Text Mining for Drug Discovery

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER FOR THE DEGREE OF MPHIL IN THE FACULTY OF COMPUTER SCIENCE

MANCHESTER, UK, 2014

DIMITRIOS PILIOURAS

piliourd@cs.man.ac.uk

@8104024

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Contents

Total number of words $\rightarrow$ 30,823

Abstract

TEXT MINING FOR DRUG DISCOVERY
Dimitrios Piliouras
A thesis submitted to the University of Manchester
for the degree of Master of Philosophy (MPhil), 2014

Over the last several decades, medical and biological research has risen to extraordinary levels, opening vast windows into the mechanisms underlying health and disease in living organisms. Integrating this knowledge into a unified framework to enhance our understanding and decision-making is a significant challenge for the research community. Efficient drug discovery and development requires methods for bridging pre-clinical data with patient data, as well as effective literature-mining, in order to estimate both efficacy and safety outcomes for new molecules and treatment approaches. Text mining is often regarded as an antidote to this exponential growth of biomedical publications.

In this thesis text-mining and natural-language-processing techniques and tools, aiming to assist with various computational aspects of drug-discovery, are presented. In particular, methods useful for modelling of pharmaco-kinetic parameters, a process by which, the pharmaceutical effects of a drug can be simulated using mathematical models, are pursued. In order to fully realise the potential of such modelling, there is a tremendous need for databases of verified pharmaco-kinetic/dynamic properties of drugs. To that end, a context-free-grammar, capable of capturing such parameters, and their potential modifications, is proposed. The fully deterministic nature of a context free grammar can be side-stepped by embedding a lexical analyser which is able to plug-in external components for specialised sub-tasks (i.e., named entity recognition). The feasibility of this approach is evaluated against a gold-standard corpus, where it is shown to be both effective and efficient, with predictive accuracy touching 90%.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Sophia Ananiadou for her continuous support in my studies and research, her patience, motivation, enthusiasm, and of course, immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

Besides my main advisor, I would like to thank my two co-supervisors: Dr. Andrew Dowsey and Dr. Yannis Korkontzelos, for their support in experiments, encouragement, insightful comments, and hard questions. Furthermore, I would like to thank the entirety of NaCTem and more specifically, Paul Thompson, Claudiu Mihaila, Riza Batista Navarro and George Kontonatsios, for contributing to a pleasant working environment in MIB (Manchester Institute of Biotechnology). It has been both an honour and an absolute joy to work with them.

Last but certainly not least, I would like to express my eternal love and deepest gratitude to the people closest to me. That is, my parents, whose financial and moral support during my undergraduate studies, made everything else possible, and my beloved girlfriend Danielle, for putting some colour on my life's canvas. I cannot imagine how would I ever finish my postgraduate studies, if it were not for her constant love and faith in me through all those hard times together, taking care of all my mental and physical needs.

Chapter 1

Introduction

1.1 Historical perspective

The field of text-mining (TM) is a relatively new discipline sprung out of the knowledge discovery in databases (KDD) and data mining (DM) communities. As it tends to happen when a discipline is born, it borrows techniques and approaches from similar but more grounded fields, before establishing its own identity (similarly to the first cars being shaped like horse carts or the first films looking like theatre plays).

Acording to Alessandro Zanasi [217], the first time he encountered the term "text mining" (TM) was when it was spoken by Charles Huot in 1994, during the IBM-ECAM (European Centre for Applied Mathematics) workshop in Paris, albeit whether it was used in the same sense as today (that is, in the context of applications such as information extraction or document classification), is unclear. A couple of years later, Ronen Feldman and colleagues offered the first contributions to the field that can be called TM with more certainty [62, 63, 61], previously refereed to as 'knowledge discovery in text' (KDT). The word 'mining' was introduced not long after, in the context of KDT, followed by the coinage of the name "text-mining", an obvious variation of the name "data-mining". This term quickly became the accepted name for the new discipline which opened new avenues of research and notable sub-fields, such as web text-mining (1998) and biomedical TM (1998). TM attracted researchers from various fields, including the KDD and DM communities, several fields of natural language processing (NLP), automatic knowledge acquisition, information retrieval (IR), and information extraction (IE), to name but a few.

Marti A. Hearst was one of the first to summarise the state of this embryonic discipline in 1999 [78]. In an effort to position its scope with respect to other disciplines such as data mining and computational linguistics, Hearst stressed that the defining quality of TM lies in its intrinsic aim to discover novel information, contrary to fields such as information retrieval and data mining. Realising this view, makes it easy to understand why TM is eternally indebted to *literature-based discovery*, a field sparked by Don Swanson with his seminal paper in 1986, "Undiscovered public knowledge" [184].

Literature based discovery was intended to be a systematic search for disparate pieces of knowledge that could be combined together to form a novel discovery. Originally, this was primarily a non-automated process. In fact, Swanson recalled that his insight was largely driven by a purely coincidental finding of two unrelated articles that could be combined to answer a question that neither of these two articles could answer individually.

1.2 Biomedical Text-Mining

Biomedicine is one of the disciplines where scientific publications are the main vehicle to disseminate information in. The very first attempts at mining the biomedical literature date back to 1998. As explained earlier, the label "text mining" may have consumed some areas that formerly went by a different name, such as knowledge acquisition and IE. Since TM builds on previous informatics and computational work on semantic analysis, dictionary creation, knowledge acquisition, classification, etc, its application to the biomedical domain is a natural extension. Particularly given the existing opportunities: explosion of literature (both in size and electronic availability); the gradual shift to electronic medical records; the ongoing work on annotated resources; and the increasing need for integration between disparate information sources. The primary engine that has fuelled this growth, is of course the internet. Even though computers and electronic communications long pre-date the internet, it is the internet that has solidified change, simply because it has dramatically lowered the cost of information access and exchange, but also brought to the social forefront the challenges and opportunities of biomedical electronic information (e.g., the Health Insurance Portability and Accountability Act of 1996; the Open Access movement).

Biomedical TM holds the promise of, and in some cases delivers a reduction in cost and an acceleration of discovery, providing timely access to required facts and explicit or implicit associations between them. Due to the specific goals of biomedical TM, biologists and clinicians are better positioned to define useful TM tasks. Cohen and Hunter [53] note that the most fruitful approaches to biomedical TM will combine the efforts and leverage the abilities of both biologists and computational linguists. Biologists and clinicians will take advantage of their ability to focus on specific tasks and experience in using the unparalleled publicly available domain-specific knowledge sources, whereas TM specialists will focus on providing system components, design and evaluation methods.

The state of biomedical TM is reviewed relatively regularly. Certain recent surveys [220, 219], special journal issues [91, 47], and books [25] in this area indicate that general-purpose text and data mining tools are not well-suited for the biomedical domain, due to its highly specialised nature, which therefore demands for highly specialised approaches of mining.

The sheer size of the entirety of the texts relevant to biology and medicine dictates a step-wise approach to biomedical TM. Typically, the goal of the first step is to reduce the set of text documents to be mined. This reduction is most commonly achieved using domain-specific information retrieval (IR) approaches, as described in *Information Retrieval: A Health and Biomedical Perspective* [80]. Alternatively, documents can be selected using clustering and classification techniques [110, 175, 38].

The biomedical community has been making extensive use of TM technologies in recent decades. Enormous progress has been made in developing tools and methods, and the TM community has witnessed some exciting developments. In addition, the rapid increase in the publication-rate in general (as reflected in the growth of the contents of PubMed/MEDLINE), the advent of high-throughput assays, which commonly produce lists of genes much larger than were seen in previous experimental methods, and the surprising surge of interest by clinicians in using electronic medical record (EMR) systems to improve the quality of care through decision support and evidence-based medicine, have energised the work in biomedical TM (BioNLP) of both clinical and literature text, to extraordinary levels. Another contributor to the growth of BioNLP has been the availability of a wide range of resources suitable for analysis or tools for assisting in such analyses. These include no-cost textual sources such as PubMed/MEDLINE and PubMedCentral; a large variety of corpora[1]; ontologies and other lexical semantic resources, such as the Gene Ontology, UMLS and the Semantic Network; databases such as Entrez Gene, DrugBank and DIP; and ‘off-the-shelf’ NLP pipelines and workflows for IE (UIMA, CTakes etc).

[1]`http://compbio.ucdenver.edu/ccp/corpora/obtaining.shtml`

1.3 Text mining for pharmacology

1.3.1 Motivation

In recent decades, the Food and Drug Administration (FDA) has increased its requirements for drug testing. Today, a new drug requires an average of 15 years and close to a billion dollars spent in research and development, before it reaches consumers [60]. Unfortunately, and in spite of millions of dollars spent on preclinical testing, much of that expense is lost on drugs that never make it to the market. It is estimated that only one in 10 drugs that enter clinical testing receives eventual FDA approval [102]. Alarmingly, for drugs in phase III that have shown evidence of effectiveness in phase II, the failure rate has increased to 50% [124]. The billions of dollars invested in basic biomedical research and clinical development of new medical products are yielding fewer and fewer innovative products. A long, expensive development process has become a major impediment and is a disincentive for new product development, according to the FDA's "Critical Path Initiative" report [74]. The report concluded that the major contributor to the inefficiency in development was the absence of innovative new methods for preclinical and clinical testing of drugs, quoting that "developers are often forced to use the tools and techniques of the last century to evaluate this centurys advances". In multiple follow up publications, clinical researchers, experimental/-computational biologists and biostatisticians from both academia and industry started to debate the challenges and opportunities of the pharmacokinetic-pharmacodynamic (PK/PD) model based approach in drug development [46, 49, 113].

1.3.2 Pharmaco-(kinetic/dynamic) modelling

Pharmacokinetics/pharmacodynamics describe the mathematics of absorption, distribution, metabolism and excretion of drugs in the body. PK/PD modelling simulates the pharmaceutical effects of a drug using mathematical equations by integrating both of its pharmaco-kinetical and pharmaco-dynamical properties. Nowadays, drug discovery is considered impossible without sophisticated modelling and computation, which can substantially reduce the cost of drug development by constructing effective simulations, identifying therapeutic strategies and making novel predictions. To that end, this thesis focuses on text mining techniques to assist PK modelling. In order to fulfil the PK modelling potential in drug development, there is an enormous need for databases of PK parameters. For example, to specify the first human dose of a new compound, one needs its corresponding in-vitro and in-vivo PK parameters based on animal studies. However, current pharmacology databases provide little PK data. DiDB is an ongoing project which manually accumulates published PK data for each drug. DrugBank [211] is a comprehensive pharmacology database which has rich annotations on the structure, mechanism, pathway and targets of drugs, but offers very sparse PK data. Though currently most PK databases still rely on manual curation to collect accurate data, this is inefficient to both, keep up with the exponentially growing scientific publications, and also to handle the large amount of varying information needs of users. In fact, one study argued that it will take years, perhaps even decades, for biomedical database construction, if we just rely on manual curation [31]. Meanwhile, computer-aided curation has been proven to be effective in maintaining the MEDLINE database. Therefore we investigate text mining as an alternative solution to manual curation for PK parameter data collection which targets to efficiently handle large scale of information and automatically extract good quality PK data.

Chapter 2

Related work

2.1 Resources for Biomedical Text Mining

As the name implies, biomedical TM is primarily concerned with biomedical textual data. Given the inherent importance of such data, it is worth introducing certain well established and widely used biomedical text collections, and this section attempts to do so. For most tasks, any degree of structure is better than no structure, and therefore it is easy to imagine why annotated corpora, are often more useful than the original raw text alone. As publicly available annotated texts continue to grow, the need for compatible annotation formats, guidelines and standards, which this section also touches upon, has been greater than ever [179]. The technical infrastructure that makes biomedical TM possible, including widely-used tools and frameworks as well as important lexical and knowledge-based repositories, are described towards the end of the section.

2.1.1 Corpora

The relationship of biomedical TM with one specific resource, namely the MEDLINE database, is particularly intimate and is hard to imagine how it could be any different, as it contains more than 20 million bibliographic references to journal articles in the life sciences (with a clear focus on biomedicine), from as far back as 1946 [179]. Perhaps more importantly, it exposes several ways of obtaining records such as abstracts and citations. For TM purposes, MEDLINE records can be downloaded using the Entrez Programming Utilities [140]. Alternatively, certain subsets of MEDLINE citations have been used in community wide evaluations and will exist in their archives [179]. Examples of such collections would be, the historic OHSUMED [169] set containing all MEDLINE citations in 270 medical journals published between 1987 and 1991, but also a more recent set of TREC Genomics Track data [170] containing ten years of citations (1994-2003). While these collections span over a pre-specified amount of time, there do exist other, more task-oriented, collections. For instance, the GENIA corpus [15] contains just under 2,000 MEDLINE abstracts which were retrieved using 3 simple MeSH terms: “human”, “blood cells” and “transcription factors”. Even though, 2,000 may not sound much at first, the GENIA corpus can be considered the most thoroughly annotated collection of MEDLINE abstracts, as it includes annotations for part-of-speech, syntax, coreference, biomedical concepts and events, cellular localisation, disease-gene associations and pathways [179]. Moreover, together with five full-text articles and a collection of radiology reports, form the basis for the BioScope corpus [206]. The earlier BioCreAtIve collections [84, 109, 28] and the PennBioIE corpus [119, 118], which contains 1100 abstracts for cytochrome P-450 enzymes and 1157 oncology abstracts, are also examples of topically-annotated subsets of MEDLINE abstracts. More recently, NaCTeM released the “Metabolite and Enzyme Corpus” which aims to serve as the gold-standard for metabolite and enzyme named entity recognition, in the context of metabolic pathways. Finally, the Collaborative Annotation of a Large Biomedical Corpus (CALBC) initiative has proposed the creation of a broadly scoped and diversely annotated corpus (about 1,000,000 Medline immunology-related abstracts) by automatically integrating the annotations from different named entity recognition systems. This “silver” corpus has recently become available to the public [160].

No matter how informative and undoubtedly useful for TM MEDLINE abstracts may be, they typically do not contain all the details present in full-text articles [179]. Some information (e.g., the exact experimental settings or the discussion of the results) is almost exclusively contained in the main body of an article. The potential of a qualitative increase in the amount of useful information available for mining, gave rise to several full-text collections [179]. For example, the aforementioned TREC Genomics Track dataset contains about 160,000 full-text articles from about 49 genomics-related journals, directly obtained from the Highwire Press [5] electronic distribution of the journals. Similarly, patients’ clinical case descriptions have been the target of full-text annotations in the ImageCLEF evaluations [137, 138]. The Colorado Richly Annotated Full Text Corpus (CRAFT) [3] is another significant contribution to the ever growing body of semantically and syntactically annotated full text collections available to TM researchers. Finally, the largest publicly available source of original (non-annotated), full-text articles is the Open Access subset of PubMed Central (PMC) [12].

Due to the growing interest in clinical TM and bio-surveillance, recent years have witnessed the rise of several clinical-text collections too [179]. Examples of these include the Pittsburgh collection of clinical reports [14], the Multiparameter Intelligent Monitoring in Intensive Care (MIMIC II) database [115], and, of course, the annotated i2b2 collections [202, 203, 204]. Social media and the web (i.e, Facebook, Twitter and various health-related blogs and community web-sites) have played the role of a corpus in several recent studies, however, the availability of such collections, or lack thereof, is not clear [179].

2.1.2 Lexical Repositories

Developing lexicons (dictionaries) is, by no means, a modern approach nor a sophisticated one. However, in cases where it is feasible, there exist major opportunities to be gained from doing so. The linguistic value of lexicons has been known for centuries to people studying natural language. After all, a dictionary of a given language, along with its respective grammar, can act as documentation for that language 'version' (languages evolve over time). There is not a single person on the planet that has never seen or has never benefited from using a dictionary when trying to learn or understand a foreign language. Of course, trying to document an entire human language's vocabulary, which is constantly evolving, falls outside the realms of feasibility. There do exist however, smaller domains, where more restrained (controlled) terminology is used. Biology and medicine related disciplines are good examples of these. In such cases, specialised lexicons are considered a precious resource for NLP.

A fairly general, and famous lexical resource is *WordNet* [127]. WordNet which is available free of charge, was developed at Princeton University and is essentially a database which aims to serve as a resource for NLP and IE applications. Each underlying concept in WordNet, is linked to a set of synonyms (synsets). For example, the concept of 'malignant neoplastic disease' (cancer) is linked with *lymphoma, carcinoma, sarcoma, leukemia or leukaemia etc.*. As of version 3.1, WordNet contains more than $117,650$ synsets, but it has been used only in a few medicine-related projects , due to its rather modest coverage of the domain. Easy integration of WordNet into applications is possible via application programming interfaces (APIs) that have been developed for all major programming languages.

A more medically oriented lexical resource is the *Specialist Lexicon* (SL) from the UMLS project described in the next section. This provides the lexical information needed for processing raw text, not only in the biomedical domain [40]. It is in fact, a general English lexicon, which includes many biomedical terms. For each term, the lexicon maintains an entry which documents syntactic (POS, allowable complementation patterns), morphological (base-form, inflectional variants) and orthographic (spelling variations) information (see 2.1). Contrary to WordNet, the SL does not include any synonyms or semantic relations between terms, however such information is present in a separate UMLS component - the MetaThesaurus. The SL is distributed as part of the UMLS and can also be integrated and queried via APIs for Java and XML. It is also included as an open-source resource in the Specialist NLP Tools [13].

While the UMLS SL, often in conjunction with WordNet, provides a reasonable coverage of the general biomedical language, it does not cover in detail specialised sub-domains like chemical entities, proteins and genes. For these sub-domains, separate, even more specialised resources exist, which can be combined with the general aforementioned ones for optimal results. Examples of publicly available specialised lexical resources for genes, proteins, chemicals and drugs (approved and experimental) can be found in figure 2.2.

Being relatively up to date and freely accessible, DrugBank proved an invaluable resource for this dissertation. It was used not only as a dictionary of drugs, but also as the ground-truth for evolving an NER model using genetic-programming (see Chapter 3).

Figure 2.1: Example SL entry for “hemoglobin” with explanations

{
  base = hemoglobin $\rightarrow$ (base form)
  spelling_variant = haemoglobin $\rightarrow$ (an extra ‘a’)
  entry = E0031208 $\rightarrow$ (unique identifier)
  cat = noun $\rightarrow$ (POS-tag)
  variants = uncount $\rightarrow$ (only singular)
  variants = reg $\rightarrow$ (regular plural: hemoglobins, haemoglobins)
}

Figure 2.2: Publicly available lexical resources for genes, proteins, chemicals and drugs

Domain	Resource	URL
Genes & Proteins	Genew	`http://www.gene.ucl.ac.uk/cgi-bin/nomenclature/searchgenes.pl`
	Entrez Gene	`http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene`
	UniProt	`http://www.uniprot.org/`
Chemicals	PubChem	`http://pubchem.ncbi.nlm.nih.gov/`
	SureChem	`https://www.surechem.com/`
	ChemIDplus	`http://chem.sis.nlm.nih.gov/chemidplus/`
	ChEBI	`http://www.ebi.ac.uk/chebi/`
	CheMBL	`https://www.ebi.ac.uk/chembl/`
Drugs	DrugBank	`http://www.drugbank.ca/`
	DiDB	`http://www.druginteractioninfo.org/`
	RxNorm	`http://www.nlm.nih.gov/research/umls/rxnorm/`
	N.D.C.	`http://www.fda.gov/Drugs/InformationOnDrugs/ucm129662.htm`
Enzymes	BRENDA	`http://www.brenda-enzymes.org/`

2.1.3 Knowledge Repositories

The biomedical domain includes a rather rich set of knowledge sources for supporting TM applications. Arguably, the most comprehensive resource is the Unified Medical Language System (UMLS) [120]. Fundamentally, the UMLS is a compilation of controlled vocabularies, actively maintained by NLM. Apart from bringing together over 100 dictionaries, terminologies and ontologies, it also comes with a semantic network that represents relations between these entries and a dictionary that contains lexicographic information, mainly about biomedical terms but also common English words [179]. Specialised resources for biomedical TM are also maintained by organisations such as the British National Centre for Text-Mining (NaCTeM) [6] and the European Bioinformatics Institute (EBI) [4]. In addition to these broad-coverage resources, there also exist more fine-grained knowledge sources focused on spealised sub-domains of biomedicine. For example, the Pharmacogenomics Knowledge Base [11] is a collection of scientific publications annotated with primary genotype and phenotype data, gene variants and gene-drug-disease relationships, which can be freely acquired and used in the context of academic research [179]. Similarly, the Pharmacokinetics (PK) ontology [213] covers certain PK drug-drug interactions (DDI), whereas, the Neuroscience Information Framework [7], comes with an ontology covering brain anatomy, cells, organisms, diseases, techniques and other aspects of neuroscience. With respect to (bio)chemistry, the SABIO-RK [212] database contains information about biochemical reactions, their kinetic rate equations with parameters and experimental conditions.

Finally, three different research groups at the university of Cambridge, collaborated on project SciBorg [56]. At the heart of the SciBorg project lies a formal language which captures some aspects of the meaning of natural language in a way that allows contributions from different sources to be combined. The combined systems can be used to extract knowledge from text for later machine use, or to give human browsers information about the structure of texts and their interconnections. By leveraging this language, they were able to develop a robust IE architecture and subsequently applied it to chemistry texts. The result was a knowledge management system for chemists capable of performing complex chemical searches such as *"search for papers describing synthesis of Troegers base which dont involve anilines"*, in an expressive query-like language. For instance, the aforementioned (rather difficult) semantic search translates to the simple query shown below:

$$\boxed{\begin{array}{l} X : Goal(X,h), h : synthesis, \\ result(h, < TB >), Source(h,y)\ \&\ NOT(aniline(y)) \end{array}}$$

Ultimately, the best knowledge source for a given TM task will be determined by the nature of the problem at hand. For example, recognising instances of relevant entities is a pre-requisite for literature-based mining (i.e. for relations between genes, diseases and drugs) [179]. Knowing the terms' corresponding semantic types in the UMLS can be a significant aid. Alternatively, individual knowledge sources, such as the Gene Ontology (GO) [30], SNOMED Clinical Terms [181] or the FDA Approved Drug Products with Therapeutic Equivalence Evaluations (Orange Book) [9], can be brought together to form a mega-resource. The resources mentioned in this section are extensively used for performing various biomedical TM tasks, and in certain cases form the basis for deriving 'meta-resources' for a specific task [179]. For instance, Rinaldi et al. [164] define several entity types required for mining the literature for protein interactions (protein/gene names, chemical compounds, cell lines, etc.), which are then used to automatically aggregate terms extracted from curated resources (i.e. UMLS) into an impressive list of more than 2,300,000 terms.

2.1.4 Infrastructure

Tools

The variety and purpose of the tools supporting biomedical TM mirrors the variety and purpose of the knowledge sources described previously [179]. Over the years, high-quality tools, not only for low-level pre-processing tasks such as sentence-segmentation, tokenisation, stemming and POS-tagging, but also for higher level extraction tasks such as named-entity-recognition (NER), relation extraction, sentiment analysis and co-reference resolution, have been developed by various groups. This section focuses on the most commonly used tools for identifying named entities (NEs) and relations and the platforms that allow building TM workflows.

According to [179], *"the single most widely used tool for NER which is based upon the UMLS is MetaMap [29]"*. Simply put, MetaMap is an application that is able to identify UMLS Metathesaurus concepts in free text. It also features the ability to detect author-defined acronyms/abbreviations, to browse the Metathesaurus for concepts even remotely related to input text, to detect the use of negation clues, word sense disambiguation (WSD) and various other technical and algorithmic features. Because of its high configurability and the fact that it relies on the entire UMLS Metathesaurus, it is not easy to determine the best configuration for a given task [179]. However, exploring the options using the interactive MetaMap website aids with such choices. MetaMap, which was provided as a service until recently, is now open source and available for download.

