: 10.1186/1471-2105-8-414.

Charles Sutton and Andrew K. McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 4, pages 93–128. MIT Press, 2006.

D. R. Swanson. Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in Biology and Medicine*, 30(1):7–18, 1986.

D. R. Swanson. Migraine and magnesium: eleven neglected connections. *Perspectives in Biology and Medicine*, 31(4):526–557, 1988.

Koichi Takeuchi and Nigel Collier. Bio-medical entity extraction using support vector machines. In *Proc. ACL Workshop on Natural Language Processing in Biomedicine*, 2003.

Javier Tamames and Alfonso Valencia. The success (or not) of hugo nomenclature. *Genome Biol*, 7:402, 2006.

Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Suppl 1):S136–S144, 2002.

C. Tarnas and R. Hughey. Reduced space hidden markov model training. *Bioinformatics*, 14, 1998.

Joshua M. Temkin and Mark R. Gilder. Extraction of protein interaction information from unstructured text using a context-free grammar. *Bioinformatics*, 19(16):2046–2053, 2003.

W. M. Thorburn. The myth of ockham's razor. *Mind*, XXVII(3):345–353, 1918. doi: 10.1093/mind/XXVII.3.345.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. Sentence and token splitting based on conditional random fields. In *Pac Assoc for Comp Ling*, pages 49–57, 2007.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

C.J. van Rijsbergen. *Information Retrieval*. Butterworth, London, 1979.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

S.V.N. Vishwanathan and A. J. Smola. Fast kernels for string and tree matching. In K. Tsuda, B. Schölkopf, and J.P. Vert, editors, *Kernels and Bioinformatics*, Cambridge, MA, USA, 2003. MIT Press.

Andrew J. Viterbi. Error bounds for convolutional codes and an asymtotically optimum decoding algorithm. *IEEE Trans. on Information Theory*, 13:260–267, 1967.

H.M. Wain, M. Lush, F. Ducluzeau, and S. Povey. Genew: The human nomenclature database. *Nucleic Acids Res.*, 30:169–171, 2002.

Cathy H. Wu, Rolf Apweiler, Amos Bairoch, Darren A. Natale, Winona C. Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, Maria J. Martin, Raja Mazumder, Claire O'Donovan, Nicole Redaschi, and Baris Suzek. The universal protein resource (uniprot): an expanding universe of protein information. *Nucleic Acids Res.*, 34(Database issue):D187–D191, 2006.

Ioannis Xenarios, Esteban Fernandez, Lukasz Salwínski, Xiaoqun Joyce Duan, Michael J. Thompson, Edward M. Marcotte, and David Eisenberg. Dip: the database of interacting proteins: 2001 update. *Nucleic Acids Res.*, 29(1):239–241, 2001.

Juan Xiao, Jian Su, GuoDong Zhou, and ChewLim Tan. Protein-protein interaction extraction: A supervised learning approach. In *Proc. Symp. Semantic Mining in Biomedicine (SMBM)*, pages 51–59, Hinxton, UK, April 10-13 2005.

Akane Yakushiji. *Relation Information Extraction Using Deep Syntactic Analysis*. PhD thesis, University of Tokyo, 2006.

Akane Yakushiji, Yuka Tateisi, Yusuke Miyao, and Jun'ichi Tsujii. Event extraction from biomedical papers using a full parser. In *Proc. Pac. Symp. Biocomput., PSB*, pages 408–419, Hawaii, USA, 2001.

Itai Yanai, Jan O. Korbel, Stephanie Boue, Shannon K. McWeeney, Peer Bork, and Martin J. Lercher. Similar gene expression profiles do not imply similar tissue functions. *Trends in Genetics*, 22(3):132–138, 2006. doi: 10.1016/j.tig.2006.01.006.

Alexander Yeh, Alexander Morgan, Marc Colosimo, and Lynette Hirschman. Biocreative task 1a: gene mention finding evaluation. *BMC Bioinformatics*, 6(Suppl 1):S2, 2005. doi: 10.1186/1471-2105-6-S1-S2.

Shi Yu, Steven Van Vooren, Leon-Charles Tranchevent, Bart De Moor, and Yves Moreau. Comparison of vocabularies, representations and ranking algorithms for gene prioritization by text mining. In *Proc ECCB*, Cagliari, Italy, 2008.

GuoDong Zhou and Jian Su. Exploring deep knowledge resources in biomedical name recognition. In *Proc. Joint Workshop on Natural Language Processing in Biomedicine and its Applications, JNLPBA*, pages 96–99, 2004.

GuoDong Zhou, Dan Shen, Jie Zhang, Jian Su, SoonHeng Tan, and ChewLim Tan. Recognition of protein/gene names from text using an ensemble of classifiers and effective abbreviation detection. In *Proc. BioCreAtIvE Workshop*, Granada, Spain, March 2004a.

GuoDong Zhou, Jie Zhang, Jian Su, Dan Shen, and ChewLim Tan. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178–1190, 2004b. doi: 10.1093/bioinformatics/bth060.

List of Figures

List of Tables

Selbstständigkeitserklärung

Hiermit erkläre ich, dass

(i) ich die vorliegende Dissertationsschrift selbstständig und ohne unerlaubte Hilfe verfasst habe;

(ii) ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze; und

(iii) mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin gemšßdes amtlichen Mitteilungsblattes Nr. 34/2006 bekannt ist.

Jörg Hakenberg

Berlin, 9. März 2009