Examples of off-the-shelf, statistical tools used exclusively for biological NER are ABNER [174], BANNER [114], LingPipe [22] and Gimli [44]. Both ABNER and BANNER are based on conditional random fields (CRFs) and rely on a large array of features. Unlike ABNER, BANNER was trained on the BioCreative 2 GM data and avoids semantic features, but it does make use of syntactic ones. Both systems exploit domain-specific language characteristics such as capitalisation, word shapes, prefixes, suffixes and Greek letters. Gimli is a relatively new and open-source biomedical entity recogniser which includes an extended set of implemented and user-selectable orthographic, morphological, linguistic-based, conjunctions and dictionary-based features. It also provides a simple and fast method to combine predictions from separate models.

Relation extraction methods are useful in discovering protein-protein interactions, and gene-binding conditions. Lexical patterns such as "Protein X binds with Protein Y" are often found in biomedical texts where the protein names are entities which are held together by the “bind” relation. Such protein-protein interactions are very useful for applications like drug discovery. Nevertheless, tools for relation extraction are not yet as mature or as readily accessible as NER tools [179]. A relatively recent comparison [89] for available tools for identifying biomedical relations (AkanePPI, Whatizit, and OpenDMAP) to a simple, regular expression-based approach and found that the simple approach performed surprisingly well. The authors conclude that high recall (touching 90%) is achievable for extracting gene-protein relations when the available tools are combined. Tools aimed exclusively at relation extraction include the proprietary AlchemyAPI [1] and the open-source jSRE [198] Java library.

Platforms

Apart from specialised tools like the ones described above, TM researchers also have access to certain more generic NLP toolkits/platforms developed by various academic and open-source communities. These platfroms tend to offer an assortment of utilities, not necessarily focused to a specific task, but towards supporting step-wise NLP. Examples of well-established, open-source NLP platforms written in Java, include the *openNLP* [27] project (Apache Software Foundation), the *stanfordNLP* [200] (Stanford University) and *GATE* [199] (Sheffield University). The *NLTK* [8] software package is the predominant toolkit preferred by users of the Python programming language. All these platforms offer a wide variety of text processing libraries to support building programs that work with human language data, such as sentence-detectors, tokenisers, stemmers, deep/shallow parsers, entity-recognisers, classifiers etc. NLTK in particular, has acquired licences and provides easy-to-use interfaces to over 50 corpora (i.e. Brown corpus [67]) and lexical resources (i.e. WordNet).

A recent trend in tool development and use is the linkage of distinct components from the various aforementioned frameworks to form standalone processing pipelines. Two examples of such platforms offering pipeline/workflow capabilities are the aforementioned “Generalized Architecture for Text Engineering” (GATE) framework and the “Unstructured Information Management Architecture” (UIMA) [65], originally developed by IBM and currently maintained in collaboration with the Apache software foundation. UIMA is, to date, the only industry standard for unstructured content analytics and therefore, has gained significant popularity amongst text-miners the past few years. Many research groups are continuously adapting their tools to function within the UIMA framework, thus enabling direct interaction with tools provided by other groups around the world. Two great examples are the Argo [158] and U-Compare [90] platforms by NaCTeM. Argo is a web-based collaborative environment for the development of text-processing workflows. It comes with many useful features out of which however, its ability to create, not only serial, but also graph-like data flows (multiple branching/merging), set it apart from other similar tools. U-Compare on the other hand, is an integrated NLP system implemented as a standalone Java application. It comes with a friendly graphical user interface which provides drag-and drop facilities for rapidly creating (serial) workflows. Evaluation facilities are also built in. External UIMA components can easily be imported for use in the system, and complete workflows can be serialised for use by other U-Compare users. More recently, U-Compare was extended to automatically convert standalone work-flows into web services [105]. The resulting web services can be registered on a central server and made publicly available.

The biomedical TM resources presented in this section represent a snapshot of the entirety of resources developed over the years. Moreover, due to its very nature, the information and tools presented are likely to become out-dated sooner than the rest of the material in this chapter. In order to keep up with the rather rapid growth of research in the area, there have been attempts by many researchers to actively maintain websites and forums where links to useful resources (e.g., BioNLP [2]) and discussion can be found [179]. The U.S. Department of Veterans Affairs and NLM have gone one step further in order to relieve individual researchers from such a time-consuming task and provide registry of biomedical text mining tools, known as ORBIT [10], which is maintained collectively by the global research community [179].

2.2 Information Retrieval

Information Retrieval (IR), as the name implies, is concerned with appropriate positioning within the vast literature space. To put it differently, what does one have to do, in order to bring the relevant literature closer to reach? Nowadays, the search-engine experience is so pervasive and familiar to all, that it completely dominates most aspects of IR in almost all scientific disciplines and in fact, it is often the case that we forget that IR is only the first step. Realising that we live in an era of information abundance, we can perhaps consider IR (in the classical text-mining context of finding documents that correspond to an information need with the aid of indexes) a solved problem. Therefore, it makes sense to focus mainly on activities subsequent to IR, that are concerned with extracting facts of interest and analysing them to infer others.

2.3 Information Extraction

As one might expect, being able to identify explicitly stated facts in text, has always been a target for all kinds of TM systems. Information extraction (IE) can be defined succinctly, as the process by which structured facts are, in some automatic fashion, captured from unstructured or semi-structured text [179]. Typically, in the biomedical domain that unstructured text is drawn from the published literature, but in some cases, clinical narratives from electronic health records or other systems that manage clinical information, can be very useful as well [179]. Despite the fact that, information from these sources could serve as the input to information retrieval (IR) systems, IE is often performed as an initial processing step for other, more advanced TM applications. Aspects of IE include Named Entity Recognition (NER), Co-reference Resolution (CR), Sentiment Analysis (SA) and Relation Extraction (RE), all of which, are all covered in detail in follow up sections.

Community-wide evaluations focusing specifically on TM within biomedicine, have been a major driver in rapidly transforming IE technologies [179]. Some examples of recent evaluation forums include BioCreAtivE [84, 109], BioNLP [97, 155], i2b2 [202, 203, 204, 205], JNLPBA [98], and LLL [143] shared tasks. The strong interest, as reflected by participation levels in such community-wide evaluation efforts, highlights the ever growing volume of unstructured text available in electronic form these days.

There are three aspects of IE which are particularly relevant when it comes to mining free text (not necessarily biomedical). Naturally, NER which is the task dedicated to the identification and classification of entities into distinct, predefined categories, comes first. The categories are, of course, defined according to the domain in question. In biomedicine, the names of drugs, proteins, genes and diseases, are most commonly sought. Normalisation of the extracted entities to a canonical, unambiguous representation via ontologies and/or dictionaries, and potentially, further classified into semantic categories, is often a desired step [179]. The second sub-task of IE relevant to mining free text is relation extraction, which aims to detect binary relationships amongst the named-entities captured previously. Again, in biomedicine these would typically be gene-disease relationships, protein-protein interactions, drug-drug interactions and clinial problem-treatment relationships. Finally, the third major subtask, event-extraction, seeks to identify highly complex and often recursive relations amongst the extracted entities. Events are characterised by a main predicate (a verb) and typically, there exist arguments and/or participants to that predicate. Events highly relevant to the biomedical domain include, for example, gene expression/regulation, protein-binding and pharmaco-kinetic (PK) drug modification.

Despite aiming to extract different types of information, the underlying methods and techniques used in all the aformentioned sub-tasks are rather similar. These include machine learning (ML), statistical analysis and other well grounded NLP techniques. Challenges and approaches to the sub-tasks of biomedical IE are discussed next.

2.3.1 Named Entity Recognition

According to [179], *"biomedical NER refers to the task of automatically identifying occurrences of, biologically or medically relevant, terms in unstructured scientific text"*. As previously mentioned, gene/protein/drug names and medical conditions/treatments constitute the entity types that are of utmost importance to researchers [117]. NER is, by no means, a single task, despite being often discussed as such. There are at least, 3 steps that make up the core process. Firstly, the exact substring boundaries of the entity in question need to be determined. Secondly, the entity needs to be assigned a semantic type. That is, it must be associated with one of the semantic classes available. Thirdly, the canonical/preferred representation of the concept the entity names needs to be selected. This could vary a simple stemmed token or synonym, to a unique identifier in some remote database where more information can be found. This last sub-task is called entity-normalisation and seeks to compensate for potential lexical and orthographic variations. Given the fact that this is one the most challenging tasks, it will be discussed further in the context of describing the many challenges involved in biomedical NER.

The reasons that make NER particularly challenging for biomedical text are manyfold. On one hand, the extremely dynamic nature of of scientific discovery these days makes it a moving target. Particularly in biomedicine, the landscape of semantically relevant entities is rapidly growing and shifting as new scientific discoveries and contributions are published [216]. On the other hand, there is a very large amount of synonyms. In bio-medicine, it is not uncommon for the same concept to be expressed using distant lexical clues. Reusing the example from Simpson et al. [179], the words "heart attack" and "myocardial infarction" refer to the same medical condition so an NER system should be able to recognise these terms as instances of the same concept, despite having been expressed quite differently. The more synonyms, for a particular concept, are in use, the harder it becomes to combine knowledge from multiple sources without access to some comprehensive synonymy resource such as the UMLS

Metathesaurus or the Gene Ontology. Even with access to such resources, some synonymy relationships may be overlooked as these resources are highly unlikely to be 100% complete at any given point in time [179]. Finally, the multitude of acronyms and abbreviated terms in the biomedical literature makes it difficult to automatically map these terms to the concepts they refer to [179]. More often than not, successful acronym and abbreviation resolution greatly depends on the context in which the terms appear since, as explained previously, the same term can refer to more than one concepts. Reusing another example from Simpson et.al [179], the abbreviation *RA* can refer to “right atrium”, “rheumatoid arthritis”, “refractory anemia”, “renal artery” or one of several other concepts [149]. To address the challenges associated with acronyms, abbreviations and synonymy, most NER systems will perform, to some extent, entity normalisation [179].

As hinted earlier and as described by Simpson et.al [179]: *“Entity normalisation refers to the process of linking entity occurrences to a somewhat, more canonical name. Although a challenging task in its own right, entity normalisation can help resolve issues resulting from synonymous terms and ambiguous acronyms/abbreviations, by associating these entities with unique and unambiguous representations. Sometimes, there may not be community-wide agreement on the preferred name for a given entity. In such cases, the goal of entity normalisation is to map the entity to a unique identifier of a concept in some terminology resource. In general, entity normalisation relies on the existence of such terminology resources, though these may be incomplete. Since normalisation is such an important component of many NER systems, it is often an implied processing step after identifying entity boundaries and assigning them to a category. Nonetheless, the entity normalisation sub-task can be evaluated independently, as indicated recent BioCreAtivE shared task evaluations [82, 135]”*

For NER systems that analyse large amounts of biomedical text, it is important to consider the performance that can be expected of the methods being deployed [179]. Typically, the performance of NER systems is measured in terms of the standard IR metrics - *precision, recall* and *F-Score*. However, certain issues make these measurements difficult to reliably obtain and compare [179].

One obvious issue is the lack of large, high-quality annotated corpora to serve as the gold-standard on which to evaluate the performance of NER systems. Ideally, the gold-standard training data must be large enough to allow the projection of experimental results to potentially, even larger text collections [179]. In addition, high inter-annotator agreement and expert-level judgement are highly desirable properties.

However, it should be noted that, the larger the dataset the less important, annotation errors and/or disagreements, become. That is to say, that a sufficiently large corpus will most likely compensate for certain annotation errors but not vice versa. Interestingly enough, Uzuner et al. [205] demonstrated that errors in the annotated gold-standard for a recent i2b2 shared task evaluation could not have affected the relative performance of competing NER systems, by more than .05%.

Another issue to consider when evaluating NER systems is how the boundaries of an identified entity are defined. Following a strict evaluation scheme translates to exactly matching both the left and right boundaries of an extracted entity with those of the annotations, whereas a loose evaluation scheme only requires that the extracted entity boundaries overlap those of the annotation [117]. It has been shown [148], perhaps not surprisingly, that choosing between a strict or loose evaluation scheme can have an impact on the relative performance of NER systems.

Recent community-wide evaluations are a testament to how good NER systems generally are. For instance, looking back at the first [216] and second [180] BioCreAtIve gene mention recognition tasks, we can see the best performing systems achieving F-scores of 83% and 87%. Similarly, the best candidates in the i2b2 concept extraction task [205] and the JNLPBA bio-entity recognition task [98], scored 85% and 73% respectively.

For instance, the best performing systems achieved F-scores of 83% and 87% for the first [216] and second [180] BioCreAtIve gene mention recognition tasks, 85% for the i2b2 concept extraction task [205], and 73% for the JNLPBA bio-entity recognition task [98]. Although NER systems may be tuned for a particular IE task, their underlying methods and approaches can be broadly grouped in 3 distinct categories - *dictionary-based, rule-based* and *statistical-based* [179].

Dictionary-based methods, being one of the most basic and intuitive approaches, rely on comprehensive lists of terms in order to identify entities of a particular semantic type [179]. From an architectural and technical standpoint, this approach is very straight forward. Systems relying on lexical resources identity entities by simply attempting to locate the candidate token within those resources. If it can be found, the classifier responds positively - otherwise negatively. When deployed as stand-alone methods, dictionary-based approaches tend to exhibit reasonably high precision [179], but they often suffer from poor recall, mainly due to phenomena which cause morphological variance (i.e. spelling mistakes or variants) [197], but also due to the very nature of exact string matching. However, low precision can also stem from homonymy

[83]. For example, there exist gene names and abbreviations thereof (i.e. “an”, “by”, and “can”) that are spelt exactly the same as certain common English words [111]. To that end, some form of ’soft’ string matching is commonly deployed to improve the precision and recall of dictionary-based systems. Certain studies are so concerned with spelling variants, that go as far as firstly generating spelling variants for the all the terms in a some resource, and augmenting the underlying dictionaries with these additional terms [194, 195]. The augmented resource can then be used for regular, exact string matching but with greater confidence. Alternatively, algorithms such as BLAST [23, 24] can be used in order to perform approximate string matching [112]. In spite of these improvements, dictionary-based methods are most often used in conjunction with more advanced NER approaches, especially in the face of data-sparsity, where having access to a “safety-net” NER model can deliver a significant performance boost [151].

Another approach to NER is to describe the compositional patterns of biomedical NEs and their surrounding context using lexical rules [179]. Examples of systems which make use of rule-based approaches include the EMPathIE [86] and PASTA [72], which are able to identify not only protein structures, but also, enzyme interactions by using context-free grammars (CFGs). Orthographic and lexical properties have also been the target of carefully crafted rules geared towards recognising protein [70] and chemical [139] names. These simpler methods can be improved by further consideration of contextual information [85] and the outcome of syntactic analysis for determining entity boundaries [68]. However, despite achieving better performance than dictionary-based approaches in most cases, the hand-crafted generation of the required rules is an expensive and time-consuming process. Moreover, since the rules are usually very specific (in order to achieve high precision), they are almost impossible to extend over other entity classes [179].

These days, possibly due to the seemingly unlimited access to computing power, more and more NER approaches tend to rely on statistical methods instead of, or at least combined with, dictionaries and/or rules [179]. Unlike the previously described approaches, statistical methods rely on some form of predictive modelling in order to identify entities. This is typically achieved by utilising ML algorithms. While supervised ML approaches require observations, typically captured in large bodies of annotated corpora, in order to be trained, recent work has investigated the potential for automatic generation of NER training data through the use of bootstrapping and other semi-supervised techniques [207, 136, 201]. Common statistical methods used for NER can be grouped as either 1:1 classification or sequence-labelling approaches.

The difference between these groups is subtle and particularly in NER, it translates to extracting features only from the token currently examined (1:1 classification), rather than consulting previous tokens as well (sequence-labelling). This ultimately means that 1:1 classification will overlook contextual information. For example, a 1:1 NER classifier would typically include only lexical and morphological features, whereas a sequence-labelling classifier would additionally include features targeted at the *N* previous tokens.

Classification-based approaches transform the NER task into a classification problem, which is applicable to individual words or groups thereof [179] (depending on the classification scheme adopted). Common classifiers deployed for biomedical NER include Support Vector Machine (SVM) [94, 128, 185, 214] and Naive Bayes [142] models. While it is not impossible to classify multi-word names, the BIO tagging scheme [159], which dictates that individual tokens are classified as being either at the beginning (B) of an entity, inside (I) the boundaries of an entity, or outside (O) the boundaries of an entity, has proven rather popular amongst researchers [179]. However, in the presence of overlapping entity boundaries, this tagging scheme is not expressive enough to describe the notion of a hierarchy, and therefore, several researchers have looked into the issue of recognising nested NEs [73, 21].

The overall performance of classification-based approaches is, in no small part, dependent on the choice of features used during training, and many authors have explored various feature combinations/permutations. For instance, two studies [94] [128] consider morpho-syntactic properties of NEs whereas Takeuchi and Collier [185] use orthographic and head-noun features. Yamamoto et al. [214] explore a mix-and-match approach with a variety of features including boundary, morpho-lexical, and syntactic properties as well as a binary dictionary feature (indicating existence of the word in dictionary). Given how severely affected by the choice of features, classification-based approaches can be, automatic feature selection is an interesting and important consideration [179]. A systematic evaluation of common features and discussion of their influence on the resulting predictive power of classification-based NER systems, can be found in [76].

As mentioned earlier, sequence-labelling systems consider sequences of words instead of individual words or phrases. They too, are trained on tagged corpora but aim to predict the mostly likely tags for an observed sequence of tokens. A common statistical framework used, not only for NER, but also for POS-tagging, is the Hidden Markov Model (HMM) [55, 176, 134, 101]. However, it needs to be noted that, in principle, HMMs assume conditional independence between features, whereas, discriminative models like Maximum-Entropy (MaxEnt), do not. In sequence tagging tasks, special-purpose features, that incorporate domain specific knowledge, can be designed. Useful sequence tagging features, such as capitalisation, POS-tag or apposition, are often non-independent. To that end, McCallum et al. [123] proposed the maximum-entropy Markov Model (MEMM), which assumes that the unknown values to be learnt are somehow connected in a Markov chain rather than being conditionally independent of each other. As an alternative, but closely related, to HMMs, MEMMs replace the transition and observation functions with a single function $P = (s|s'o)$ that provides the probability of the current state given the previous state and the current observation. In this model, as in most applications of HMMs, the observations are given - reflecting the fact that we do not actually care about their probability, only the probability of the state sequence (and hence label sequence) they induce [123]. In contrast to HMMs, in which the current observation only depends on the current state, the current observation in a MEMM may also depend on the previous, or in fact, the N previous states. It can then be helpful to think of the observations as being associated with the state-transitions rather than with states themselves. Due to the increased freedom in choosing features to represent observations, MEMM-based methods have been very common and successful in NER tasks [66, 57]. That said, Conditional Random Fields (CRFs) are often shown to exhibit superior performance in biomedical NER [145, 173]. A Conditional Random Field (CRF) is an undirected graph whose nodes correspond to $X \cup Y$, where Y is a set of target variables and X is a (disjoint) set of observed variables. The graph is parametrised with a set of factors $\phi_1(D_1)$, ... , $\phi_n(D_n)$, in the same way as a regular Markov network (which they are descendants of), but instead of encoding the distribution $P(X,Y)$, the conditional distribution $P(Y|X)$ is encoded instead. In order to have this kind of parametrisation correspond naturally to a conditional distribution, we want to avoid maintaining a probabilistic model over X and therefore, CRFs exclude candidates that involve only variables in X [104]. This is one of the main strengths of CRFs as it allows us incorporate into the model a rich set of observed variables whose dependencies may be complex or poorly understood. It also allows us to include continuous variables whose distribution may not be a simple parametric form, which in turn, allows the use of domain-specific knowledge in order to construct highly informative features without worrying about modelling their joint distribution [104]. A CRF will encode a conditional distribution as follows:

$$\begin{aligned} P(Y|X) &= \frac{1}{Z(X)}\tilde{P}(Y,X) \\ \tilde{P}(Y,X) &= \prod_{i=1}^{m}\phi_i(D_i) \\ Z(X) &= \sum_{Y}\tilde{P}(Y,X) \end{aligned} \tag{2.1}$$

2 variables in H (non-chordal Markov network) are connected by an (undirected) edge whenever they appear together in the scope of some factor [104].

CRFs seem to consistently outperform MaxEnt and HMM approaches for most TM taks [16, 121], at least with regards to accuracy. Generally speaking, conditioning the target on what was observed (CRFs & MaxEnt) provides benefits over generative modelling approaches (HMMs) where the entire joint distribution of X needs to be encoded [104]. Similarly with MaxEnt, a CRF will make no assumptions about the data (contrary to the independence assumption made by HMMs). To phrase it differently, both put emphasis on what was actually observed during training. On the other hand, their most prominent difference is how they encode the conditional distribution. We already saw that a CRF is globally conditioned on some observation X. The same is true for MaxEnt, only this time, the conditioning is not global - the model makes a decision for each state independently of the other states [88]. With this in mind, we could say that a CRF is essentially a MaxEnt model over the entire sequence [121]. Thinking of it this way, also helps us understand why CRFs are more expensive to compute than MaxEnt. While MaxEnt will find the parameter values that maximise the conditional likelihood of each class separately (local maximum), a CRF will find the values that maximise the conditional likelihood of all classes (global maximum).

The intuition behind MaxEnt is to build a distribution by periodically adding features that only pick out a subset of the observations. The total distribution is then constrained by those features to match the empirical distribution observed in the training set. Finally, the most uniform distribution which accords with the constraints, is chosen [123]. At first glance, this may seem perverse but the principle is conceptually very simple. We need to model everything that is known (observed) but assume nothing about that which is unknown (not-observed). Therefore, a MaxEnt model will not assume anything it has never encountered before, thus, will never go beyond the training data. A MaxEnt model will encode a conditional distribution as follows:

$$\begin{aligned} P(Y|X) &= \frac{1}{Z} \exp \sum_i w_i f_i \\ Z &= \sum_C P(Y|X) \end{aligned} \tag{2.2}$$

(f_i denotes feature i and w_i its weight)
(Z is just a normalisation factor to make the probabilities sum to 1)

By looking at how MaxEnt encodes the conditional probability between 2 variables, it becomes apparent why MaxEnt belongs to the family of exponential (also known as *log-linear*) classifiers.

Many approaches are not restricted to a single method for performing NER but rather rely on multiple techniques and various resources [179]. These hybrid approaches have been shown to be quite effective in combining dictionary or rule-based approaches with statistical methods [151]. To demonstrate the advantages of hybrid approaches, Abacha et al. [17] performed a comparison of common rule-based and statistical approaches to medical NER, and concluded that hybrid approaches utilising both ML and domain-specific knowledge exhibit superior performance [179]. Numerous hybrid biomedical NER systems have been developed over the years. For example, Sasaki et al. [171] follow a dictionary-based approach to protein-NER, in parallel with POS-tagging. A CRF-based statistical approach is then utilised to reduce the number of false positives and negatives in the resulting tagged sequence. Other methods attempt to create meta-learners from multiple statistical models [179]. A good example of this is demonstrated by Zhou et al. [218], who identify protein and gene names by deploying a meta-learner derived from two HMMs (trained on distinct corpora), whose outputs are combined with an SVM. Similarly, a meta-learner for protein-NER is derived from three SVMs (trained on different corpora and feature sets), whose outputs are then combined with a fourth one, in [126]. Finally, Piliouras et al. [151] merge predictions originating from DrugBank, a MaxEnt model and genetically evolved lexical patterns, in order to overcome the issue of data-sparsity. Cai and Cheng [43] also propose consulting several ML classifiers for improved generalisation and stability in the system.

2.3.2 Relation Extraction

NER is only the beginning for most real-world IE tasks in biomedicine. The next step involves determining associations between the captured NEs. The association can only be concerned with only 2 entities, in which case it is called a 'binary association', or in fact more, which is rather common in biomedicine. These complex associations are discussed later in Section 2.3.3. Simpson et al. [179] define relation extraction to be the task of identifying occurrences of certain types of relationships between pairs of NEs. Perhaps more importantly, it is emphasised in [179] that, "although common entity classes (e.g., genes or drugs) are generally quite specific, the types of identified relationships may be broad, including any type of biomedical association, or they may be specific, for example, by characterizing only gene regulatory associations".

IE tasks have centred around a variety of biomedical relations. In the current genomic era, genes, proteins and their pairwise interactions have been the prime suspects of most of the surrounding work [179]. In addition, protein-protein interactions (PPIs) have been extensively researched in biomedical IE, as they form the basis of our understanding of bio-processes [179]. Other interesting research targets interactions between proteins and point mutations [116], proteins and their binding sites [45], genes and diseases [51] and genes and phenotypic context [122]. Considering the ever-growing prominence of electronic health record systems in medicine [205], researchers working within the clinical domain have focused more on relationships between patients' conditions and the corresponding tests and/or treatments [179].

Biomedical relation extraction faces many of the same challenges as NER, including of course, the construction of extensive, high quality annotated corpora for training and evaluation purposes. Additionally, Simpson et al. [179] make it clear that, *"when compared with the annotation of NEs, the annotation of relations is considerably more complex due to the fact that these are generally expressed as discontinuous spans of text and the types of relations considered are usually application-specific [26]. Moreover, since there is often little consensus regarding how to best annotate certain types of relations, the resulting resources are largely incompatible, and, as a consequence, the quality of the methods relying on these resources is difficult to evaluate. For example, Pyysalo et al. [153] performed a comparative analysis of five PPI corpora and discovered that the performance of state-of-the-art PPI extraction systems, varied on average by 19 percentage points and by as much as 30 percentage points on the evaluated corpora. Participation in community-wide evaluations that are dedicated to the relation extraction task is of utmost importance for obtaining annotated corpora".*

Relation extraction tasks have been a component of several recent evaluation tasks. The LLL genic interaction challenge [143], the BioCreAtIve PPI extraction task [108], and the *i2b2* relation extraction task [205] are good examples. The LLL challenge aimed to extract protein and gene relationships from MEDLINE abstracts with the best-performing system reporting an F-score of 54%. The BioCreAtIve task had a wider scope and consisted of four subtasks, all related to PPI extraction. These challenges included the classification of PubMed abstracts as relevant or irrelevant for PPI annotation, the identification of binary PPIs from full-text articles, the extraction of protein interaction methods, and the retrieval of the text where the association occurs. The best-performing system reported an F-score of just under 35% for extracting binary PPI relations. Finally, within the clinical domain, the i2b2 relation extraction challenge aimed at capturing medical problem-treatment, problem-test, and problem-problem relationships in clinical notes [179]. In particular, participants were given tasks, such as, to determine whether two co-occurring problem and treatment concepts were related and if so, to capture clues of improvement or worsening of patients' condition after treatment. The best performing system on this challenge achieved an F-score of 74%. Simpson et al. [179] point out that, *"much like the forums dedicated to evaluating the NER task, community-wide evaluations like these have undoubtedly played an instrumental role in the development and evolution of relation extraction approaches and resources"*.

Relation extraction approaches have become more and more sophisticated over time. Their evolution gradually shifted from simple techniques such as co-occurrence statistics, to complex methods utilising syntactic analysis and dependency parsing [179]. Popular approaches to the relation extraction task are described next.

Collecting co-occurrence statistics, that is, how often terms appear close together, can be considered the simplest method for inferring whether two NEs are related or not. Higher frequency of co-occurrence typically translates to greater chances of some relation existing between them. Of course, co-occurrence alone says nothing about the actual type, or the direction of the relation [179]. Nonetheless however, the importance of other properties should not be underestimated. For instance, co-occurrence statistics often provide useful hints as to how strong (or weak) some relation might be, and that is very useful in its own right [179]. The study by Chen et al. [48], presents a perfect example of such analysis. They were able to compute how strongly certain drugs are related to certain diseases, simply by analysing the co-occurrence statistics of these entities as witnessed in clinical records and small parts of the literature. It is not unusual for systems that rely solely on co-occurrence, to exhibit high recall and low precision [179].

Other methods typically involve the acquisition of extraction rules and/or context-free-grammars (CFGs) [186, 150]. Rule-based methods aim to capture the inherent linguistic patterns underlying certain relation types. High precision and low recall (the opposite of co-occurrence approaches) is to be expected from such systems [179]. Much like NER, the task of crafting the rules can be mitigated to domain experts [208], or they can be derived automatically from annotated corpora using ML [77]. As usual, both options come with their own set of requirements and trade-offs. More concretely, outsourcing the task to a domain expert will most likely deliver optimal results, but is bound to be expensive and of course, requires that such an expert, not only exist in the first place, but is also able and willing to undertake the task. Much in the same way, attempting to analyse large amounts of annotated corpora using ML algorithms requires that such corpora exist or that it is somehow easy to construct, and that their licence allows such access and usage.

Relations involving biomedical entities have been the target of many statistical approaches [179]. A supervised machine learning system, trained on shallow features extracted from oncology reports, which is able to detect and extract various clinical relationships in patient narratives, is presented by Roberts et al. [168]. On the same domain, Rink et al. [167] uncovered relations between medical conditions, treatments and tests mentioned in electronic medical records using a system that combines supervised ML and lexical, syntactic, and semantic context features. Similarly, relations between diseases and treatments from PubMed abstracts, but also between genes and diseases in the human GeneRIF database, were targeted by CRFs in [41]. Finally, much like in NER, hybrid methods are becoming ever more popular. For instance, such an approach is presented in [32], where the authors make use of hand-crafted patterns developed by domain experts in conjunction with SVM classification to identify relations occurring between diseases and treatments in biomedical publications.

Exploiting syntactic information has recently received a lot of attention by relation extraction researchers [179]. In particular, the idea that relation extraction can be performed on the output of dependency parsers has been widely explored. For example, Fundel et al. [71] extracted abstracts from MEDLINE and used them to produce dependency trees. Three simple relation extraction rules were subsequently applied to the resulting syntactic structures in order to identify gene and protein associations. Similarly, a combination of syntactic patterns obtained from dependency parses, were used by Rinaldi et al. [165], in order to be able to make the literature act like a database which can be queried for interactions between proteins and genes. Miyao et al. [132] demonstrated that MEDLINE abstracts can be annotated with predicate-argument structures through the use of deep parsing. Relational concepts are then uncovered by structurally matching the semantic annotations. In later work, Miyao et al. [133] presented a comprehensive overview of various parsers and their output representations. An evaluation regarding their ability to boost accuracy was also performed (in the context of a PPI extraction system) [179].

Given that the availability of large corpora containing relational annotations is constantly growing, many approaches these days tend to prefer ML algorithms to extract useful information from syntactic structures instead of applying hand-crafted rules [179]. In fact, kernel-based ML research, has produced kernels capable of measuring the similarity between syntactic parse trees or graphs [179]. A good example of an approach which utilises such kernels is given by Airola et al. [18], who describe an all-paths graph kernel for computing the similarity between dependency graphs. The kernel function is then used in training a "least-squares" SVM, which in turn, identifies PPIs. On a similar note, four genic relation extraction kernels based on the shortest syntactic dependency path between two NEs, are suggested by Kim et al. [100]. Lastly, Miwa et al. [130] propose a framework for combining the output of multiple kernels with that of syntactic parsers, for PPI recognition.

Syntactic analysis is often complemented by shallow semantic parsing, which consists of the detection of semantic arguments associated with the predicate of a sentence and their subsequent classification into 'roles'. For instance, given the sentence *"Alice sold the car to Bob"* the task would be to recognize the verb "to sell" as representing the predicate, "Alice" as representing the seller (agent), "the car" as representing the goods (theme), and finally, "Bob" as representing the recipient. A semantic representation of this kind is at a higher-level of abstraction than a syntax tree. This means that even though the syntactic form may change (i.e. active to passive voice), the semantic roles involved will stay the same. Such a role-labelling system (BIOSMILE) was developed by Tsai et al. [191]. Internally, it uses a MaxEnt model to extract biomedical relations from a portion of the GENIA corpus. As discussed next, the semantic role labelling of biomedical NEs has enabled the extraction of many complex associations and interactions [179].

2.3.3 Event Mining

The last six to seven years, the focus of modern biomedical IE has shifted from identifying binary relations towards the significantly more difficult and ambitious task of extracting complex, and often, nested *events*. *Events* are typically characterised by verbs, potentially nominalised [179]. For example, given the sentence "glnAP2 may be activated by NifA", the verb "activated" specifies the event, and *glnAP2*, *NifA* specify the arguments to that event. Contrary to the the case of simple binary relations, an event and its arguments are tagged with both concept labels and semantic roles [179]. In the example given above, the verb *activated* indicates an event of type "positive-regulation", whose arguments are expected to be a protein (NifA) (acting as the cause) and a gene (glnAP2) (acting as the theme) [26]. Another important differentiation is that events are fully composable, and thus, can be nested, with one event functioning as a participant to some other. Such a composition can appear even if the sentence is short and structurally simple. For example, in the sentence "RFLAT-1 activates RANTES gene expression" two events are encountered [26]. One of them is indicated by the nominalised verb *expression* whose theme is RANTES (a gene), and the other one is indicated by the verb *activates* whose cause is RFLAT-1 (a protein) and whose theme is the gene expression event itself. Consequently, event representations are capable of expressing many different types of interactions with an arbitrary number of entities and events related by an assortment of thematic roles [179].

Since, both the syntactic and semantic structure of a sentence needs to be analysed, certain semantic processing and deep-parsing techniques have proven extremely useful for effective event-mining (EM) [179]. Given that, events can be effectively represented by predicate-argument relationships [210], one can imagine why dependency-parsing in particular, can be an invaluable aid for EM. Despite the overall complexity of the task, EM can be applied to many aspects of biomedical research. According to Simpson et al. [179], the annotation of biological pathways and the enhancement of existing databases are examples of tasks increasingly being targeted by EM systems.

Similarly with NER and relation-extraction, the growing interest in EM these days has been driven, in no small part, by the availability of corpora containing the annotations necessary for the training and evaluation of statistical EM approaches. The BioInfer corpus [154] was the first publicly available corpus to incorporate such annotations, in the general biomedical domain. Not long after, other event-annotated corpora followed suit, including the GENIA Event Corpus [100] and the Gene Regulation Event Corpus (GREC) [189]. Interestingly, the GENIA corpus remains one of the most widely used resources in biomedical TM. In fact, the data-sets for early BioNLP shared tasks on EM [97, 99] were prepared based on GENIA [179].

The first community-wide evaluation of EM methods was the BioNLP shared task of 2009 [97]. Its main challenge was to identify events related to protein biology abstracts contained in MEDLINE. The actual event types that were targeted included gene expression, transcription, localisation, binding and regulation. The *binding* event type was significantly more demanding than the rest, as it required the detection of an arbitrary number of arguments. By the same token, events of type *regulation*, were notably complex since they allow other events to act as their cause or theme (nesting) [179]. The best-performing system reported an F-score of 52% on the primary task. On the next BioNLP shared task of 2011 [99], the evaluation from the previous meeting was reiterated, but also augmented by including further tasks targeting events in other biological sub-domains. A decent improvement in the community was witnessed on the the subtask directly comparable with that of the first meeting, as an F-score of 57% was reported by the best-performing system. Systems submitted at the BioNLP shared task meetings utilised an array of techniques including ML, Markov-Logic networks, and of course, dependency-graphs [179]. Approaches to biomedical EM are described next.

Most EM systems follow a step-wise approach that divides the task into a sequence of three distinct phases [179]. Typically, the first phase is concerned with predicting a candidate set of event triggers. Trigger words are often the verbs (or nominalised version thereof) that indicate a particular event type (e.g. "binds", "activates", "inhibits"). Determining whether any recognised NEs or trigger words are manifestations of event arguments, comes next. The final stage in the process focuses around attaching arguments to event triggers according to constraints on the type, and sometimes the number of, arguments allowed by a particular event type. In addition, certain state-of-the-art EM systems augment this process by adding an extra step which usually targets negation or speculation cues [141] (i.e. EventMine [131]).

The aforementioned high-level architecture is a common approach to the EM task, as witnessed by the majority of participating systems on the BioNLP EM challenges. For instance, the the best-performing system on the BioNLP '09 EM task, described in [37], used an extensive set of features to train separate multi-class SVMs for detecting event triggers and arguments. In particular, features derived from dependency graphs were heavily relied on. The system then used a set of hand-crafted rules to tag recognised events with their corresponding arguments. In later work, the authors combined this approach with the BANNER NER system, in order to perform EM on an untagged subset of citations from PubMed [36]. An EM approach similar to that of Bjorne et al., is given by Miwa et al. [129]. However, instead of relying on manually derived rules for attaching event participants to triggers, an improvement is demonstrated, stemming from the use of a ML classifier with additional features, tailored specifically for this step. On the other hand, Buyko et al. [42] construct a dictionary of common event triggers. Event participants are then recognised by an ensemble of feature and kernel-based classifiers. Similarly, Kilicoglu et al. [96] also use a dictionary-based approach to identify triggers, but instead of relying on ML, they manually develop rules and heuristics based on syntactic dependency paths to detect participants. Finally, a pattern-based approach is presented by Cohen et al. [54]. Their method relies on the ontology-driven OpenDMAP system [87] in order to define the types of entities and events, as well as the potential constraints surrounding their respective arguments.

More recently, joint prediction approaches have been explored [179]. These primarily seek to address the issue of errors flowing downstream, which most of the above approaches allow. For example, by decoupling the *event-trigger* and *argument-detection* tasks, a system will most likely fail to extract an event, unless it detects the trigger first. A method which jointly predicts events and arguments, is proposed in [152]. Their system, being a Markov-logic based one, predicts, for each word, whether it is a trigger, and for each syntactic dependency edge, whether it is an argument path towards an event theme/cause [179]. Following this work, a small set of joint prediction models is argued to be computationally simpler than previous work [162] and lead to better EM performance, by Riedel and McCallum [163].

2.3.4 Numerical expressions

Numerical expressions are a common phenomenon in both text and speech, and of special importance to all areas of science. Nonetheless, they also present special challenges to NLP applications. According to Habash et al. [75], the challenges behind identifying numerical expressions include:

- an infinite set of possible expressions
- multiple script formats (e.g., digits, multi-token sequences or a mix of both)

Of course, not all NLP applications have the same needs and/or aims. For example, in machine translation numerical expressions are typically subject to normalisation, such that converting them to the desired target language is straight forward [75]. On the other hand, in speech recognition the model could have been trained on digit-free samples. Similarly, applications aiming to convert text to speech, will have to do several layers of transformations in order to go from a written numerical expression to a spoken word. In IE, it is important to be able to associate these numerical expressions with the entities they refer to and also, with surrounding modifiers, in order to enable semantic inference. Therefore, to be able to process such expressions in realistic contexts, methods to determine the exact expression span and to convert its content into a normalised, digit-based form are needed. Being able to generate digits from some normalised form can also be useful but is a significantly easier task [75].

According to Habash et al.[75], the majority of work on number identification has focused on out-of-context conversions from word to digit (and vice versa), and even though work on this front has been carried out for many languages, it remains exclusively rule-based. Examples include English [161], Swedish [178], Finnish [92] and Arabic [19, 58]. Especially impressive is the 'Numeral Translator' [39], which is able to translate numerical expressions to over 80 languages. However, on top of being out-of-context, these approaches are not by any means perfect. That is, they often fail to handle even small variations of input formats, perhaps because the authors have chosen to keep their rule-set small. Reusing the example from [75], *"the aforementioned Numeral Translator can translate 'forty-seven thousand three' into '47003', but cannot parse 'forty-seven thousand and three' or 'forty seven thousand three' (omitted hyphen), which are common variations"*.

Apart from general numerical expressions, existing research has also targeted highly specialised categories such as, temporal and monetary expressions (quantifying time and money respectively). Temporal expressions are particularly interesting for the biomedical and clinical domains, as they usually denote key aspects of certain parameters, treatments and/or symptoms. Despite the specificity difference between the two categories, the methods utilised for identifying, extracting and normalising numerical expressions remain similar across both. For example, Kovacevic et al. [106] describe a system which combines rule-based with ML approaches that rely on morphological, lexical, syntactic, semantic, and domain-specific features. The ML module of their system uses CRF models trained for event extraction, while recognition and normalisation of temporal expressions, are handled by manually crafted rules.

Technically speaking, raw integers can be extracted effectively and efficiently with regular-expressions. More complex numbers such as arbitrary decimals and composite expressions that combine alpha-numeric characters (e.g., 8,15 mg/h) need to be decomposed to their constituents and handled separately. Regular expressions do allow for internal grouping but are very error-prone when scaled and, more importantly, they do not compose. As a consequence, on the programming level, taking a composite expression apart can only be expressed as a monolithic transformation (A $\rightarrow$ Z). We pursue the idea of viewing the same decomposition as a set of discrete transformations derived from composing functions (A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ .... Z). As it will be explained in Chapter 4, this has a couple of interesting properties. In short, since the decomposition is lifted a level higher than regular-expressions, we can mix and match components (functions) such that each constituent is handled by whichever component is more appropriate. For example, when identifying drug prescriptions with regular-expressions, one will probably try grouping the drug-name and the dose separately within the same regular-expression. Ideally, we would like to be able to express that the *drug-prescription-recogniser* is a composition of the *drug-recogniser* and the *dose-recogniser*, but there is no way to do that with regular-expressions. Moreover, the dose itself can be considered a composite entity as it typically includes some numerical value, some unit of measure and some interval. It quickly becomes apparent that a pure regular-expression based solution will quickly become extremely fragile for any sufficiently complex pattern. On the contrary, viewing the transformation in discrete steps allows for heterogeneous models to be combined. Regular expressions can still be used wherever appropriate. In the example presented here, we could use a dictionary as the *drug-recogniser*, a regular-expression to capture all expressions involving numbers, which are then passed to a statistical model that gives the final answer (*dose-recogniser*). The models live inside functions which, when composed, give the *drug-prescription-recogniser*.

Pharmaco-kinetic parameters

As hinted earlier, modern drug development relies heavily on PK modelling. Generally speaking, once a drug has entered the body, its PK parameters are constantly subject to change. Such modifications can occur due to a variety of reasons, and are in turn likely to cause further, possibly undesired, alterations in the drug's course through the body. Consequently, one may experience from minor unwanted symptoms to severe injuries, and even death, depending on the extent of the modification. Therefore, it follows that having a clear picture of what can or cannot alter the PK parameters of a particular molecule is invaluable information for a drug developer and in fact, for anyone involved in the drug production $\rightarrow$ prescription $\rightarrow$ consumption pipeline.

To our knowledge, mining for PK parameters has only been reported by one other research group so far. However, there have been studies addressing general kinetics parameter mining issues. Ordinary differential equations (ODEs) used for general biological kinetic system (i.e. enzyme kinetics) modelling are quite similar to those used in drug modelling. Therefore, strategies for kinetics parameter mining can have high relevance to drug PK parameter mining. One study [192] presented a tool (KIND) for kinetics parameter extraction. As part of its NER phase, labels describing type, value and annotation span were POS-tagged and used for feature detection. Subsequently, the tagged entities were then matched against a collection of lexical rules, specifically tailored for identifying and extracting these parameters and their related annotations. The authors report decent performance in both classifying relevant sentences, and kinetic parameter extraction ([P = 76% - R = 87%] and [P = 75% - R = 90%] respectively). Another, rather similar, TM algorithm [79] for chemical and biological kinetics data relies on dictionary-NER prior to a rule-based assocication of the recognised terms. The dictionaries are manually developed by experts. This method was tested against PubMed abstracts, where manual verification of the results showed recall values between 51% and 84%, and precision values ranging from 55% to 96%, depending on the category explored (organism, enzyme or ligand) [209].

The one study that truly focuses on extracting PK numerical parameters and their potential modification is that of Zhiping et al. [209] and his subsequent thesis, which describes their sequential mining strategy. Firstly, an entity template library was built in order to retrieve PK relevant articles (IR phase). A set of tagging and extraction rules were then deployed on the retrieved abstracts, to identify, and subsequently extract, PK data (IE phase). Moreover, in order to estimate the PK parameter population-average mean and between-study variance, the author(s) developed a linear mixed meta-analysis model and an expectation-maximisation (E-M) algorithm to describe the probability distributions of PK parameters. Finally, a cross-validation scheme was adopted to clear out false-positive mining results. Using this approach to mine PK data for the drug *midazolam*, an 88% precision rate and 92% recall rate are reported, balancing to an F-score of 90%. It appears to outperform an SVM based mining approach, which led to an F-score of 68.1%. According to the author(s), repeating the experiment on seven additional drugs leads to similar performance.

Another recent study concerned with extracting and classifying, not necessarily PK, but numerical expressions in general, is that of Narisawa et al. [93]. Their work mainly focuses on novel methods for modelling numerical common sense: *"the ability to infer whether a given number (e.g., three billion) is large, small, or normal for a given context (e.g., number of people facing a water shortage)"*. Two approaches for acquiring numerical common sense are explored. According to the authors [93], *"both approaches start with extracting numerical expressions and their surrounding context from the Web. Context is defined to be the verb and its arguments that appear around a numerical expression. The first approach estimates the distribution of numbers co-occurring within a particular context and examines whether a given value is large, small, or normal, based on the distribution. The second one utilises textual patterns with which speakers explicitly expresses their judgement about the value of a numerical expression. Both approaches were shown to be effective for numerical expression extraction and reasoning"*.

In order to extract and aggregate numerical expressions in various documents, they mapped the numerical expressions to richer semantic representations and extracted their context. It is argued that *"the semantic representation of a numerical expression consists of three fields: the value or range of the real number(s), the unit (a string) and optional modifiers"* [93]. During normalisation, spelling variants (e.g., kilometre and km) were captured and auxiliary units were transformed into their corresponding, predefined canonical units (e.g., 2 tons and 2,000 kg becomes 2,000,000 grams). In addition, certain accompanying modifiers such as 'over', 'about', or 'more than', affect the value accordingly.

Since most PK modifications are quantifiable and usually expressed in literature in the context of some comparison or condition, the aforementioned work of Narisawa et al. has high relevance for extracting PK numerical parameters. In particular, certain aspects of extracting and normalising numerical expressions were adopted and successfully applied for extracting reported numerical drug properties (e.g., dosage), in this thesis (see Chapter 4).

Chapter 3

Dealing with data sparsity in drug NER

3.1 Introduction

As explained in the previous chapter, NER is the task of identifying members of various semantic classes, such as *persons*, *mountains* and *vehicles* in raw text. In biomedicine, NER is mainly concerned with the following classes: *proteins*, *genes*, *diseases*, *drugs*, *organs*, *DNA sequences*, *RNA sequences*, but possibly others too [52]. Drugs (as pharmaceutical products) are special types of chemical substances with high relevance for biomedical research and in particular, PK/PD modelling. An overview of approaches to NER was also given in the previous chapter. For example, a simplistic and rather naive approach to NER is to directly match textual expressions found in a relevant lexical repository against raw text. However, recall that, even though this technique can sometimes work well, more often it suffers certain serious limitations. Firstly, its accuracy heavily depends on the completeness of the dictionary. However, as terminology is constantly evolving, especially in bio-related disciplines, producing a complete lexical repository is far from feasible. Secondly, direct string matching completely overlooks ambiguity and variability issues [25]. Ambiguous dictionary entries refer to multiple semantic types, and therefore contextual information needs to be considered for disambiguation. Typically, statistical learning models are deployed to address these issues.

[0]This chapter was published as a full paper in Healthcare Informatics (ICHI), 2013 IEEE International Conference `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6680456`. An extended version was recently submitted for peer-review to the Journal of Artificial Intelligence in Medicine (AIM)

In such approaches the entire task of NER is formalised as a classification problem in which an input expression is either classified as an entity or not. Supervised learning methods are reported to achieve superior performance than unsupervised ones, but previously annotated data are essential for training [25]. Human annotated data, also known as gold-standard data, guarantee best results in exchange for the cost and manual effort of experts. For this reason, manually annotated corpora for NER are often of limited size and for a particular domain.

In this Chapter, we present a method for improving drug NER performance in cases where either very limited or no gold-standard training data is available. Our method includes of a *voting system* able to combine predictions from a number of recognisers. Moreover, genetic-programming (GP) was used in order to evolve string-similarity patterns based on common suffixes of single-token drug names occurring in DrugBank [211]. In succession, these patterns are used to compile regular expressions in order to generalise dictionary entries in an effort to increase coverage and tagging accuracy.

We compare the performance of several 'standard' NER approaches against ours on a single gold-standard testing-corpus (PK corpus [213]). Even in the worst-case scenario, where no gold data is available, our best combination of non-gold models exhibits classification performance comparable with that of the gold model. Having access to robust drug NER models is extremely useful for higher level extraction tasks that totally depend on recognising these entities, such as mining for drug-drug interactions (DDIs) or in fact, PK parameters, which is the long-term goal of this thesis.

The rest of this paper is organised as follows: section 3.2 summarises previous work on ways of dealing with data sparsity in general NER (a detailed overview of bio-NER approaches is provided in Chapter 2). Section 3.3 describes the dictionaries and data used in our experiments, as well as the experimental methodology followed. Section 3.4 discusses the experiments and their results, followed by 'Discussion' and 'Conclusions & Future work'(Sections 3.5 and 3.6 respectively).

3.2 Related Work

Usually data sparsity in NER is dealt with by generating data semi-automatically or fully automatically. However, the resulting data is of lower quality than gold-standard annotations. Since supervised learners are based on annotation statistics, existing research has mainly focused on quick automatic generation of good quality of training data. Towards the same ultimate goal, our approach aims to overcome the restrictions of data sparsity or unavailability in the biomedical domain. In the next 3 paragraphs we present recent advances in the area of automatic generation of data, whereas, in the last paragraph of this section we focus on techniques tailored to deal with existing, limited amounts of data.

Usami *et al.* [201] describe an approach for automatically acquiring large amounts of training data from a lexical database and raw text that relies on reference information and coordination analysis. Similarly, Vlachos and Gasperin [207] obtain noisy training data by using a few manually annotated abstracts from FlyBase (`www.flybase.org`). Their approach uses a bootstrapping method and context-based classifiers to increase the number of NE mentions in the original noisy training data. Even though they report high performance, their method requires some minimum curated seed data. Similarly, Thomas *et al.* [187] demonstrated the potential of distant learning in constructing a fully automated relation extraction process. They produced two distantly labelled corpora for protein-protein & DDI extraction, with knowledge found in databases such as IntAct [95] for genes and DrugBank [211] for drugs. *Active learning* (AL) is another framework that can be used for reducing the amount of human effort required to create a training corpus [59, 188]. In AL, the most informative samples are chosen from a big pool of human annotations by a Maximum Likelihood model in an iterative and interactive manner. It has been shown that active learning can often drastically reduce the amount of training data necessary to achieve the same level of performance compared to pure random sampling [190].

A related approach, *"accelerated annotation"*, is presented in [196]. Similarly to AL, this framework allows one to produce named entity (NE) annotations for a given corpus at reduced cost. However, unlike AL, it aims to annotate all occurrences of the target NEs, thus minimising the sampling bias, which is inevitable in AL approaches. Despite the similarities between the two frameworks, their goals are different. While AL aims to optimise the performance of the corresponding tagger, accelerated annotation aims to construct an unbiased NE annotated corpus.

On a rather larger scale, the Collaborative Annotation of a Large Biomedical Corpus (CALBC) initiative [160] is a European Support Action concerned with the automatic generation of a very large, community-wide shared corpus annotated with a wide range of biomedical entities. Generating such a corpus requires that annotations from different automatic annotation systems are integrated and harmonised.

Even though there is great value in generating quality training data automatically or semi-automatically, other researchers have focused on making the most of existing annotations. Tsuruoka *et al.* [193] used logistic regression to learn a string similarity measure from a dictionary, useful for soft string matching. In contrast, our approach learns string patterns. Kolarik *et al.* [103] utilised NLP techniques to extract drug-related term candidates from textual expressions found in DrugBank. They automatically augmented these resources with novel descriptions of pharmacological effects of drugs by applying these lexico-semantic patterns to MEDLINE abstracts. Similar methods have also been applied to tasks as recognising drug-disease interactions [48] and interactions between compounds and drug-metabolising enzymes [64]. Hettne *et al.* [81] describe a rule-based approach, primarily intended for term filtering and disambiguation. It helps to identify names of drugs and small molecules by incorporating several dictionaries such as the UMLS, MeSH, ChEBI, DrugBank, KEGG, HMDB and ChemIDplus. They report an overall performance of 67% precision and 40% recall of their combined dictionary on the Fraunhofer corpus. An earlier experimental system, EDGAR [166], extracts both genes and drugs as well as relationships between them by combining several databases with statistical NLP methods. Unfortunately, no evaluation results are reported. In the clinical domain, Sanova *et al.* [172] released the *clinical Text Analysis and Knowledge Extraction System* (cTAKES), as open-source software. Their system, which builds on top of other popular open-source projects such as UIMA and openNLP, was designed for information extraction from electronic medical records (EMR). They report 71.5% F-score for their NER component when tested against their private gold-standard test set. Sasaki *et al.* [171] followed a bi-level dictionary-based statistical approach. Firstly, protein name candidates are located using a dictionary. Strings are mapped to parts of speech (POS), where the POS tag-set is augmented with a special tag for proteins. In succession, sequential labelling applies to reduce false positives and false negatives in the POS/PROTEIN tagging results. The model can be expanded by manually adding NE entries to the dictionary, not by re-training. An F-score of 73.78% is reported on protein NER against the JNLPBA-2004 shared task test set, which contains 404 MEDLINE abstracts.

3.3 Data and Methods

The proposed method requires at least two key resources: a comprehensive lexical repository, such as a dictionary or lexicon, and large amounts of raw text in the same domain. A small gold-standard corpus can enhance NER performance if available. Our method could potentially be applied to recognise any type of biomedical named entities. Significant progress has already been made in recognising genes and proteins and so we focused on identifying entities of type *drug*.

3.3.1 Data

DrugBank

As our dictionary, we chose to use DrugBank [211] because it is relatively up-to-date and it provides a mapping between drug-names and common synonyms. DrugBank currently contains more than $6,700$ entries including 1447 FDA-approved small molecule drugs, 131 FDA-approved bio-tech (protein/peptide) drugs, 85 nutraceuticals and 5080 experimental drugs. We pre-processed the dictionary by normalising and mapping all official drug terms to their synonyms.

PharmacoKinetic Corpus

The PharmacoKinetic Corpus [213] is manually annotated and consists of 240 MEDLINE abstracts annotated and labelled on the basis of MESH terms relevant to Pharmacokinetics such as drug names, enzyme names and pharmacokinetic parameters, e.g. clearance. Half of the corpus is intended for training (invivo/invitro-train) and half for testing (invivo/invitro-test). It is freely available at: `rweb.compbio.iupui.edu/corpus/`. As a pre-processing step, all annotations concerning entities other than drugs were removed, since this study is concerned with detecting drugs only.

Raw text

Nowadays, acquiring large amounts of raw text is not a difficult task, even for very specialised domains. Public electronic repositories of open-access articles exist for most scientific domains and usually can be queried via RESTful web services. In biomedicine, for example, UK PubMed Central (Europe PMC, `www.europepmc.org/`) is an article database which extends the functionality of the original PubMed Central (PMC, `www.ncbi.nlm.nih.gov/pmc/`) repository. For the purposes of this study we used a small subset of the entire UKPMC database which includes more than $2,000,000$ papers. The sample we used was created by Mihăilă and Navarro [125], totalling 360 pharmacology and cell-biology related articles. As a pre-processing step, the corpus was sentence-splitted and tokenised. Part-of-speech (POS) tagging was omitted from the process since we did not plan to use the POS tags as features during training.

3.3.2 Methodology

In this section we describe our NER classifier. To classify labels of tokens, we used *Maximum-Entropy* (MaxEnt) modelling, also known as *multinomial logistic regression* [34], which tries to maximise the conditional likelihood of classes by assuming that the best model parameters are the ones for which each feature's predicted expectation matches its empirical expectation. In other words, MaxEnt tries to maximize entropy while conforming to the probability distribution drawn by the training set. As a MaxEnt implementation, we used MAXENT.SF, which is part of the Apache openNLP project. The features used in the classifier modelled by MaxEnt are listed below. For each token we calculate:

- **the current and** ± 2 **tokens**
- **character n-grams**: ± 2 tokens
- **sentence**: binary feature indicating if the token appears at start or end of a sentence.
- **token-type**[1] of the current and ± 2 tokens
- **previous map**: a binary feature indicating if the current token was previously seen as a NE.
- **prefix**
- **suffix**
- **dictionary**: binary feature indicating if the token exists in the dictionary

We acquired evaluation statistics for several 'standard' NER approaches in order to establish a baseline. A simple voting-system was developed which is able to aggregate predictions from several NER systems, after hypothesising that the combined output from several NER systems will improve over the results of single classifiers that

[1]Token types:

- INITIAL-CAPITAL-LETTER
- ALL-LOWERCASE-LETTER
- ALL-LETTERS
- ALL-DIGITS
- CONTAINS-PERIOD
- CONTAINS-DIGIT
- CONTAINS-SLASH
- CONTAINS-HYPHEN
- CONTAINS-LETTERS
- CONTAINS-UPPERCASE

were deployed as standalone. Our voting algorithm assumes that DrugBank makes no false-positive predictions. This assumption is not true, if DrugBank includes non-drug entities but, since dictionaries are produced manually, we consider them ideal. Ambiguous NEs might also affect the validity of this assumption. We observed very little such ambiguities in our dictionary, thus, we accept the hypothesis to hold in the domain of drug NEs. Algorithm 1 summarises the voting system.

At a second experimental stage, we de-constructed the dictionary into 2 distinct models: (a) a model trained on text solely annotated by the dictionary, and (b) an evolved set of string-patterns that attempts to accurately cover common suffixes of single-token drug names.

For evaluation, we used the standard Information-Retrieval (IR) metrics: Precision (**P**), Recall (**R**) & F-Score ($\mathbf{F}_1$) [156].

Algorithm 1 Aggregation of predictions

```
List L $\rightarrow$ [ ]
for all $SENTENCES : TEXT$ do
    Map M $\rightarrow$ $\{: prediction : confidence\}$
    for all $TOKENS : sentence$ do
        if dictionary exists then
            if $dictionary.predict(t) \rightarrow POSITIVE$ then
                PUT M $\{prediction\ 1.0\}$
            else
                for all $M : MODELS$ do
                    if $M.predict(t) \rightarrow POSITIVE$ then
                        STORE $\{prediction\ confidence\}$
                    end if
                end for
                PUT M $\{prediction$ max-$confidence\}$
            end if
        end if
    end for
    DROP $overlapping/intersecting$ spans*
    ADD L M
end for
return L
* Rules for dropping spans:
- Identical/Intersecting: first span is kept
- Contained: Contained spans are dropped
```

Classifier	**P**	**R**	**F_1**
Dictionary	**99.7%**	78.5%	87.8%
Dictionary + synonyms	93.4%	78.9%	85.6%
MaxEnt(g[1])	98.3%	84.5%	91.0%
Perceptron(g)	97.5%	72.0%	82.8%
MaxEnt(g) + Dictionary	99.1%	**88.4%**	**93.4%**
Perceptron(g) + Dictionary	97.6%	84.3%	90.4%

Table 3.1: Results of baseline classifiers, trained on gold-standard data

3.4 Experiments

3.4.1 Baselines

Firstly, we tested how the dictionary performs, with and without including synonyms. Secondly, we trained two NE recognisers, namely a MaxEnt and a perceptron classifier, on half the PK corpus (invivo/invitro-train) and tested them on the other half (invivo/invitro-test). Finally, we used our prediction aggregation algorithm to combine predictions originating from the dictionary, with predictions originating from the classifiers.

Table 3.1 presents the results from our baseline experiments. It is worth noting that the pure dictionary-based approach is not 100% precise as our voting system assumes. Careful error analysis revealed that there are at least two entities, i.e. “nitric oxide” and “tranylcypromine” that have not been tagged in the gold-standard corpus. Consequently, the evaluator marks them as false-positives while in fact, they are perfectly correct predictions. Another interesting observation is that including synonyms causes precision to degrade. Synonyms in DrugBank often include acronyms, which have not been tagged appropriately in the test corpus. As before, the evaluator classifies them as false-positives.

In general, we can see that both the dictionary and the classifiers exhibit very high precision and good recall, whereas combining the two has a minimal positive effect on overall performance. The perceptron classifier, despite training significantly faster, consistently showed inferior performance compared to MaxEnt.

Our baseline experiments show that, despite acquiring state-of-the-art precision, there is still space for improvement with regards to recall. High precision indicates that the model extracts some very informative features while training, whereas not so

[1]g $\rightarrow$ *gold*

high recall essentially reflects lack of enough training data. Ideally, we would need more gold-standard annotations, however, as discussed previously, this is not always feasible.

3.4.2 Combining heterogeneous models

Attempting to improve recall, we trained separate models purely on silver data, i.e. data annotated by direct string-matching dictionary entries. Annotation coverage ultimately depends on how up-to-date the dictionary is. DrugBank is a good candidate for this task, as it is a comprehensive dictionary of drugs and also freely available. The 360 full papers mentioned in Section 3.3.1 were annotated and partitioned into 30 collections, each one containing 12 items. This was done in an effort to incrementally check whether the addition of silver annotations has any positive or negative effects on the classifier's performance. We found that we had to include all 30 partitions in order to witness any kind of improvement.

The MaxEnt classifier trained on silver annotation data achieves marginally higher precision and significantly lower recall than the same classifier trained on gold-standard data. This is expected, since the silver annotations reflect the contents of the dictionary, only. Trained on a mixture of gold and silver data, the MaxEnt classifier achieves 0.5% lower precision and 0.3% higher recall than its equivalent trained on gold-standard data.

Including the dictionary boosts the recall of the MaxEnt classifier trained on a mixture of gold-standard and silver annotation data by 1.3% in comparison with its baseline equivalent. The last 2 rows of Table 3.2 show that all statistics were slightly boosted just by utilising these extra, easy to produce silver annotations.

In all our experiments so far, the best achieved recall is 89.7%, far less than precision, thus, we focus on improving it. Careful examination of false-negatives reveals that most of them are either acronyms (e.g. HMR1766), long chemical descriptions (e.g. 5beta-cholestane-3alpha,7alpha,12alpha-triol) or terms whose lexical morphology is particularly different than the usual morphology of drugs (e.g. grapefruit juice). We attempted to capture acronyms by employing a state-of-the-art acronym disambiguator, AcroMine [146], however did not disambiguate any of the acronyms in question, listed below:

- ANF
- PO4
- HMR1766
- MDZ 1'-OH
- E3174
- RPR 106541
- MDZ 4-OH

Classifier	**P**	**R**	**F$_1$**
MaxEnt(s[1])	98.7%	47.1%	63.7%
MaxEnt(s) + Dictionary	99.2%	78.5%	87.6%
Perceptron(s)	98.3%	76.9%	86.3%
Perceptron(s) + Dictionary	**99.3**%	78.5%	87.7%
MaxEnt(g+s)	97.8%	84.8%	91.0%
MaxEnt(g+s) + Dictionary	98.6%	**89.7%**	**93.9%**
Perceptron(g+s)	97.2%	79.1%	87.2%
Perceptron(g+s + Dictionary	98.0%	85.1%	91.1%

Table 3.2: Results of classifiers trained on gold-standard and silver annotation data

Determining the frequency threshold value for including features when training a probabilistic classifier, i.e. the number of times a feature must be encountered so as to be considered, is particularly difficult, especially in presence of data sparsity. This frequency threshold value controls the compromise between precision and recall. For our experiments we set this threshold at 5. Experimenting with lower values detrimentally affects precision. As discussed, a number of false-negatives were missed due to their morphology which is different than the usual morphology of drugs. These two facts, suggest that probably some informative features did not qualify due to the frequency threshold value.

3.4.3 Evolving string-similarity patterns

In this section we pursue improving recall by learning string similarity patterns based on dictionary knowledge. Exploring ways to restore the predictive power the model could have if more training data were available, we develop a mechanism to deal with these easy, yet elusive false-negative cases, discussed in the previous section. We attempt to genetically evolve string patterns that can then be used as regular expressions to capture drug names that are not present in the dictionary.

Following the work of Tsuruoka *et al.* [193], we also use a form of regression in order to learn common string patterns of drug names. More specifically, we used GP, also known as “symbolic regression”, a technique deeply rooted in the basic principles of Darwinian evolution which allows the evolution of programs at the symbolic level [107]. GP is used in this task as a global optimisation algorithm. The pseudo-random sampling inherent in GP means that no hard guarantees about the final outcome can be made. However, the randomness also enables a good coverage of the fitness landscape

[1]s $\rightarrow$ *silver*

and therefore avoids falling into local optima, which is essential to solve our problem. Furthermore, the self-driven nature of evolution is robust as it makes little to no assumptions about the fitness landscape, thus mitigating any bias during the learning stage, which enables it to produce meaningful solutions where other global optimisation algorithms can falter [107]. The whole notion of evolution, as put forward by Charles Darwin, is based on sexual reproduction (crossover), random mutation and natural selection, hence, learning by means of evolution is a good fit for our use-case as it offers the attractive possibility of finding decent solutions with no prior knowledge. We only need to define a fitness function (a measure for judging how fit a candidate is), a function-set (the functions available to the GP system) and a terminal-set (the constants available to the GP system). The functions and terminals are essentially, the primitives with which a program in GP is built. Loosely speaking, terminals provide a value to the system and functions process a value already in the system. Typically, a GP system that deals with numerical calculations will need, at least, the four basic arithmetic operations as function-set (+, -, *, /) and the numbers [0-9] for terminal-set ('0' can in fact be quite dangerous, as it has the power to cancel out entire branches of the tree, but that is outside the scope of this paper). In our case, since we are dealing with strings, it makes sense to use the characters [a-z] for terminals and several string-manipulating functions (e.g. *"split"*, *"join"* ,*"concat"*) for our function set. In addition, we included some terminals (characters) that are needed in order to build meaningful regular expressions, such as: | \ * + *? ( ) [ ].*

Each 'organism' in the genetic population is a little program. When executed, the program produces a string which is assigned a score according to the fitness function. For this purpose, all the single-word terms were extracted from DrugBank and were used as 'test-data' within the fitness function, which simply returns the proportion of matches as a measure of fitness. In case the string produced is not even a valid regular-expression, the candidate receives negative score and will most likely be disregarded in the next generation. For instance, a candidate who matches 50/6,700 terms in DrugBank is obviously fitter than some other one who matched only 10/6,700 terms, who in turn, is fitter than someone whose string could not even be compiled. Unfortunately, GP did not achieve anything less than 100% error when trying to match entire tokens, and so we limited the testing scope to the last 4, 5 or 6 characters of each token. This decision was made after observing that word-endings tend to be more similar than word-beginnings in drug names, mainly for conformance with the United States

Adopted Name (USAN) stem grouping [1], but possibly for marketing reasons as well. This had a major positive effect on the population in most executions, which confirmed our suspicions.

After 200 experiments with 80 generations per experiment and 10,000 individuals per generation, the 30 best evolved individuals were selected. Each individual is a function which builds a string that represents a potentially common suffix, in the form of a regex pattern. The pattern produced by the best individual managed to match 130 terms (7.3%) in the test-set! At first glance, this looks impressive, however, it is worth remembering that the evolutionary process performed evaluation using a list of singletons and not actual sentences. As a consequence, these patterns will most likely introduce false-positives if applied directly on real text, thus, decreasing precision. As a final step, we aim to keep the best of the best patterns only, i.e. the least likely to introduce false-positives. Thankfully, this can be done in various ways. Since the number of patterns is small, the cost of manual checking by a domain expert is limited. Even non-experts could probably accomplish this checking task. Or the problem can be tackled algorithmically. In fact, the latter is how we approached it. We calculated all possible combinations of sets of 5 string patterns and fired a long evaluation process where each combination was evaluated only for false-positives on 10 randomly selected paragraphs from the original training set (PK corpus). After two days of evaluations we acquired 5 sets of patterns (25 in total) which achieved the least false-positives. These 25 patterns were reduced to 11 unique ones after removing duplicates and those that would clearly introduce false-positives, the most representative of those being: *"m?ine"* (fluvoxamine vs examine). Finally, we augmented these 11 patterns by wrapping them with the following string:

```
\b+(\d?\,?\d'?\-?)?\w+<pattern>+\b
```

The strings `\b+` and `+\b` at the start and end of the pattern respectively, fire only when the token matches the GP-pattern and is a whole word, whereas, the string `(\d?\,?\d'?\-?)?\w+` specifies optional triggers mainly for matching hydroxylated compounds (e.g. if it matches 'midazolam', it should also match '4-hydroxymidazolam', '4,5-hydroxymidazolam' and '4,5'-hydroxymidazolam').

Manually augmenting the patterns returned from the GP training as described above, can be considered a heuristic, albeit a rather minimalistic one. It is common knowledge in biochemistry that all organic compounds go through a process called *oxidative*

[1]http://www.ama-assn.org/ama/pub/physician-resources/medical-science/united-states-adopted-names-council/generic-drug-naming-explained.page?

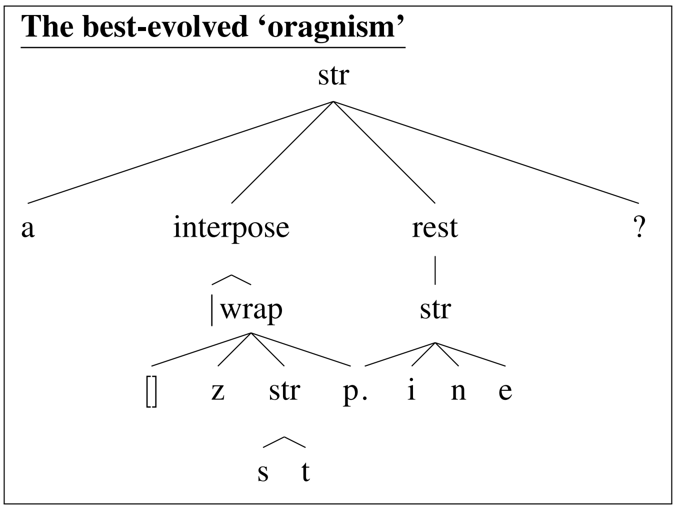
Evolved patterns	**Augmented patterns**	**Matches**	**Example**
a(z|st|p)ine?	\b+(\d?\,?\d’?\-?)?\w+a(z|st|p)ine?+\b	130	nevirapine
(i|u)dine?	\b+(\d?\,?\d’?\-?)?\w+(i|u)dine?+\b	72	lepirudin
azo(l|n)e?	\b+(\d?\,?\d’?\-?)?\w+azo(l|n)e? +\b	62	fluconazole
tamine?	\b+(\d?\,?\d’?\-?)?\w+tamine?+\b	44	dobutamine
zepam	\b+(\d?\,?\d’?\-?)?\w+zepam+\b	17	bromazepam
zolam	\b+(\d?\,?\d’?\-?)?\w+zolam+\b	13	haloxazolam
(y|u)lline?	\b+(\d?\,?\d’?\-?)?\w+(y|u)lline?+\b	12	enprofylline
artane?	\b+(\d?\,?\d’?\-?)?\w+artane?+\b	11	eprosartan
retine?	\b+(\d?\,?\d’?\-?)?\w+retine?+\b	10	hesperetin
navir	\b+(\d?\,?\d’?\-?)?\w+navir+\b	9	saquinavir
ocaine	\b+(\d?\,?\d’?\-?)?\w+ocaine+\b	9	benzocaine

Table 3.3: Evolved and augmented patterns

degradation when they come in contact with air. Hydroxylation is the first step in that process and converts lipophilic compounds into water-soluble (hydrophilic) products that are more readily excreted. We observe many mentions of such compounds in pharmacology papers, and therefore we attempt to capture them with this simple regex.

The GP paradigm parallels nature in that it is a never-ending process. Theoretically speaking, GP can produce ‘organisms’ of arbitrary complexity with minimal human intervention. In practise however, and particularly when evolving code, arbitrary complexity is rarely desired because it is very easy for the model to over-fit, start deviating substantially from a good solution approximation, or simply become unreadable. Two simple and widely used techniques of addressing this are a) stop the evolution process when a number of iterations (generations) has been reached and b) control complexity by limiting how deep the tree can get. We did both. Organisms were not allowed to grow above a certain depth (10) unless they achieve extraordinary performance ($>$ 35%) which did not occur.

The patterns were evolved assuming that they will be tested on single word terms. This was achieved by simply not including the space character in the terminal set thus, making it impossible for a pattern to include it.



3.4.4 Evaluation of evolved patterns

We evaluated the 'fittest' of patterns as a separate classification model which, however, enjoys privileges similar to the dictionary during aggregation. This means that positive predictions of the pattern model are assigned a probability of 100%. Table 3.4 shows the results obtained. As a first observation, classifier ensembles trained both on gold-standard and silver annotation data do not perform better than classifier ensembles trained on gold-standard data, only. Combining the dictionary and the pattern model compensates for the lack of a lower-quality model both for the MaxEnt and the perceptron classifier. Comparing tables 3.1, 3.2 and 3.4 demonstrates how we gradually moved from the recall range 84% - 88% to 89% - 93%, while keeping precision above 96% - 97%. In fact, since there are some verified annotation inconsistencies in the test corpus, reported precision is slightly lower than the precision on an consistent test-set. More specifically, some terms, such as *3-hydroxyquinidine*, *cycloguanil* and *4-hydroxyomeprazole*, have not been appropriately tagged as drugs.

Classifier	**P**	**R**	**F_1**
MaxEnt(g)+Dictionary+Patterns	97.3%	93%	95.1%
MaxEnt(g+s)+Dictionary+Patterns	97.3%	93%	95.1%
Perceptron(g)+Dictionary+Patterns	95.8%	88.9%	92.3%
Perceptron(g+s)+Dictionary+Patterns	96%	88.8%	92.3%

Table 3.4: Evaluation results of ensembles that contain the pattern classifier

Classifier	**P**	**R**	**F$_1$**
MaxEnt(s)+Dictionary+Patterns	97.4%	85.4%	91.0%
Perceptron(s)+Dictionary+Patterns	97.3%	85.1%	90.8%

Table 3.5: Results of classifiers that did not use gold-standard data

3.4.5 Ignoring gold-standard data

In our experiments so far, we assumed that at least some gold-standard data is available for training. However this might not always be the case. In this section, we are concerned with *"How much worse would results be, in the absence of a gold-standard training set?"* This is an important question because, as discussed earlier, gold-standard annotations are time consuming and costly. Ignoring expensive annotations, we experiment with classifiers trained on the easy-to-produce automatically generated annotations, the dictionary and the pattern model. The same gold-standard corpus was used for testing. The results obtained are shown in Table 3.5.

Comparing these results with the ones from our baseline experiments, presented in Table 3.1, shows that the MaxEnt classifier trained solely on silver annotation data, combined with the dictionary and the pattern model achieves similar performance to the MaxEnt classifier trained on gold-standard data. This result is encouraging, since it suggests that access to gold-standard data is not always a requirement for high performance drug-NER.

3.5 Discussion

Using a lexical database to annotate named entities in raw text is not a new concept. In fact, since lexical databases are manually annotated, annotating sentences for named entities from scratch certainly contains some level of effort duplication. We attempted to automate the annotation process by utilising such resources. Unfortunately, our results show that using the dictionary as a direct annotator of NEs achieves top precision but limited recall. Classifiers trained on gold-standard annotations achieved comparable precision but much higher recall.

For these reasons, we attempted to experiment with methods to pre-process DrugBank before using it as a NE annotator. To increase recall we generalised dictionary entries into regular expression patterns. We were expecting that the patterns would be able to capture named entities that were not listed in the dictionary but share common morphological characteristics, such as suffixes or prefixes, with some dictionary entries.

Obtaining such patterns automatically, that are accurate, is hard. We feel that our list of patterns is neither perfect nor complete. Perhaps a pharmacologist cooperating with a regular-expression expert can find more or better ones (more general) in a fraction of the time it took us. Even though this sounds tempting, we are trying to move as far away as possible from spoon-feeding knowledge humans already possess into our models. At this point, we leave that path to those not committed to such experimental restrictions to explore. In the future, and perhaps as part of a practical application, it would be very interesting to compare our automated method with expert-driven regular expressions.

Throughout the whole experiment, we relied heavily on our prediction-aggregation algorithm. This, again, is not perfect. It makes several assumptions about the world that may not hold in a different context. Moreover, as the algorithm gives a strong emphasis on the deterministic models, the precision of each such model had to be ensured. Precision dropped from 100% on the test-set and the 20 random paragraphs from the training-set, to 98% on the testing-set, mainly due to the annotation inconsistencies mentioned in Sections 3.4.1 and 3.4.4, meaning the only true false-positives were a couple of mentions of *"pyridine"*. Despite being far from perfect, this algorithm lies at the heart of this study and enabled us to de-construct the dictionary in pieces. From the dictionary we managed to extract common word-ending patterns. The dictionary was also used as an annotator for an entire corpus. It was the dictionary that gave us the synonyms which, despite having contributed very little, are generally very useful information to have. This deconstruction was entirely facilitated by this algorithm that was able to use the pieces as separate models. These types of "voting-systems" are becoming increasingly popular mainly for increased performance but also for overall stability of the resulting classifier [33, 20, 215, 35, 177].

It also has to be noted that both sets of gold-standard data are of roughly the same size. Contrasting this with other similar NER experiments, we find that the testing-set is usually a lot smaller than the training-set regardless of the evaluation scheme (holdout or cross-validation). This is due to the fact that the problem of data-sparsity is pervasive across the entire text-mining & NLP discipline (with regards to probabilistic training). In practice, this means that there is rarely enough training data, thus splitting it in two equally-sized pieces will most likely not lead to satisfying statistics. We decided from the start to leave the data as is, in order for the experiments to be as easily reproducible as possible.

Our results demonstrate that, even though availability of gold-standard data is certainly helpful, it is not a strict requirement with regards to drug-NER. Drugs often share several morphological characteristics which reduces the contextual information that is needed in order to make informed predictions. Nonetheless, it remains to be seen whether our combination of heterogeneous models will hold on to its performance when tested against a larger corpus.

3.6 Conclusions & Future work

This study mainly focuses on achieving high performance drug-NER with very limited or no manual annotations. We achieved this by merging predictions from several heterogeneous models including, models trained on gold-data, models trained on silver-data, DrugBank and finally, the evolved regex patterns. We have shown that, state-of-the-art performance in drug-NER is within reach, even in presence of data sparsity. Our experiments also show that combining heterogeneous models can achieve similar or comparable classification performance with that of our best performing model trained on gold-standard annotations. We have shown that in the pharmacology domain, static knowledge resources such as dictionaries actually contain more information than is immediately apparent and therefore can be utilised in other, non-static contexts (i.e. to devise high-precision regex patterns). Including synonyms in the dictionary or disambiguating acronyms did not improve results in this study mainly due to certain design decisions that surround the PK corpus. More specifically, none of the tagged acronyms were identified by AcroMine, whereas most of the identified synonyms have simply not been tagged appropriately in the test-set. Generally speaking however, we would expect a significant performance boost from applying these methods.

We plan to extend this work in the future. First of all, we plan to take advantage of all the annotations in the PK corpus. Having a good story for both drugs but also, drug-targets is essential for the task of identifying relationships and interactions between the two. We are also very interested to see if we can improve on, or find more of such accurate regex-patterns in order to enrich our 'safety-net' model. Finally, we feel that our prediction-aggregation algorithm gives too much emphasis on the deterministic models (dictionary or regex). Revisiting the algorithm is inevitable. We would like a somewhat more sophisticated policy for assigning probability to the predictions originating from the static models. Assigning 100% probability, as we currently do, will, most likely, not be acceptable in other domains.

Despite the aforementioned limitations and future plans, the models and algorithms produced have been shown to be robust and are thus, ready to contribute towards the greater goal of recognising PK parameters. How these models will be integrated in that future solution is unclear at this point.

Chapter 4

Extraction of PK interactions

4.1 Introduction

As advances into disease pathology and molecular function continue to generate ever growing amounts of data describing small molecules' interactions, there exists an important need to capture and store these relationships in structured formats, in order to enable sophisticated computational analysis. Even though there do exist efforts that create repositories of such information in computer-friendly form, populating these sources is typically a slow and expensive process, as domain experts must manually interpret and extract interaction relationships from a large body of the current literature. Automating the extraction of interactions from unstructured text, would significantly augment the content of these repositories and provide means for managing the exponential growth of the literature.

As discussed in Chapters 1 and 2, PK/PD modelling forms, in no small part, the basis for modern drug research and development. Using models, meaningful PK parameters may be defined which can be used to find relationships between the drug kinetic profile and the physiological process which drives its absorption, distribution and elimination. For instance, compartmental models allow us to easily define the clearance which depends on the drug elimination process, or the volume of distribution which depends on the drug distribution in the tissues. Models provide also an easy way to get an estimate of drug absorption after extra-vascular administration (bio-availability).

In this chapter, we describe a system for extracting PK interactions from unstructured text. By constructing a context free grammar (CFG), within which, we embed a lexical analyser, we show that efficient parsers can be constructed for extracting these, highly domain-specific, relationships and interactions from free text, with high rates of accuracy. Contrary to other published, usually statistical, techniques, the choice of a CFG takes away many of the complexities of NLP, by focusing on domain specific structure as opposed to analysing the semantics of a given language [186]. Additionally, by embedding a lexical analyser which can arbitrarily transform or tag tokens, we provide a level of abstraction for adding new rules, possibly for extracting other types of biological relationships beyond PK interactions.

4.2 Background

Various techniques for recognising relevant names have been proposed and discussed in Section 2.3.1. For instance, the use of standardised dictionaries containing the names and synonyms of relevant entities could be a simple but effective way for recognising these entities in free form text. However, this technique remains limited as certain names, or variations thereof, not present in the dictionaries produce large amounts of false negatives. Some researchers have addressed the issue of many false negatives by performing approximate string matching rather than strict matching, by using unique database identifiers mapped to a set of names, by using hand-crafted 'templates' and of course, by using probabilistic ML algorithms.

Similarly to the problem of NER, there has been a range of varying techniques published for extracting relationships from scientific literature. These are discussed in Section 2.3.2. Again, in the last decade, techniques which revolve around ML have dominated the research space with very few exceptions.

Despite being the trend of the decade in many disciplines, supervised ML models and their overall performance is typically depicted by the quality and size of the data sets used to train them. We showed in the previous chapter that, at least in NER and in some cases, lower-quality data can be generated quickly, and that practically gold performance can be achieved by combining several less powerful models. The methods we presented should not be interpreted as overcoming the need for data but rather as overcoming the need for large amounts of gold data. If anything, the entirety of that study is a tribute to how important training data and knowledge repositories are.

4.3 System & Methods

4.3.1 Overview

In this section, we describe an alternative method for extracting interactions from natural language, which despite not relying on ML, achieves high rates of accuracy by embedding a lexical analyser in a CFG. The CFG itself is designed specifically for parsing PK DDIs (pharmacokinetic drug-drug interactions) and was constructed by examining a small number (20) of artificial sentences such as the ones listed next:

- $DRUG_0$ increases the PK_0 of $DRUG_1$ 2-fold.
- Co-administration of $DRUG_0$ with $DRUG_1$ decreased the PK_0 by 50%.
- PK_0 of $DRUG_0$ is MOD_0 reduced when co-administered with $DRUG_1$ $DOSE_1$.

We aim to show that CFGs can be viewed as an easily extensible platform for extracting interactions and are powerful enough to describe most free language structure, while restricted enough to facilitate efficient parsing. Our method for extracting PK interactions from unstructured texts can be decoupled into three separate parts:

1. a set of NER models responsible for recognising drug and PK-property names
2. a set of functions (lexical analyser) that rely on the aforementioned models, responsible for tokenising and tagging relevant terms
3. a parser constructed around a CFG responsible for interpreting the collection of tokens and output parse-trees based on the rules of the grammar

The system was designed and built using the Clojure[1] programming language, and utilised the InstaParse[2] compiler to generate the parser. Clojure, a language hosted on the JVM, was chosen because of its interoperability potential with other JVM languages, but also because of its functional underpinnings.

4.3.2 NER Models

For an in-depth description of the NER models used for recognising drug-names, refer to Chapter 3. Recognising PK properties is a much simpler task, as the terms describing them are well defined and limited in number (10-12). Therefore, the problem can be easily tackled with a minimal set of regular-expressions.

[1] `http://clojure.org/`

[2] `https://github.com/Engelberg/instaparse`

4.3.3 Lexical Analyser

The lexical analyser is a component designed to accept distinct sentences as input. It then uses a pluggable tokeniser, to parse the sentence and produce a stream of tokens. In addition, it injects certain invisible tags in the grammar. These tags refer to specific predicates previously defined. Predicates are single argument functions that return a boolean value (true/false). This indirection allows the CFG to delegate to external functions at runtime, which in turn allows for the CFG to embed a mix of heterogeneous techniques (e.g. probabilistic for drug-NER, deterministic for PK-NER). Even though the lexical analyser and parser are separate component processes, they do communicate via these special invisible tags, enabling other external third-party tools to be easily integrated or swapped in.

4.3.4 Context-free Grammar & parser

The parser was developed using a set of grammar production rules allowing for the detection of PK interactions. As mentioned earlier, the production rules were derived by manually analysing a small number of artificial sentences. The use of CFGs for validating structure in natural language was first proposed almost 60 years ago by Chomsky [50], according to whom, a CFG for representing production rules has the following key components:

1. A set of tokens T, known as terminal symbols
2. A set of non-terminals N disjoint from T
3. A set of productions P of the form $a \rightarrow b$, where $a \in N$ and b is a sequence of one or more symbols from $N \cup T$
4. The start symbol S where $S \in N$.

Therefore, the language generated by a CFG can be enumerated by repeatedly applying production rules, starting with the start symbol S, and replacing non-terminals with their associated production rules until all non-terminals have been exhausted.

In order to extract PK interactions from unstructured text we developed the grammar illustrated in Figure 4.1 using EBNF (Extended Backus-Naur Form). Table 4.1 shows the injected tags, their corresponding functions and the basis for the current implementation. All the pre-injected tags appearing on the right side of the grammar

Analyser tags	**Corresponding functions**	**Current implementation**
NER_PK	PK recogniser	Deterministic (regex)
NER_DR	drug recogniser	Probabilistic (MaxEnt)
WORD	tokeniser	Probabilistic (Perceptron)
DIGIT	digit recogniser	Deterministic (regex)
NUMC	numbers expressed in alphabetic chars	Deterministic (regex)
CYPX	CYP recogniser	Deterministic (regex)
INT	integer recogniser	Deterministic (regex)
PUNCT	punctuation recogniser	Deterministic (regex)
PAREN	parentheses recogniser	Deterministic (regex)
ACTION	interaction recogniser	Deterministic (regex)
SQBR	square brackets recogniser	Deterministic (regex)
PREPO	preposition recogniser	Deterministic (regex)
NEGATOR	negation clue recogniser	Deterministic (regex)
ABBRX	abbreviation recogniser	Deterministic (regex)
BEX	verb 'to be' recogniser	Deterministic (regex)
VING	'...ing' ending verbs recogniser	Deterministic (regex)
ADVERB	'...ly' ending adverbs recogniser	Deterministic (regex)
REPEAT	'once', 'twice', 'thrice' recogniser	Deterministic (regex)
PERCENT	percent recogniser	Deterministic (regex)
ADMIN	administration recogniser	Deterministic (regex)

Table 4.1: Analyser tags and their associated external functionality

are surrounded with ">>" to denote their indirect nature. As one would expect, since these tags are 'invisible', they do not appear on the left side of the grammar.

Generally speaking, parsing methods fall into one of the two categories, *top-down* and *bottom-up*. In top-down approaches, also widely known as *recursive descent*, construction starts at the root node and progresses downstream towards the leaves, while in bottom-up approaches, construction starts at the leaves and progresses upstream towards the root. The parser generated by InstaParse is a top-down one which, however, allows the use of left recursion by utilising state-of-the-art techniques described by Frost et al. [69] (our grammar is not left-recursive but it is good to know that it can be).

4.3.5 Examples

In order to help the reader visualise and understand the parse tree generated by the parser, we provide 2 examples in 2 different visual formats. The following 2 representative sentences were used.

```
S  =  PHRASE+   END
PHRASE = (DDIPK / DDI / COADMIN / ENCLOSED ) | TOKEN ( TOKEN >>PUNCT>>?)*
DDIPK = OBJECT?  PK  (BE | TO)?  OBJECT?  EFF?
DDI =  PRECIPITANT  MECH  OBJECT  PK | PRECIPITANT  MECH  'in'
EFF =  MAYBE?  BE?  (SIGN | FOLD)?  MECH  (ADV | FOLD)?  (PRECIPITANT | PK)?
TOKEN = ((PRECIPITANT / OBJECT / DRUG) | DOSE | ROUTE | NUM  |  PK | PERCENTAGE
         XFOLD | XFACTOR | CYP | ABBR |  MECH | SIGN | GROUP | TO | ENCLOSED )
         / >>WORD>>
INC-DEC = ('increase' | 'decrease') / >>ACTION>>
FOLD =  (NUM  '-'  ('and' | ',')? )+  'fold'
COADMIN = >>ADMIN>>  PRECIPITANT  | >>ADMIN>> TO?  DRUG  (PRECIPITANT | 'and')
GROUP = 'placebo' | ('healthy' | 'normal')  'adult'?  ('male' | 'female')?
        ('volunteers' | 'subjects')
XFOLD = FOLD  EFF?
XFACTOR = BY  'factors'  (TO | >>VING>>?  'between')  (NUM ('and'  NUM)*)
ROUTE =  >>ROUTEX>>
BY = 'by'
UNIT = 'mg' | 'g' | ('microgram' 's'?) | 'mcg'
VOLUME = 'ml' | 'mL'
DOSE = (NUM  '-'?  NUM?)  UNIT  INTERVAL?
ABBR = >>ABBRX>>
INTERVAL = >>REPEAT>>  (ADV | 'a'  TIME) | (NUM | >>NUMC>>)  'times'  ('per'  T
           ('/' | 'a')  (TIME | VOLUME) | ABBR
TIME =  'hour' | 'day' | 'week'
PERCENTAGE = NUM  ('%' | >>PERCENT>>)
ENCLOSED = >>PAREN>> | >>SQBR>>
NUM =  >>DIGIT>>
CYP =  >>CYPX>>
ADV =  >>ADVERB>>
PRECIPITANT = (BY | 'with' | 'as')  DRUG / ('presence' | #'dos(es?|ing)' |
              'addition' | >>ADMIN>>  'of'  DRUG / DRUG  ('treatment' |
              'therapy' | 'administration' | 'causes' | EFF) / 'when'  DRUG
PARENDOSE = '(' DOSE  ('for'  NUM TIME 's')? ')'
DRUGDOSE = PRECIPITANT  PARENDOSE
2DRUG = DRUGDOSE  'administered'?  'with' 'a single'?  DRUGDOSE
OBJECT = TO  'either'?  DRUG '\\'s'? / DRUG  PK  BE / DRUG  'metabolism'
DRUG = ROUTE?  >>NER_DR>>   'hydrochloride'? ('up to'?   DOSE)?
PK =  MECH?  'the'?  'mean'?  (OBJECT | DRUG)? >>NER_PK>>  OBJECT?
MECH =  MAYBE?  BE? >>ACTION>> ((FOLD | (BY?  ((NUM 'x') | PERCENTAGE) | OBJECT
SIGN =  ADV | NEG
MAYBE = 'can' | 'may'
NEG = >>NEGATOR>>
BE = >>BEX>>
END =  '.'
```

Figure 4.1: The CFG production rules

1. *The pharmacokinetics of oral conivaptan (20 - 40 mg/day) were unchanged with coadministration of either captopril 25 mg or furosemide up to 80 mg/day.*
2. *Exposure to didanosine is significantly increased when coadministered with tenofovir disoproxil fumarate [Table 5 and see Clinical Pharmacokinetics (12.3, Tables 9 and 10)].*

Figure 4.2 shows the raw nested data-structure, whereas Figures 4.4 and 4.3 depict the trees of sentence 1 & 2 respectively, in a more traditional way.

4.4 Evaluation scheme

In order to test our approach, we used the *PK DDI* corpus[1]. This is a new corpus of sections from FDA-approved drug package inserts (PIs) that have been manually annotated for PK drug-drug interactions by a pharmacist and a drug information expert. The two annotators reached consensus on 592 PK DDIs, 3,351 active ingredient mentions, 234 drug product mentions, and 201 metabolite mentions present in over 200 PI sections extracted from 64 drug PIs. The corpus is provided to the non-profit research community under a Creative Commons licence.

Not every single sentence in this corpus is PK relevant. In fact, the majority of them are not. There would practically be no benefit from deploying our system on irrelevant sentences, so we manually reduced the corpus into a set of 342 'gold' sentences, which do include PK interactions, and therefore are relevant to this study. Once the relevant sentences were collected, the parser derived from the CFG was deployed on them. The 'gold' test-set distributed with the aforementioned corpus, comes in a .csv format with the following columns per interaction:*FileName, Precipitant Type, Precipitant, Precipitant Annotator, Precipitant Span Start-Offset, Precipitant Span End-Offset, Object Type, Object, Object Annotator, Metabolite active ingredient, Object Span Start-Offset, Object Span End-Offset, Modality, Interaction phrase-type, Interaction Phrase, Interaction Phrase Span Start-Offset and Interaction Phrase Span End-Offset.* A subset of these columns, namely the *Precipitant, Object, Modality and Interaction phrase* can directly be extracted from the resulting parse-tree, whereas the *Interaction Phase Type* (e.g. qualitative vs quantitative) can be inferred by examining the presence, or absence, of tags such as $XFOLD$, $XFACTOR$ and $PERCENTAGE$. All the rest of the

[1]`http://dbmi-icode-01.dbmi.pitt.edu/dikb-evidence/package-insert-DDI-NLP-corpus.html`

```
[:S
 [:PHRASE
  [:TOKEN "The"]
  [:TOKEN
   [:PK
    "pharmacokinetics"
    [:OBJECT "of"
      [:DRUG [:ROUTE "oral"]
         "conivaptan"]]]]
  [:TOKEN
   [:ENCLOSED
    "("
    [:DOSE
     [:NUM "20"]
     "-"
     [:NUM "40"]
     [:UNIT "mg"]
     [:INTERVAL "/"
       [:TIME "day"]]]
    ")"]]
  [:TOKEN "were"]
  [:TOKEN [:SIGN [:NEG "unchanged"]]]
  [:TOKEN "with"]
  [:TOKEN "coadministration"]
  [:TOKEN [:OBJECT "of" "either"
          [:DRUG "captopril"]]]]
 [:PHRASE [:TOKEN
          [:DOSE [:NUM "25"]
            [:UNIT "mg"]]]]
 [:PHRASE
  [:TOKEN "or"]
  [:TOKEN
   [:DRUG
    "furosemide"
    "up to"
    [:DOSE [:NUM "80"] [:UNIT "mg"]
      [:INTERVAL "/" [:TIME "day"]]]]]]
 [:END "."]]
----------------------------
[:S
 [:PHRASE
  [:DDIPK
   [:PK "Exposure"
     [:OBJECT "to"
       [:DRUG [:ROUTE "oral"]
         "didanosine"]]]
   [:EFF
    [:BE "is"]
    [:SIGN [:ADV "significantly"]]
    [:MECH "increased"]]]]
 [:PHRASE
  [:COADMIN
   "when"
   "coadministered"
   [:PRECIPITANT "with"
     [:DRUG "tenofovir"]]]]
 [:PHRASE
  [:TOKEN "disoproxil"]
  [:TOKEN "fumarate"]
  [:TOKEN
   [:ENCLOSED
    "[Table 5 and see
      Clinical Pharmacokinetics
     (12.3, Tables 9 and 10)]"]]]
 [:END "."]]
```

Figure 4.2: Example parse-trees in raw nested format

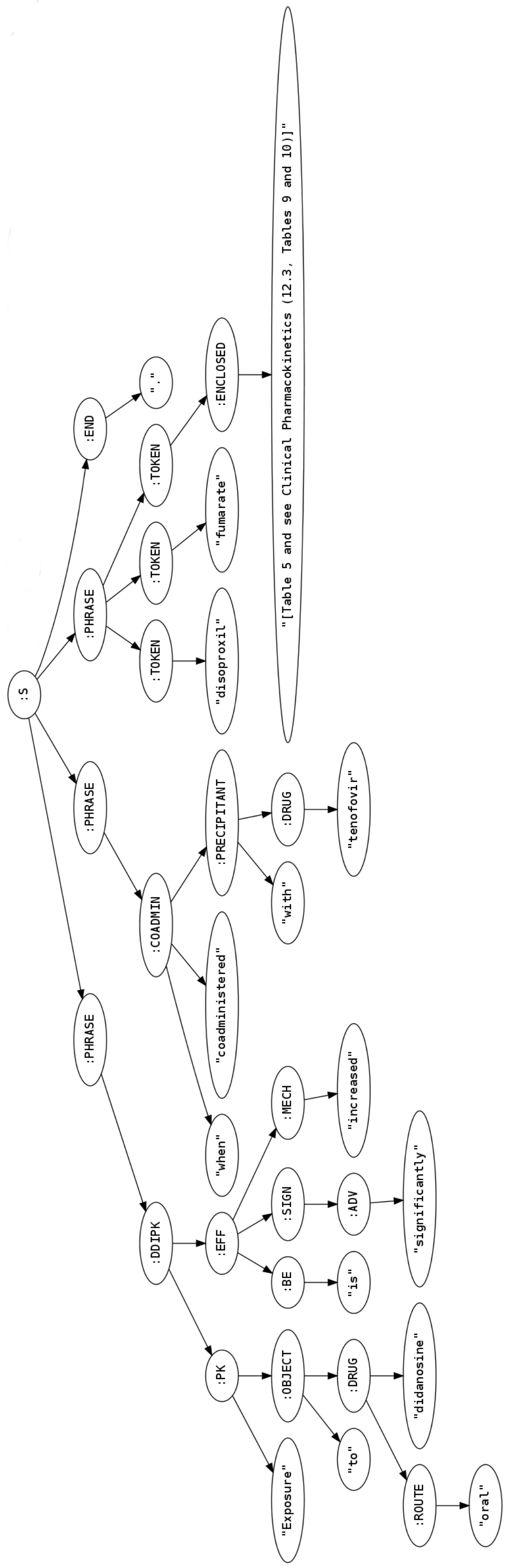


Figure 4.3: Parse-tree of example sentence 2 in tree format

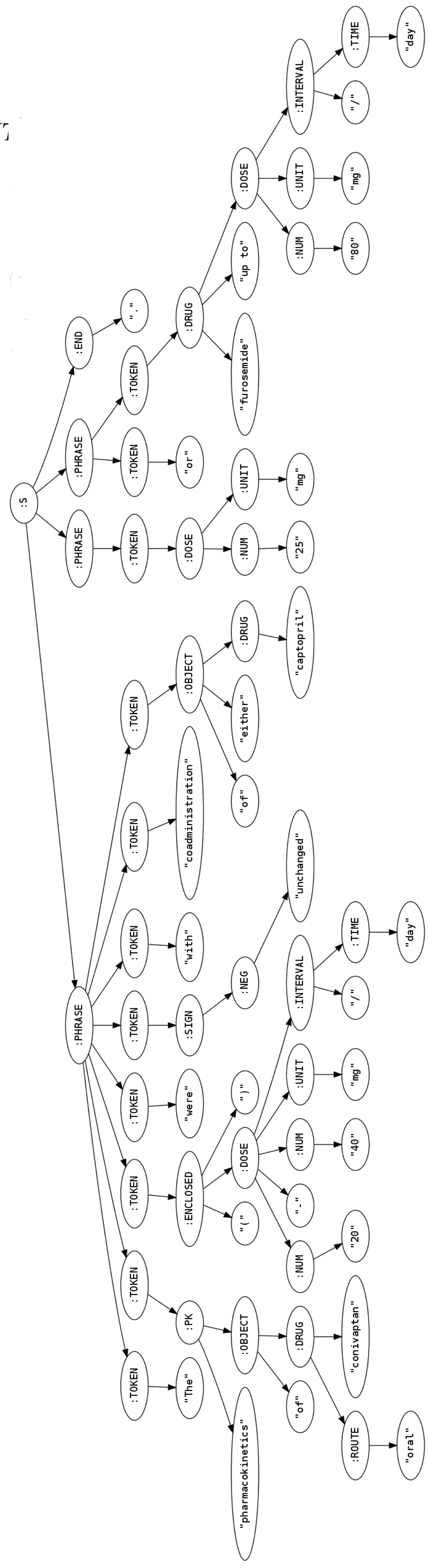


Figure 4.4: Parse-tree of example sentence 1 in tree format

columns were disregarded as they relate more to the annotation process, rather than the interactions themselves.

4.5 Results and Discussion

We measure performance using the standard IR evaluation metrics, Precision (P), Recall (R) & F-Score (F_1). Analysis of the output generated by the system demonstrated perfect recall and precision rates for recognising drug names (100%). Such NER performance was expected, as the drugs mentioned in this particular corpus are all FDA-approved compounds, which our NER models have no trouble recognising. With respect to recognising PK interactions, we calculated P = 88.9% and R = 93.1%, which gives an F-score of 90.9%.

Even though such results cannot be considered state-of-the-art, they show that the system can perform accurate extraction of PK interaction data when the lexical analyser and parser encounter sentences that match the specified grammar. This includes complex sentences such as the one depicted in Figure 4.3, where the grammar correctly identified the observation that tenofovir can positively affect the exposure to didanosine. Moreover, the grammar is tuned to identify common negation clues, which in turn means it can accurately extract observations that dispute potential interactions, such as those depicted in Figure 4.4, where it was correctly extracted that neither captopril nor furosemide (at specific doses) can affect the pharmacokinetics of conivaptan.

For the cases in which the system generated false positives and/or negatives, the root cause was typically due to one of the following reasons:

1. The grammar confused the precipitant for the object (or vice versa)
2. The grammar identified no precipitant and no object

Additionally, our system has further limitations regarding the parsing of very long and complex sentences that often convey several distinct ideas. Given that the grammar rules were constructed by a non-expert and from a small number of artificial examples mimicking medical language, there are bound to be cases where the text will match either too many (false positives) or too little (false negatives) grammar rules. An example sentence that exhibits structural patterns not modelled by the grammar, and which demonstrates the problem is the following:

"In a study comparing the disposition of intravenously administered diazepam before and after 21 days of dosing with either sertraline (50 to 200 mg/day escalating

dose) or placebo, there was a 32% decrease relative to baseline in diazepam clearance for the sertraline group compared to a 19% decrease relative to baseline for the placebo group (p < 0.03).”

Unfortunately, the grammar cannot model notions such as *relative to baseline* and *before and after 21 days of dosing*. Since this can be mainly attributed to the medical inexperience of the author of the rules, it could potentially be addressed by enriching the grammar rules at a later stage. Nonetheless however, examples like the above do demonstrate how various unstructured text representations can negatively impact overall performance. We note that in general, the incidence of false entries we re minimal as indicated by the the high level of precision and recall rates achieved.

We are currently unaware of any other studies targeting PK interactions which utilised the PK-DDI corpus, and therefore direct comparison of this work with the work of others is not feasible. Generally speaking, the use of rule-based methods has been more or less abandoned by the community, which is now in favour of probabilistic approaches. Interestingly enough, our hybrid approach performed quite efficiently and effectively, while making use of limited computing resources and no training corpus. We therefore conclude that our reported recall and precision rates for extracting PK interactions from free text shows the potential to be able to mine significantly larger bodies of biomedical literature in order to populate structured representations for capturing interaction data for further computational analysis.

4.6 Conclusion

Concluding, we have demonstrated that the problem of recognising PK interactions can be tackled by using a CFG to recognise domain-specific patterns used to describe them. We have shown that the use of an embedded lexical analyser facilitates the use of external NER models and functions, which effectively reduce the problem of IE into one of pattern-matching, that can be solved efficiently by the CFG. This approach can be classified as simple and lightweight as it reduces a lot of the complexities associated with NLP. As a bonus, one can expect much better results, if the rules are somehow reviewed, or even better, constructed by a domain-expert.

Chapter 5

Software & Integration

5.1 Introduction

Nowadays, the TM and NLP community has a assortment of, not only methods and techniques, but also software tools, to choose from in order to conduct research. Research groups around the world are constantly developing standalone TM tools. However, despite the availability of numerous methods and techniques for various TM tasks, combining different tools requires substantial effort, time and expertise. In fact, it is a common secret amongst text-miners that often, the *pre-processing* time overhauls the actual *processing* time.

Typically, a tool is developed using a certain preferred data representation, programming conventions and preferences, as determined by the individual research group or organisation. In order to build complex text mining applications or pipelines, it is often required to combine multiple tools, possibly designed by different groups. The current practice of independent and disconnected tool development poses a hindrance to tool interoperability and integration. In order to use a new tool or a new dataset, TM researchers spend a substantial amount of time developing algorithms for processing the new data format. This heterogeneity in data representations slows down the development of powerful applications, thus leading to inefficiencies in research and innovation.

As discussed in Chapter 2, there have been some efforts to promote interoperability among text analytics tools, namely the UIMA and GATE frameworks. Development of UIMA or GATE compliant solutions requires the entire tool to be (re)written into framework specific constructs. The complexities and steep learning curve associated with these frameworks keeps them from being broadly accepted as a development and data sharing standard [183]. In addition, none of these two frameworks offer any capabilities for dynamically converting existing code into UIMA/GATE compliant. Consider for instance, the NER component of openNLP. There are really only two ways of making it an UIMA component. The first way would be to simply amend the existing source code to obey UIMA's contract for components. Doing so has several drawbacks. First and foremost, since the original source file was modified, this requires recompiling the entire openNLP package. Secondly, we would be complecting openNLP and UIMA semantics within the same source file. Thirdly, every software developer understands that adding or removing code can, and probably will, introduce defects. Finally, obeying any of the contacts of UIMA, essentially means that a transitive dependency to UIMA is introduced as well. The other possibility would be to simply write a wrapper class, which obeys UIMA's contract but delegates to the actual openNLP NER component. This class could then be distributed separately and the user can decide when to use it and what dependencies to bring in. This is certainly preferable as it introduces none of the aforementioned problems. However, it does introduce a brand new one which is most evident in statically-typed languages like Java, and particularly in situations where there is an existing code base that needs to be made UIMA/GATE compliant. In such cases, introducing a wrapper usually leads to significant code re-factoring as the types involved are no longer the same. Depending on the size and complexity of the project this can have serious counter-productive consequences. The underlying reason for this limitation has been known to Computer scientists for decades as the "Expression Problem" [147] (also known as "the extensibility problem"). Several modern languages including Scala, Clojure and Groovy provide semantics that address this problem quite elegantly. Unfortunately, for mainstream, typically older, languages such as C++, Java and CSharp, one has to resort to highly academic research and consequently, to complex/unwieldy data types and rather unconventional programming patterns.

Another limitation of current tools and frameworks, is the fact that almost none of them offers any facilities for constructing, not necessarily serial, work-flows. By non-serial, we mean, potentially multiple branching or merging of component nodes. This is quite serious as most high-level TM components usually expect input from, and need to broadcast their result to, multiple other components. Take for instance two of the most popular chunkers, the ones found in openNLP and stanfordNLP. Both of them, in their original form (not their UIMA counterparts), expect input from the tokeniser and the POS-tagger. If we were to draw this, it would be a triangle with 2 nodes feeding input to the third. Unfortunately, none of the two aforementioned frameworks provide any facilities for specifying that topology declaratively. The user is left on his own, with the only option being to, manually and explicitly, express these relationships in the code. UIMA, even though it does support multiple branching/merging points by providing not only multiple views of annotations, but also, annotation destroyers, the configuration and wiring between these advanced components remains explicit. Alternatively, the problem can be sidestepped by incrementally building annotations and passing them all downstream in a single CAS object, in order to simulate graph semantics. This way, the UIMA equivalent of the chunkers mentioned earlier, can look into the annotations built so far and choose to utilise the ones from the tokeniser and POS-tagger. It should be obvious that the serial and incremental nature of this approach has unfortunate implications, not only when there are multiple components producing the same types of annotations, but also for parallelism and overall memory footprint, as the system is forced to hang on to data that is either not needed at all, or will be needed much later. In order to make this clearer, consider the following analogy:

Assume, for simplicity reasons, that you want to find the standard-deviation of a collection of numbers - let us call that *xs*. The approach taken by a typical UIMA workflow would be reflected in the following pseudo-code:

i. Create an empty result
ii. Count xs and put it result
iii. Retrieve ‘count’ from result, use it to calculate the mean and put it in the result
iv. Retrieve ‘count’ from result, use it to calculate the mean-square and put it in the result
v. Retrieve ‘mean’ and ‘mean-square’ from result, use them to calculate the variance and put it in the result
vi. Retrieve ‘variance’ from result and use it to calculate the standard-deviation and put it in the result
vii. Hand the entire result to the user

It should be easy to spot that data is unnecessarily held on to until the algorithm finishes. There is no conceptual reason why we could not have disregarded ‘count’ after step 4, or both ‘mean’ and ‘mean-square’ after step 5. Similarly, there is no reason why steps 3 and 4 could not have been calculated in parallel, as they depend on the same constant value. With sufficiently large input, the limitations of such an approach become evident. Of course, in some cases the entire result is needed and thus, there is no other option other than storing everything. Our simplistic analogy would reflect this case if we were asking for all the statistics, instead of just the standard-deviation.

It needs to be emphasised that Argo, does improve on this matter by incorporating a flexible mechanism for cloning a CAS at each branch in the work-flow. When multiple analysis engines merge into a single component, Argo merges individual CASes coming from the different branches into a single CAS that supports multiple views [157].

5.2 Overview

In this chapter, we present our efforts and contributions with respect to software tools aimed at helping TM researchers in achieving integration and interoperability. More specifically, three key areas were identified as *'lacking'*, within the software space:

1. Multiple format annotation of entities using a set of lexical resources. A library or framework is usually expected to provide support for an annotation tool, and not the other way around. For instance, it is up to the openNLP team to provide format support for *brat* [182], rather than the *brat* team to provide support for openNLP. This seems quite odd as it is brat which is the specialised annotation tool.

2. Flexible and dynamic generation of classes compliant with the various platforms and frameworks. Because these platforms were developed in a statically typed language, they all make the same assumption, which is that, a conforming class must be written before the program actually runs. An existing piece of code that does the task, cannot be 'converted' at runtime. With the advent of more and more dynamically typed and compiled languages on the JVM, this is becoming a bigger and bigger issue. Developers in such languages often end up writing large amounts of boilerplate code in order to have, even a trivial solution, retrofitted to a particular framework. The exact same symptom can be observed in other frameworks like *apache Hadoop* that have nothing to do with NLP. Much like UIMA and GATE however, the programming model offered is intrinsically linked with the programming language the system was developed in. In a sense that is to be expected. Nonetheless however, that model feels awkward, at best, to developers coming from functional dynamic languages like Python, Groovy or Clojure.

3. Declarative creation of graph-like work-flows, using a mix-and-match approach. Ideally, graph work-flows should be defined declaratively, with dependency resolution implicitly and efficiently handled by the system. The user should not be concerned with what order the components will run in or in fact, with any other internal execution details.

Our open-source contributions towards these three areas, and more, are presented and discussed next.

5.3 Contributions

5.3.1 PAnnotator

Introduction

PAnnotator stands for “parallel annotator”, and is a high performance, dictionary-based, annotator for TM and other NLP-related tasks. PAnnotator aims to address the first problem mentioned in the previous section, which is the fact that TM researchers often need to annotate the same set of documents using various different representations. A good example of this would be NER. Typically, an NER researcher will want to try out various models found in various libraries or frameworks which, as we explained earlier, are in complete disagreement about how to feed the data in. Therefore, in the absence of an automated tool that knows how to generate output ready to be consumed by these libraries, the time that is needed to move between them grows in linear relation with the number of them. PAnnotator solves this problem, at least in the NER space, by encoding all the various library expectations, thus freeing the human from peculiar technicalities.

Features

Most of PAnnotator’s features are oriented around performance and ease of use, given the fact that annotating large amounts of documents is a CPU-intensive job, and that not all NLP researchers are expert programmers. Currently, PAnnotator integrates with the NER modules of openNLP (Java), stanfordNLP-core (Java) & NLTK (Python). These are all very popular and actively maintained libraries.

- produces annotations compliant with either openNLP, stanfordNLP or NLTK

- supports a pluggable sentence splitter and tokeniser
- supports logging
- features 2 serial mapping strategies (*lazy* vs *eager*)
- features 4 parallel mapping strategies (*lazy-parallel, pool-parallel, fork-join & map-reduce*)
- can be accessed programmatically from any JVM-based language, or as a standalone Java application
- very efficient with good scaling characteristics
- low memory footprint

PAnnotator was relied upon heavily in order to generate training data for the NER work presented in Chapter 3. Apart from the author himself, there are another 12 confirmed (through correspondance) users of this tool.

Future work

Currently, PAnnotator integrates with the NER modules of openNLP (Java), stanfordNLP-core (Java) & NLTK (Python). Plans for the future include adding support for the NER module of GATE and replacing the CLI with a simpler graphical user interface (GUI) to make it even easier to use.

5.3.2 clojuima

Clojuima was born out the author's need to combine UIMA with the Clojure programming language, and essentially started out as mini Clojure wrapper for UIMA. However, it was quickly realised that there were greater hindrances to overcome before even considering wrapping. Strictly speaking, clojuima is not a software tool, but rather a tutorial and a sample project which demonstrate how to:

- Turn regular Clojure functions into UIMA components, thus leveraging UIMA's workflow and evaluation capabilities
- Use UIMA-components as regular functions in your Clojure code, thus leveraging the plethora of available components

Combining UIMA with Clojure presents quite a unique set of problems. Apart from highly stateful, UIMA relies heavily on the java.lang.reflect API for instantiating Classes and is tightly coupled to XML descriptors. Clojure, being a dynamic language, offers constructs that are able to generate classes that satisfy a particular abstraction, or even override a particular implementation from a parent class, at runtime. The need for predefined XML descriptors goes completely against the dynamic nature of Clojure, and essentially, imposes counter-intuitive work patterns. We strongly believe that given a function *foo* that performs some task, there should exist means of turning that into UIMA compliant, without having to modify or recompile existing code, nor being distracted by having to write large amounts of XML. Interestingly enough, this belief is shared by many other programmers/researchers, as indicated by the popularity of the *Apache uimaFIT* [144] library.

In a nutshell, *uimaFIT* provides Java annotations for describing UIMA components which can be used to directly describe these components in the code. This greatly simplifies refactoring a component definition (e.g. changing a configuration parameter name). It also makes it possible to generate XML descriptor files as part of the build cycle rather than being performed manually in parallel with code creation. uimaFIT also makes it easy to instantiate UIMA components without using XML descriptor files at all by providing a number of convenience factory methods which allow programmatic/dynamic instantiation of UIMA components. In order to give the reader a better understanding of the idiosyncrasies involved with creating and using bare UIMA components directly, it is worth mentioning that, even though uimaFIT is rather limited in terms of its functional aims (it only really aims at simplifying programmatic interaction with UIMA), the project still managed to reach a notable size (approx. 15,000 LOC), while introducing certain 'heavy-weight' dependencies (e.g. Spring) along the way. Nonetheless, the project is now more or less complete and therefore, Java developers can indeed enjoy a much simpler and XML-free programmatic experience with UIMA, via uimaFIT.

While uimaFIT does recover some of the dynamism abandoned by UIMA, by abstracting over XML descriptors, Clojure developers still face a major bottleneck, as the amount of wiring needed between the two, remains quite significant. This is in part due to certain limitations in Clojure's *'proxy'* construct, but also due to uimaFIT's heavy reliance on Java annotations, which are currently difficult to express in Clojure code (only supported in the 'deftype' construct and not in regular functions). For a more detailed overview of the technical issues, refer to the project's GitHub page[1].

Features

Being a tutorial, clojuima walks the reader through the following major points:

- the conflicting design principles between Clojure and UIMA
- why Clojure's built-in *proxy* and *:gen-class* constructs are insufficient
- a complete implementation of a *proxy*-like construct (UIMAProxy.java) tailored specifically for UIMA components, which sits on top of uimaFIT (written in pure Java and distributed with the sample project)
- real world examples of how to use the aforementioned construct
- clean separation of application logic from UIMA-specific code
- the beginnings of an idiomatic Clojure wrapper for UIMA

By the end of the tutorial, the reader should be able to to convert arbitrary Clojure functions into UIMA compliant components, by simply proxying them with the newly available UIMAProxy. The amount of extra code that is needed is minimal (two small functions) and is mainly concerned with writing the annotation indices into the Cas object (Cas-annotator) and retrieving them (Cas-extractor), for which, helper functions are provided. However, as dictated by the original aims of clojuima, the processing function is UIMA agnostic and needs to know nothing about the annotation function or the extractor function, which are UIMA-specific. This rationale allows for true separation of general-purpose functions from UIMA-specific ones, but also for abstracting over certain common UIMA operations. For instance, the following simple function can be used in the majority of cases, in order to extract the annotations from the Cas object:

```
(defn extractor [^JCas c]
  (.getDocumentText c))
```

Similarly, for tasks such as sentence-detection, tokenisation and NER, the following function can be used to write the annotation indices to the Cas object:

[1]https://github.com/jimpil/clojuima

```
(defn post-process
"Given a JCas, some annotation result and the original input,
 calculate the appropriate indices and inject them into the CAS."
[jc res original-input]
(inject-annotation! jc [Annotation 0 (count original-input)])
 (doseq [[_ [b e]] (calculate-indices original-input res)]
   (inject-annotation! jc [Annotation b e])) )
```

Clojuima addresses the second problem, as stated in Section 5.1, and has been warmly welcomed as a contribution by the uimaFIT team, appearing in the "Language Zoo" section of the project's wiki page[1].

5.3.3 hotel-nlp

Introduction

Hotel-nlp is by far, the most challenging and ambitious software endeavour the author has ever worked on, and it tries to address the third, and hardest, point as presented in Section 5.1. That is, being able to declaratively create work-flows that resemble graph topologies, with potentially multiple branching/merging points.

To our knowledge, there is only one other system offering similar capabilities, namely the Argo TM workbench [158], previously discussed in Section 2.1.4. In fact, Argo offers much more than that. In spite of being in beta stages of development, it is a rather mature web-based, text analytics platform. Argo builds on top of UIMA, as does U-Compare [90], also discussed in Section 2.1.4.

Being an entirely browser-based solution, Argo makes a significant trade-off, namely, accessibility over programmability. In other words, Argo cannot be accessed programmatically, and therefore cannot be considered an API. This may not be an issue for casual use, but users who require a somewhat finer control over their processes are, again, left on their own.

In addition, a key goal of hotel-nlp is to enable pluggable support for different underlying NLP library implementations. To fully understand this, consider the following real world analogous project, taken from a slightly different domain (statistics).

Within the Java ecosystem, there must exist more than 15 different matrix manipulation frameworks (e.g., jblas, jama, colt, jeigen, ujmp etc), completely foreign to each other. A user typically, picks one and sticks with it, at least, on a per project

[1]https://code.google.com/p/uimafit/wiki/Documentation?tm=6

basis. Unification of two or more of these libraries can only be done on client-side code and only through wrapping. We already saw that ultimately, this is a limitation of the programming language, as Java interfaces cannot be extended to concrete classes without modifying and recompiling them (*the expression problem*). Other languages however, do not exhibit this limitation. In such languages, abstractions can be cleanly extended to foreign concrete types without recompiling them. As a result, abstracting over all these libraries in a consistent manner could, and should, be lifted away from the end user, leaving him/her with an idiomatic API that allows for any library-specific data-type to be plugged in, without worrying about where it came from. Obviously, that API can then be used by any user of that programming language.

Having understood the above analogy, the reader is invited to visit the actual project[1] that addresses what was described in the above analogy. Despite living in different domains, *hotel-nlp* and *core.matrix* share many common goals and implementation strategies.

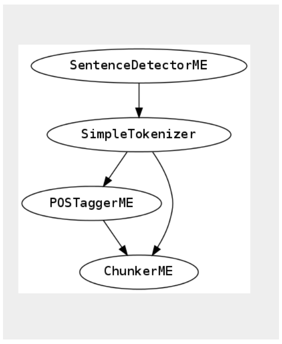
Overview

In this Section, we present the prototype for a complementary platform, which aims at simplifying interoperability and integration between foreign TM components. Even though this aim may sound very similar to UIMA's aim, hotel-nlp takes a radically different approach which enables, not only dynamic creation of compliant classes (via clojuima), but also, fully programmatic construction of graph work-flows. As a consequence, components can be selected using a mix-and-match approach, across multiple libraries or platforms, and work-flows can be built completely declaratively and dynamically. Contrary to UIMA which simulates graph semantics, a hotel-nlp work-flow is a true graph data-structure with minimal and implicit dependencies between the nodes. Revisiting the *standard-deviation* analogy expressed in Section 5.1, we present an alternative, more expressive and efficient way of specifying the exact same computation, below:

```
{:n    (fn [xs]   (count xs))
 :m    (fn [xs n] (/ (sum identity xs) n))
 :m2   (fn [xs n] (/ (sum #(* % %) xs) n))
 :v    (fn [m m2] (- m2 (* m m)))
 :std* (fn [v]    (sqrt v))}
```

[1]https://github.com/mikera/core.matrix

Comparing the two approaches, it becomes evident that the logic has now moved from a series of statements to a single data-structure, which internally encodes all the relationships needed. It does this by examining the function parameter names and matching them with the outer keys of the graph. For example, the function responsible for calculating the variance (:v), encodes dependencies to some m & m2 variables expected to be present as keys in the graph itself. The advantages of such a data-structure are many fold. Firstly, as far as the user is concerned, there is no notion of ordering as the correct order will be inferred when the graph is compiled. Secondly, each step depends only on what it absolutely needs and everything else quickly becomes eligible for garbage-collection. In the above case, since only *std* is starred, nothing else is needed by the time :v completes, hence memory can be reclaimed. In addition, nothing stops :m and :m2 to be evaluated in parallel as they need nothing from each other. In an effort to show a more realistic example, we assemble the openNLP chunking graph, discussed in Section 5.1, using this model:



```
(defgraph opennlp-chunking-graph
  :SD  (fn [sentence] (run opennlp-ssplit sentence))
  :TOK (fn [SD]  (run opennlp-tok SD))
  :POS (fn [TOK] (run opennlp-pos TOK))
  :CHU (fn [TOK POS]  (run opennlp-chunk TOK POS)) )
```

Given that none of the original openNLP classes had to be modified or wrapped in any way, in order to have them comply with the abstractions behind hotel-nlp, we argue that this is the simplest, cleanest, and yet most expressive, way to integrate foreign components and define graph-like relationships between them. The example presented does not mix-and-match any foreign components but it could have easily done. For instance, swapping out the openNLP tokeniser for the stanfordNLP one is as easy as replacing *opennlp-tok* with *stanfordnlp-tok* (assuming it has been previously defined). Similarly, for using the UIMA HMM-Tagger[1] instead of the openNLP POS-tagger, it suffices to replace *opennlp-pos* with *uima-hmm-pos*. As explained in Section 5.1, the equivalent process for integrating any library/tool with UIMA or GATE would most likely involve some sort of wrapping, whereas hotel-nlp can effortlessly extend its

[1]http://uima.apache.org/downloads/sandbox/hmmTaggerUsersGuide/hmmTaggerUsersGuide.html

abstractions over foreign types, thus making it possible to safely add behaviour to any concrete type without wrapping it. This is enabled by the programming language hotel-nlp is written in, and therefore can easily be achieved in other modern languages (i.e. Scala, FSharp). We invite the reader to contemplate the amount of effort, expertise, and ultimately, glue-code, it takes to achieve similar pieces of functionality in Java, as witnessed by the openNLP-uima[1], UIMA-GATE[2] and *cleartk-stanford-corenlp*[3] interoperability layers. Thousands of lines of code, whose sole purpose is to bridge the gap between UIMA and the corresponding API. *hotel-nlp* integrates with uimaFIT, and by consequence with UIMA, using, essentially, a single Java class (via the *clojuima* methodology) totalling 83 lines of code. Moreover, *hotel-nlp* provides Clojure bindings for openNLP, stanfordNLP and GATE in just over 500 lines of code[4], without coercing the original types (i.e. wrapping).

Features

As explained in the previous section, *hotel-nlp* primarily aims at simplifying interoperability and integration between foreign (incompatible) tools. In addition, it aims to provide a collection of tools and algorithms, useful for a number of less common TM tasks, but these are, by no means, unique.

At the top level, *hotel-nlp* provides a set of abstractions[5] that form the basis for interoperability between components. The public API exposed by *hotel-nlp* can be divided in four distinct packages:

- **core**: a small collection of convenience macros to assist the user in defining components and serial/graph work-flows, with the minimum amount of code possible. Despite being the “tip if the iceberg”, from a user’s perspective, this is the entry point to *hotel-nlp*.
- **externals**: the core of the interoperability layer. All the extension points between external libraries live in this namespace and they can be ‘activated’ on demand. This means that external dependencies can be loaded selectively by the user. For instance, there is no reason to activate the extensions for openNLP, which of

[1]`http://opennlp.apache.org/documentation/1.5.3/apidocs/opennlp-uima/`
[2]`http://gate.ac.uk/sale/tao/splitch20.html#chap:uima`
[3]`http://cleartk.googlecode.com/git/cleartk-stanford-corenlp/`
[4]`https://github.com/jimpil/hotel-nlp/blob/master/src/hotel_nlp/externals/bindings.clj`
[5]`https://github.com/jimpil/hotel-nlp/blob/master/src/hotel_nlp/protocols.clj`

course would require for openNLP to be included in the classpath, unless one plans to use openNLP components in some ‘alien’ context (e.g. GATE). In fact, the granularity of extensions can be even finer. One can choose to activate the extensions points for only a subset of components within a particular library or framework. Continuing with the previous example, one can go a step further, and only activate the extension points for all the NER components of openNLP, but nothing else.

- **algorithms**: a collection of high performance algorithms including *Levenshtein-Distance* (for approximate string matching), *N-Grams* (for n-gram modelling), *Smith-Waterman* (for local sequence alignment), *HMM & Viterbi* (for finding the most likely sequence of states), *Optics & K-Means* (for clustering), and more, implemented in native Clojure. Some of these algorithm implementations exhibit unique features, typically absent from equivalent implementations in other languages. For instance, the HMM model found in *hotel-nlp* can be trained on a corpus X, or two corpora Y and Z such that $Y + Z = X$, with exactly the same resulting model. Similarly, the K-Means algorithm can be deployed in parallel, something which, at the time of writing, very few and highly academic implementations achieve.

- **tools**: a collection of small projects that can also be as standalone. A good example of this is PAnnotator, previously discussed in Section 5.3.1. Recall that PAnnotator features a pluggable sentence-splitter and tokeniser. Fully realising this feature, requires that PAnnotator is used in the context of hotel-nlp, as PAnnotator itself does not define any abstractions. To put it differently, when using PAnnotator directly, the only thing one can “plug-in” is regular functions. In contrast, using PAnnotator as a hotel-nlp tool allows for any openNLP/stanfordNLP/GATE/UIMA-compliant sentence-splitter or tokeniser, to be “plugged-in”. Other tools include a trainable name generator useful for capturing linguistic patterns of domain-specific nomenclature and randomly generating similar sounding ones (i.e. for generating novel drug names)), a wikipedia crawler/ scrapper, a reasonably fast spell-checker and a parallel brute-force string-finder (for cracking passwords). Since many of these tools do not relate directly with the aims behind hotel-nlp, none of the code in this package is implicitly loaded by hotel-nlp.

- **helper**: a large collection of helper functions. This package is slightly cluttered and not particularly focused, as it includes many disparate pieces of functionality ranging from efficient document-readers (e.g. csv or tsv) to complex string transformations. In addition, some of the most important data-types reside in this package.

- **app**:*hotel-nlp*'s latest feature is a rudimentary Graphical User Interface (GUI). Currently, the GUI can still be considered experimental and is only functional with respect to integrating the various libraries and frameworks. In other words, a user can perform common tasks like sentence-detection, tokenisation, POS-tagging, chunking and parsing using any component from any of the aforementioned libraries. None of the extra tools are present in the GUI. Admittedly, the GUI is not very useful at this point.

The table below presents a comparison of the aforementioned platforms regarding their high-level architectural considerations and major features. Even though it quickly becomes evident that hotel-nlp is nowhere near as mature as the other solutions, it does provide interoperability hooks for all major libraries and platforms. GATE also allows, to some degree, the conversion between GATE's "document-analysers" and UIMA's "analysis engines". It needs to be noted that, much like GATE, hotel-nlp is primarily intended to support programmatic access and deployment of TM components, whereas U-Compare and Argo mainly focus on interactivity via a user-interface, which is thought to be a more intuitive and easier to grasp, general-audience medium [158]. Naturally, commonalities between these platforms are bound to exist. Generally speaking, replacing UIMA or GATE is not one of hotel-nlp's aspirations. Instead, hotel-nlp recognises that a plethora of quality components already exist and thus, was designed and built to be symbiotic with the other platforms and libraries. Admittedly, such a design consideration is usually overlooked. For example, UIMA-based solutions only care about UIMA-compliant components. GATE on the other hand, as it was previously mentioned, does provide some rudimentary facilities for converting GATE components to UIMA ones and vice-versa. Even though this is certainly useful and in line with hotel-nlp's philosophy, there are several other libraries that GATE knows nothing about.

As it currently stands, *hotel-nlp* has several rough edges, and there lies the reason for not having been released yet. Overall, it is a significant code-base in need of significant care before it can be considered consistent and polished. Nevertheless, we consider this prototype platform a successful proof-of-concept, with the concept being

	U-Compare	**GATE**	**hotel-NLP**	**ARGO**
Serial work-flows	+	+	+	+
Graph work-flows	0	0	+	+
Serialisation	+	+	+	+
Evaluation	+	+	0	+
Utilities	+	+	0	+
API	-	+	+	0
GUI	+	+	0	+
Plugin support	+	+	+	+
Audio/Video/Image	+	0	-	+
Ontology support	-	+	-	+
Interop. layers	-	0	+	-
Standalone	+	+	+	-
Online	-	-	-	+
Remote collaboration	-	-	-	+

Table 5.1: Comparison of NLP platforms:"+" fully supported, "0" partially supported, "-" not supported

the fact that in certain programming languages, interoperability and integration can go both ways ($platform_x \longleftrightarrow platform_y$) and they need not have an impact on the types involved, thus allowing for a clean separation between platform-specific code and user code.

Future Work

What the future holds for *hotel-nlp* is unfortunately, unclear, as it depends on certain human factors. In an ideal, world where the project could grow at a steady pace, there are several issues that need to be addressed. These are mostly of technical nature (e.g. bugs, performance issues, code-organisation) and thus, falls outside the scope of this thesis.

5.3.4 PKarus

Introduction

Simply put, *PKarus* can be thought of as the software embodiment of the work presented in Chapters 3 and 4. The research and experiments discussed in those chapters produced various models, which are all included in this project. The static NER models are autonomous and can be used in any openNLP code-base. Similarly, the evolved regex patterns can be used as is, in any JVM-based language. Translation to a language that runs on different platform (e.g. Python, .NET, Ruby) is a matter of minutes for someone familiar with both languages. Unfortunately, the situation is quite different for the parser generated from the CFG, presented in Chapter 4. Even though the CFG itself, being a simple string, is fully portable, the same cannot be said for the resulting parser, as it depends on Clojure-specific constructs and libraries. Users of JVM-based languages can of course, address this by programming against PKarus and thereby Clojure, from their own language. This approach is theoretically viable but interacting with a language from within a different one usually means that the resulting code will most likely be somewhat, unidiomatic. A more elegant solution would be to use PKarus in the context of *hotel-nlp* which, in order to facilitate the highly polymorphic behaviour between components, had to be structured around records and protocols. Protocols in Clojure have similar semantics with Java interfaces, whereas records are stateless objects that can implement the behaviour specified in protocols. This alone, makes code structured around protocols immediately usable from Java, Scala or Groovy, with familiar, to developers of these languages, semantics.

Features

PKarus has limited functionality and it targets highly domain-specific language, namely pharmacological/biomedical. The top-level API exposed by *PKarus* is essentially, two functions. These are summarised below:

- **recognise-drugs**: As the name implies, this is the function responsible for drug NER. In its simplest form, this function expects a string as input and utilises the best combination of models, as presented in Chapter 3, in order to output a hash-map from drug-names to their corresponding start/end offsets. Overloaded signatures of this function, that enable a finer control over what models are being used, are provided for convenience. The state-of-the-art NER performance demonstrated by our models combined with the programmability of the resulting data-structure, means that this function can trivially be used to produce dictionaries. All one needs is some means of extracting the keys out of the final hash-map, something which naturally, any language provides.

- **parse-pk**: This is the function responsible for generating parse-trees that capture PK interactions, according to the CFG presented in Chapter 4. Contrary to *“recognise-drugs”*, *parse-pk* currently has no overloaded signatures, and thus, there is no way for the user to reach into the lexical-analyser components. Therefore, a simple string serves as the input and a nested data-structure (the parse-tree) as the output. It needs to be noted that *PKarus* does not have any tree-traversing opinions or capabilities. It is completely up to the user to choose how to transform or traverse the resulting parse-tree. Finally, this function depends on *"recognise-drugs"* for its NER step, which in turn, depends on the presence of a sentence-splitter and tokeniser.

Much like most of the software mentioned in this Section, *PKarus* is a rather small and focused tool. It does two things, and two things only. Nonetheless however, it performs both tasks quite efficiently and effectively as demonstrated by the results presented in the corresponding Chapters.

Future Work

Just like *PAnnotator*, *PKarus* will benefit greatly from being integrated with *hotel-nlp*. Doing so would enable a certain degree of polymorphism within the analyser, thus allowing the user to potentially swap out certain components. Moreover, becoming a *hotel-nlp* tool, also means that the functionality can be easily used in the context of other platforms. For example recall that, *clojuima* provided a methodology for turning any function into UIMA compliant (and vice-versa), which *hotel-nlp* faithfully implements.

5.3.5 Other

The author was accepted as a contributor to the apache openNLP project on May 2012. Since then, several extensions, improvements and bug-fixes have been submitted. The prediction aggregation algorithm mentioned in Chapter 3 and a brand new Dictionary class with better hashing, are currently being reviewed for inclusion in the 1.6 release.

5.4 Discussion

This chapter presented the author's software contributions in the wider TM and NLP discipline. The code for every single project that discussed is on GitHub[1] and visible to anyone. Admittedly, the majority of the code, is written in Clojure. However, efforts

[1]`https://github.com`

have been made to make these tools accessible not only to Clojure developers, but also to Java developers as well. In fact, certain tools, such as *PAnnotator*, were created with primarily, the non-programmer in mind.

Overall, the tools presented in this chapter contain novel functionality, at least within the Clojure ecosystem, and in some cases, even in the wider, vast JVM landscape. Historically speaking, whenever a new language comes along, enthusiasts around the world start porting and translating algorithms to that language. With very few exceptions, doing that was generally, avoided in this thesis. In other words, there is far more *bridging* than *porting* code. The exceptions have mostly to do with licensing issues (not everything is open-source), and the fact that some solutions out there are simply, unsatisfactory. For example, a good example of an open-source library that performs well and has been architectured sensibly, would be the *Snowball Porter Stemmer*[1]. There is absolutely no reason for anyone, ever to reimplement this functionality on any JVM-based language. The same cannot be said for the, seemingly simple, N-Grams algorithm. In this case, there is absolutely no reason to even consider depending on some foreign library, as the entirity of N-Grams can be implemented from scratch, in a single line of Clojure code[2]. Similarly, the HMM implementation found in *hotel-nlp* offers a rather unique feature, which is the ability to build a model incrementally, instead all at once. This could useful for several reasons. In the context of POS-tagging, for example, it might be the case that the desirable corpus is split in many pieces or that some pieces are to become available in the future. To our knowledge, no other HMM implementation lends itself well to such circumstances. Typically, a model cannot be updated after construction. Therefore, “reinventing the wheel”, in such cases, can be justified on the grounds of adding useful and novel functionality.

We conclude that, even though most of the resulting artefacts (models and software) are primarily tailored for drug-discovery, significant effort has gone into making them as general and as easily accessible as possible. For the most part, the fact that these tools are written in a new and unconventional language, can be considered an implementation detail, which the end-user need not be aware of. Given that, some of these tools already have users and that they proved indispensable for conducting the experimental work presented in Chapters 3 and 4, we can only assume that, at least the motivation and core principles behind them are sound. The actual implementation is bound to be less than ideal, simply because it is rather difficult for a single maintainer

[1]`http://snowball.tartarus.org/index.php`

[2]`https://github.com/jimpil/hotel-nlp/blob/master/src/hotel_nlp/algorithms/ngrams.clj#L8`

to manage all these projects in parallel to his/her studies. As a consequence, corners are often being cut, which in turn, can lead to bugs, API inconsistencies, and much more.

Chapter 6

Conclusion

This thesis summarises the work undergone by the author in the last 18 months of research. Chapters 3 and 4 have a strong focus around novel methods and techniques for specific, real-world problems, whereas Chapter 5 is clearly concerned with the wider infrastructure that supports, and ultimately enables, experimental research in the field. Taking into account the importance of both aspects and the results presented so far, we consider the contributions presented worthy of attention, and in some cases, deserving further pursuit.

The experimental methods discussed are directly applicable to certain computational aspects of drug-discovery. For instance, despite the growing availability of corpora, data-sparsity is still a major problem for many highly specialised domains and tasks, such as pharmacology and PK-parameter mining. Chapter 3 suggests approaches to minimise the impact of this problem when recognising drug names, which in turn, is a prerequisite for effective relation extraction and event-mining in this area. In fact, part of the success of the method presented in Chapter 4, can be attributed to the robust NER models produced previously. Moreover, since Chapter 4 utilised a good proportion of the the infrastructure discussed in Chapter 5, it can be thought of as the work that ties everything else together.

While at first glance, Chapter 5 may sound overly critical of existing tools and platforms, it needs to be emphasised that the critique has very little to do with the tools themselves. Platforms and tools were conceived at a specific point in time and realised in a specific programming language/paradigm. In the absence of time-travelling facilities, there is nothing that can be done about it. Blaming the tools without offering alternatives, or blindly disregarding them, is by no means, constructive. On the other hand, finding ways to cleanly integrate or combine them, while at the same time protecting the user from unnecessary complexities, we believe adds great value not just to the TM and NLP communities, but also to the fledgling Clojure ecosystem.

Finally, recall that all the concrete artefacts produced have been made freely available to anyone who is willing to ask or look for them. In particular, the MaxEnt models can be retrieved directly from the author, while the various tools and libraries, except hotel-nlp, can be automatically fetched via standard software project management systems like *Maven* (Java), *Gradle* (Groovy) or *Clojars* (Clojure). More details can be found on each project's GitHub page. Despite the interesting ideas and motivation behind it, *hotel-nlp*, in its current state, is neither mature, nor polished enough to be released to the public.

Bibliography

[1] AlchemyAPI. http://www.alchemyapi.com/.

[2] BioNLP. http://bionlp.org/.

[3] CRAFT: The Colorado richly annotated full text corpus. http://bionlp-corpora.sourceforge.net/CRAFT/.

[4] European BioInformatics Institute. http://www.ebi.ac.uk/.

[5] High Wire Press. http://www.highwire.org/.

[6] National Center for Text Mining. http://www.nactem.ac.uk/.

[7] Neuroscience Information Framework. http://www.neuinfo.org/.

[8] NLTK. http://nltk.org/.

[9] Orange Book: Approved Drug Products with Therapeutic Equivalence Evaluations. http://www.accessdata.fda.gov/Scripts/cder/ob/default.cfm.

[10] ORBIT project. http://orbit.nlm.nih.gov/.

[11] Pharmacogenomics Knowledge base. http://www.pharmgkb.org/.

[12] PubMed Central - Open Access Subset. http://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/.

[13] Specialist NLP Tools. http://lexsrv3.nlm.nih.gov/Specialist/Home/index.html.

[14] University of Pittsburg NLP repository. http://www.dbmi.pitt.edu/nlpfront.

[15] GENIA corpus: a semantically annotated corpus for Bio-text-mining. *Bioinformatics*, 19(suppl 1):i180–i182, July 2003.

[16] Literature mining in support of drug discovery. *Briefings in Bioinformatics*, 9(6):479–92, Nov. 2008.

[17] A. B. Abacha and P. Zweigenbaum. Medical entity recognition: a comparison of semantic and statistical methods. In *Proceedings of BioNLP 2011 Workshop*, BioNLP '11, pages 56–64, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[18] A. Airola, S. Pyysalo, J. Bjorne, T. Pahikkala, F. Ginter, and T. Salakoski. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*, 9(Suppl 11):S2, 2008.

[19] F. Al-Anzi. Sentential count rules for arabic language. *Computers and the Humanities*, 35(2):153–166, 2001.

[20] K. Al-Kofahi, A. Tyrrell, A. Vachher, T. Travers, and P. Jackson. Combining multiple classifiers for text categorization. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 97–104, New York, NY, USA, 2001. ACM.

[21] B. Alex, B. Haddow, and C. Grover. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, BioNLP '07, pages 65–72, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[22] Alias-i. LingPipe 4.1.0. `http://alias-i.com/lingpipe`, 2008.

[23] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[24] S. F. Altschul, T. L. Madden, A. A. Schffer, 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 Research*, 25(17):3389–3402, 1997.

[25] S. Ananiadou and J. Mcnaught. *Text Mining for Biology And Biomedicine*. Artech House, Inc., Norwood, MA, USA, 2005.

[26] S. Ananiadou, S. Pyysalo, J. Tsujii, and D. B. Kell. Event extraction for systems biology by text mining the literature. *Trends in Biotechnology*, 28(7):381 – 390, 2010.

[27] Apache Software Foundation. openNLP. `http://opennlp.apache.org/`.

[28] C. N. Arighi, Z. Lu, M. Krallinger, K. B. Cohen, W. J. Wilbur, A. Valencia, L. Hirschman, and C. H. Wu. Overview of the BioCreative III Workshop. *BMC Bioinformatics*, 12(S-8):S1, 2011.

[29] A. R. Aronson and F.-M. Lang. An Overview of MetaMap: Historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236, 2010.

[30] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics*, 25(1):25–29, May 2000.

[31] W. A. Baumgartner, K. B. Cohen, L. M. Fox, G. Acquaah-Mensah, and L. Hunter. Manual curation is not sufficient for annotation of genomic databases. *Bioinformatics*, 23(13):i41–i48, 2007.

[32] A. Ben Abacha and P. Zweigenbaum. A Hybrid Approach for the Extraction of Semantic Relations from MEDLINE Abstracts. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 6609 of *Lecture Notes in Computer Science*, pages 139–150. Springer Berlin Heidelberg, 2011.

[33] P. N. Bennett, S. T. Dumais, and E. Horvitz. The combination of text classifiers using reliability indicators. *Inf. Retr.*, 8(1):67–100, Jan. 2005.

[34] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, Mar. 1996.

[35] Y. Bi, S. Mcclean, and T. Anderson. Combining rough decisions for intelligent text mining using dempster's rule. *Artif. Intell. Rev.*, 26(3):191–209, Nov. 2006.

[36] J. Björne, F. Ginter, S. Pyysalo, J. Tsujii, and T. Salakoski. Scaling up biomedical event extraction to the entire pubmed. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, BioNLP '10, pages 28–36, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[37] J. Björne, J. Heimonen, F. Ginter, A. Airola, T. Pahikkala, and T. Salakoski. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, BioNLP '09, pages 10–18, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[38] K. W. Boyack, D. Newman, R. J. Duhon, R. Klavans, M. Patek, J. R. Biberstine, B. Schijvenaars, A. Skupin, N. Ma, and K. Brner. Clustering more than two million biomedical publications: Comparing the accuracies of nine text-based similarity approaches. *PLoS ONE*, 6(3):e18029, 03 2011.

[39] B. Bringert. Numeral translator. `http://www.cs.chalmers.se/~bringert/gf/translate/`, 2004.

[40] A. C. Browne, G. Divita, A. R. Aronson, and A. T. McCray. UMLS Language and Vocabulary Tools. *AMIA Annual Symposium proceedings AMIA Symposium AMIA Symposium*, 2003:798, 2003.

[41] M. Bundschus, M. Dejori, M. Stetter, V. Tresp, and H.-P. Kriegel. Extraction of semantic biomedical relations from text using conditional random fields. *BMC Bioinformatics*, 9(1):207, 2008.

[42] E. Buyko, E. Faessler, J. Wermter, and U. Hahn. Event extraction from trimmed dependency graphs. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, BioNLP '09, pages 19–27, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[43] Y. Cai and X. Cheng. Biomedical Named Entity Recognition with Tri-Training Learning. In *Biomedical Engineering and Informatics, 2009. BMEI '09. 2nd International Conference on*, pages 1–5, 2009.

[44] D. Campos, S. Matos, and J. L. Oliveira. Gimli: open source and high-performance biomedical name recognition. *BMC Bioinformatics*, 14(1):54, 2013.

[45] D. T.-H. Chang, Y.-Z. Weng, J.-H. Lin, M.-J. Hwang, and Y.-J. Oyang. Protemot: prediction of protein binding sites with automatically extracted geometrical templates. *Nucleic Acids Research*, 34(suppl 2):W303–W309, 2006.

[46] M. Chang, S. Kenley, J. Bull, Y.-Y. Chiu, W. Wang, C. Wakeford, and K. McCarthy. Innovative approaches in drug development. *Journal of Biopharmaceutical Statistics*, 17(5):775–789, 2007. PMID: 17885865.

[47] W. W. Chapman and K. B. Cohen. Guest Editorial: Current issues in biomedical text mining and natural language processing. *Biomedical Informatics*, 42(5):757–759, Oct. 2009.

[48] E. S. Chen, G. Hripcsak, H. Xu, M. Markatou, and C. Friedman. Automated acquisition of diseasedrug knowledge from biomedical and clinical documents: An initial study. *Journal of the American Medical Informatics Association*, 15(1):87–98, 2008.

[49] J. Chien, S. Friedrich, M. Heathman, D. Alwis, and V. Sinha. Pharmacokinetics/pharmacodynamics and the stages of drug development: Role of modeling and simulation. *The AAPS Journal*, 7(3):E544–E559, 2005.

[50] N. Chomsky. Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124, 1956.

[51] H. W. Chun, Y. Tsuruoka, J. dong Kim, R. Shiba, N. Nagata, and T. Hishiki. Extraction of gene-disease relations from MEDLINE using domain dictionaries and machine learning. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 4–15, 2006.

[52] A. M. Cohen and W. R. Hersh. A survey of current work in biomedical text mining. *Briefings in bioinformatics*, 6(1):57–71, Mar. 2005.

[53] K. B. Cohen and L. Hunter. Getting started in text mining. *PLoS Comput Biol*, 4(1):e20, 01 2008.

[54] K. B. Cohen, K. Verspoor, H. L. Johnson, C. Roeder, P. V. Ogren, W. A. Baumgartner, Jr., E. White, H. Tipney, and L. Hunter. High-precision biological event extraction with a concept recognizer. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, BioNLP

'09, pages 50–58, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[55] N. Collier, C. Nobata, and J.-i. Tsujii. Extracting the names of genes and gene products with a Hidden Markov model. In *Proceedings of the 18th conference on Computational linguistics - Volume 1*, COLING '00, pages 201–207, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[56] A. Copestake, P. Corbett, P. Murray-rust, A. Siddharthan, S. Teufel, and B. Waldron. An architecture for language processing for scientific texts. In *4th UK E-Science All Hands Meeting*, 2006.

[57] P. Corbett and A. Copestake. Cascaded classifiers for confidence-based chemical named entity recognition. *BMC Bioinformatics*, 9(Suppl 11):S4, 2008.

[58] A. Dada. Implementation of the arabic numerals and their syntax in gf. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, Semitic '07, pages 9–16, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[59] I. Dagan and S. P. Engelson. Committee-based sampling for training probabilistic classifiers. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. Morgan Kaufmann, 1995.

[60] J. A. Dimasi, R. W. Hansen, and H. C. Grabowski. The price of innovation: new estimates of drug development costs. *Journal of Health Economics*, pages 141–185, 2003.

[61] R. Feldman. Mining associations in text in the presence of background knowledge. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 343–346, 1996.

[62] R. Feldman and I. Dagan. Knowledge Discovery in Textual Databases (KDT). In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 112–117, 1995.

[63] R. Feldman, I. Dagan, and W. Kloesgen. Efficient algorithms for mining and manipulating associations in texts. In *Proceedings of EMCSR96*, pages 949–954, Apr. 1996.

[64] C. Feng, F. Yamashita, and M. Hashida. Automated extraction of information from the literature on chemical-cyp3a4 interactions. *Journal of Chemical Information and Modeling*, 47(6):2449–2455, 2007.

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

[66] J. Finkel, S. Dingare, H. Nguyen, M. Nissim, C. Manning, and G. Sinclair. Exploiting context for biomedical entity recognition: from syntax to the web. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, JNLPBA '04, pages 88–91, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[67] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.

[68] K. Franzén, G. Eriksson, and F. Olsson. Protein names and how to find them. In *International Journal of Medical Informatics*, volume 4, pages 49–61, Swedish Institute of Computer Science, 2002.

[69] R. A. Frost, R. Hafiz, and P. Callaghan. Parser combinators for ambiguous left-recursive grammars. In *Practical Aspects of Declarative Languages*, pages 167–181, 2008.

[70] K. Fukuda, T. Tsunoda, A. Tamura, and T. Takagi. Toward Information Extraction: Identifying protein names from biological papers. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 707–718, 1998.

[71] K. Fundel, R. Kffner, and R. Zimmer. RelExRelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[72] R. Gaizauskas, G. Demetriou, P. Artymiuk, and P. Willett. Protein Structures and Information Extraction from Biological Texts: The PASTA System. *Journal of Bioinformatics*, 19(1):135–143, 2003.

[73] B. Gu. Recognizing nested named entities in GENIA corpus. In *Proceedings of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis*, BioNLP '06, pages 112–113, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[74] K. Gupta and R. Mahajan. Food and drug administration's critical path initiative and innovations in drug development paradigm: Challenges, progress, and controversies. *Journal of Pharmacy And Bioallied Sciences*, 2(4):307–313, 2010.

[75] N. Habash and R. Roth. Identification of Naturally Occurring Numerical Expressions in Arabic, 2008.

[76] J. Hakenberg, S. Bickel, C. Plake, U. Brefeld, H. Zahn, L. Faulstich, U. Leser, and T. Scheffer. Systematic feature evaluation for gene name recognition. *BMC Bioinformatics*, 6(Suppl 1):S9, 2005.

[77] J. Hakenberg, C. Plake, U. Leser, H. Kirsch, and D. Rebholz-Schuhmann. Lll'05 challenge: genic interaction extraction with alignments and finite state automata. In *Proceedings of Learning Language in Logic Workshop (LLL05) at the 22nd Int Conf on Machine Learning*, pages 38–45, 2005.

[78] M. A. Hearst. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 3–10. Association for Computational Linguistics, 1999.

[79] S. Heinen, B. Thielen, and D. Schomburg. Kid - an algorithm for fast and efficient text mining used to automatically generate a database containing kinetic information of enzymes. *BMC Bioinformatics*, 11(1):375, 2010.

[80] W. Hersh. *Information Retrieval: A Health and Biomedical Perspective (Health Informatics)*. Springer, 3rd edition, Nov. 2008.

[81] K. M. Hettne, R. H. Stierum, M. J. Schuemie, P. J. M. Hendriksen, B. J. A. Schijvenaars, E. M. v. Mulligen, J. Kleinjans, and J. A. Kors. A dictionary to identify small molecules and drugs in free text. *Bioinformatics*, 25(22):2983–2991, Nov. 2009.

[82] L. Hirschman, M. Colosimo, A. Morgan, and A. Yeh. Overview of BioCreAtIvE task 1B: normalized gene lists. *BMC Bioinformatics*, 6(Suppl 1):S11, 2005.

[83] L. Hirschman, A. A. Morgan, and A. S. Yeh. Rutabaga by any other name: extracting biological names. *Biomedical Informatics*, 35(4):247–259, 2002.

[84] L. Hirschman, A. Yeh, C. Blaschke, and A. Valencia. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC Bioinformatics*, 6(Suppl 1):1–10, 2005.

[85] W.-J. Hou and H.-H. Chen. Enhancing performance of protein name recognizers using collocation. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*, volume 13 of *BioMed '03*, pages 25–32, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[86] K. Humphreys, G. Demetriou, and R. Gaizauskas. Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures. In *Proceedings of the Pacific Symposium in Biocomputing*, pages 502–513, 2000.

[87] L. Hunter, Z. Lu, J. Firby, W. Baumgartner, H. Johnson, P. Ogren, and K. B. 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(1):78, 2008.

[88] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2 edition, 2009.

[89] R. Kabiljo, A. Clegg, and A. Shepherd. A realistic assessment of methods for extracting gene/protein interactions from free text. *BMC Bioinformatics*, 10(1):233, 2009.

[90] Y. Kano, W. A. Baumgartner, L. McCrohon, S. Ananiadou, K. B. Cohen, L. Hunter, and J. Tsujii. U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*, 25(15):1997–1998, 2009.

[91] H. Karsten and H. Suominen. Mining of clinical and biomedical text and data: Editorial of the special issue. *International Journal of Medical Informatics*, 78(12):786787, 2009.

[92] L. Karttunen. Numbers and finnish numerals. *SKY Journal of Linguistics*, 19:40–421, 2006.

[93] N. Katsuma, W. Yotaro, M. Junta, O. Naoaki, and I. Kentaro. Is a 204 cm man tall or small ? acquisition of numerical common sense from the web. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, Aug. 2013.

[94] J. Kazama, T. Makino, Y. Ohta, and J. Tsujii. Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the ACL-02 Workshop on Natural Language Processing in the Biomedical Domain*, pages 1–8, Phildadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[95] S. Kerrien, B. Aranda, L. Breuza, A. Bridge, F. Broackes-Carter, C. Chen, M. Duesbury, M. Dumousseau, M. Feuermann, U. Hinz, C. Jandrasits, R. C. Jimenez, J. Khadake, U. Mahadevan, P. Masson, I. Pedruzzi, E. Pfeiffenberger, P. Porras, A. Raghunath, B. Roechert, S. Orchard, and H. Hermjakob. The IntAct molecular interaction database in 2012. *Nucleic Acids Research*, 40(D1):D841–D846, 2012.

[96] H. Kilicoglu and S. Bergler. Syntactic dependency based heuristics for biological event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, BioNLP '09, pages 119–127, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[97] J.-D. Kim, T. Ohta, S. Pyysalo, Y. Kano, and J. Tsujii. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9. Association for Computational Linguistics, June 2009.

[98] J.-D. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, JNLPBA '04, pages 70–75, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[99] J.-D. Kim, S. Pyysalo, T. Ohta, R. Bossy, N. Nguyen, and J. Tsujii. Overview of bionlp shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, BioNLP Shared Task '11, pages 1–6, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[100] S. Kim, J. Yoon, and J. Yang. Kernel approaches for genic interaction extraction. *Bioinformatics*, 24(1):118–126, Jan. 2008.

[101] S. Kinoshita, K. B. Cohen, P. Ogren, and L. Hunter. Biocreative task1a: entity identification with a stochastic tagger. *BMC Bioinformatics*, 6(Suppl 1):S4, 2005.

[102] I. Kola and J. Landis. Can the pharmaceutical industry reduce attrition rates? *Nature Reviews Drug Discovery*, 3(8):711–716, Aug. 2004.

[103] C. Kolărik, M. Hofmann-Apitius, M. Zimmermann, and J. Fluck. Identification of new drug classification terms in textual resources. *Bioinformatics*, 23(13):i264–i272, 2007.

[104] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 1 edition, Aug. 2009.

[105] G. Kontonatsios, I. Korkontzelos, B. Kolluru, P. Thompson, and S. Ananiadou. Deploying and sharing u-compare workflows as web services. *Journal of Biomedical Semantics*, 4(1):7, 2013.

[106] A. Kovacevic, A. Dehghan, M. Filannino, J. A. Keane, and G. Nenadic. Combining rules and machine learning for extraction of temporal expressions and events from clinical narratives. *Journal of the American Medical Informatics Association*, 2013.

[107] K. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, 1st edition, May 1999.

[108] M. Krallinger, F. Leitner, C. Rodriguez-Penagos, and A. Valencia. Overview of the protein-protein interaction annotation extraction task of BioCreative II. *Genome Biology*, 9(Suppl 2):S4, 2008.

[109] M. Krallinger, A. Morgan, L. Smith, F. Leitner, L. Tanabe, J. Wilbur, L. Hirschman, and A. Valencia. Evaluation of text-mining systems for biology: overview of the Second BioCreative community challenge. *Genome Biology*, 9(Suppl 2):1–9, 2008.

[110] M. Krallinger, A. Valencia, and L. Hirschman. Linking genes to literature: text mining, information extraction, and retrieval applications for biology. *Genome Biology*, 9(Suppl 2):S8, 2008.

[111] M. Krauthammer and G. Nenadic. Term identification in the biomedical literature. *Journal of Biomedical Informatics*, 37(6):512–526, Dec. 2004.

[112] M. Krauthammer, A. Rzhetsky, P. Morozov, and C. Friedman. Using BLAST for identifying gene and protein names in journal articles. *Gene*, 259(1-2):245 – 252, Dec. 2000.

[113] R. L. Lalonde, K. G. Kowalski, M. M. Hutmacher, W. Ewy, D. J. Nichols, P. A. Milligan, B. W. Corrigan, P. A. Lockwood, S. A. Marshall, L. J. Benincosa, T. G. Tensfeldt, K. Parivar, M. Amantea, P. Glue, H. Koide, and R. Miller. Model-based drug development. *Clinical Pharmacology & Therapeutics*, 82(1):21–32, 2007.

[114] R. Leaman and G. Gonzalez. Banner: An executable survey of advances in biomedical named entity recognition. In *Proceedings of the Pacific Symposium in Biocomputing*, 2008.

[115] J. Lee, D. Scott, M. Villarroel, G. Clifford, M. Saeed, and R. Mark. Open-access MIMIC-II database for intensive care research. In *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE*, pages 8315–8318, 2011.

[116] L. C. Lee, F. Horn, and F. E. Cohen. Automatic extraction of protein point mutations using a graph bigram association. *PLoS Comput Biol*, 3(2):e16, 02 2007.

[117] U. Leser and J. Hakenberg. What makes a gene name? Named entity recognition in the biomedical literature. *Briefings in Bioinformatics*, 6(4):357–369, 2005.

[118] M. Liberman, M. Mandel, and GlaxoSmithKline Pharmaceuticals R&D. PennBioIE CYP 1.0. *Linguistic Data Consortium*, Nov. 2008.

[119] M. Liberman, M. Mandel, and P. White. PennBioIE Oncology 1.0. *Linguistic Data Consortium*, Nov. 2008.

[120] D. Lindberg, B. Humphreys, and A. McCray. The Unified Medical Language System. *Methods of Information in Medicine*, 32(4):281–291, 1993.

[121] Y. Liu, E. Shriberg, A. Stolcke, and M. Harper. Comparing HMM, Maximum Entropy, and Conditional Random Fields for Disfluency Detection. In *Proceedings of Interspeech*, pages 3313–3316, Lisbon, Sept. 2005.

[122] Y. A. Lussier, T. Borlawsky, D. Rappaport, Y. Liu, and C. Friedman. PhenoGO: Assigning Phenotypic Context to Gene Ontology Annotations with Natural Language Processing. In *Pacific Symposium on Biocomputing*, pages 64–75. World Scientific, 2006.

[123] A. McCallum, D. Freitag, and F. C. N. Pereira. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[124] J. Mervis. Productivity counts–but the definition is key. *Science*, 309(5735):726, 2005.

[125] C. Mihăilă and R. T. B. Batista-Navarro. What's in a name? entity type variation across two biomedical subdomains. In *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 38–45, Apr. 2012.

[126] S. Mika and B. Rost. Protein names precisely peeled off free text. *Bioinformatics*, 20(suppl 1):i241–i247, 2004.

[127] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38:39–41, 1995.

[128] T. Mitsumori, S. Fation, M. Murata, K. Doi, and H. Doi. Gene/protein name recognition based on support vector machine using dictionary as features. *BMC Bioinformatics*, 6(Suppl 1):S8, 2005.

[129] M. Miwa, R. Saetre, J.-D. Kim, and J. Tsujii. Event extraction with complex event classification using rich features. *Journal of Bioinformatics and Computational Biology*, 8(1):131–146, 2010.

[130] M. Miwa, R. Saetre, Y. Miyao, and J. Tsujii. Protein-protein interaction extraction by leveraging multiple kernels and parsers. *International Journal of Medical Informatics*, 78(12):39–46, 2009.

[131] M. Miwa, P. Thompson, J. McNaught, D. Kell, and S. Ananiadou. Extracting semantically enriched events from biomedical literature. *BMC Bioinformatics*, 13(1):108, 2012.

[132] Y. Miyao, T. Ohta, K. Masuda, Y. Tsuruoka, K. Yoshida, T. Ninomiya, and J. Tsujii. Semantic retrieval for the accurate identification of relational concepts in massive textbases. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 1017–1024, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[133] Y. Miyao, K. Sagae, R. Saetre, T. Matsuzaki, and J. Tsujii. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25(3):394–400, Feb. 2009.

[134] A. Morgan, L. Hirschman, A. Yeh, and M. Colosimo. Gene name extraction using flybase resources. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*, volume 13 of *BioMed '03*, pages 1–8, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[135] A. Morgan, Z. Lu, X. Wang, A. Cohen, J. Fluck, P. Ruch, A. Divoli, K. Fundel, R. Leaman, J. Hakenberg, C. Sun, H.-h. Liu, R. Torres, M. Krauthammer, W. Lau, H. Liu, C.-N. Hsu, M. Schuemie, K. B. Cohen, and L. Hirschman. Overview of BioCreative II gene normalization. *Genome Biology*, 9(Suppl 2):S3, 2008.

[136] A. A. Morgan, L. Hirschman, M. Colosimo, A. S. Yeh, and J. B. Colombe. Gene name identification and normalization using a model organism database. *Journal of Biomedical Informatics*, 37(6):396–410, Dec. 2004.

[137] H. Müller, I. Eggel, J. Reisetter, C. E. Kahn, and W. Hersh. Overview of the clef 2010 medical image retrieval track. In *CLEF 2010 working notes*, 2010.

[138] H. Müller, A. García Seco de Herrera, J. Kalpathy-Cramer, D. Demner-Fushman, S. Antani, and I. Eggel. Overview of the imageclef 2012 medical image retrieval and classification tasks. In *CLEF 2012 working notes*, 2012.

[139] M. Narayanaswamy, K. E. Ravikumar, and K. Vijay-Shanker. A Biological Named Entity Recognizer. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 8, pages 427–438, 2003.

[140] National Center for Biotechnology Information. Entrez Programming Utilities Help. `http://www.ncbi.nlm.nih.gov/books/NBK25501/`, June 2009.

[141] R. Nawaz, P. Thompson, and S. Ananiadou. Negated bio-events: analysis and identification. *BMC Bioinformatics*, 14(1):14, 2013.

[142] C. Nobata, N. Collier, and J. ichi Tsujii. Automatic term identification and classification in biology texts. In *In Proc. of the 5th NLPRS*, pages 369–374, 1999.

[143] C. Ndellec. Learning Language in Logic - Genic Interaction Extraction Challenge. In *Proceedings of the Learning Language in Logic 2005 Workshop at the International Conference on Machine Learning*, 2005.

[144] P. Ogren and S. Bethard. Building Test Suites for UIMA Components. In *Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing (SETQA-NLP 2009)*, pages 1–4, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[145] D. Okanohara, Y. Miyao, Y. Tsuruoka, and J. Tsujii. Improving the scalability of semi-Markov conditional random fields for named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 465–472, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[146] N. Okazaki, S. Ananiadou, and J. Tsujii. Building a high quality sense inventory for improved abbreviation disambiguation. *Bioinformatics*, 2010.

[147] B. C. d. S. Oliveira and W. R. Cook. Extensibility for the masses: Practical extensibility with object algebras. In *Proceedings of the 26th European Conference on Object-Oriented Programming*, ECOOP'12, pages 2–27, Berlin, Heidelberg, 2012. Springer-Verlag.

[148] F. Olsson, G. Eriksson, K. Franzén, L. Asker, and P. Lidén. Notions of correctness when evaluating protein name taggers. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[149] S. Pakhomov. Semi-supervised Maximum Entropy based approach to acronym and abbreviation normalization in medical texts. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 160–167, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[150] G. Petasis, V. Karkaletsis, G. Paliouras, and C. D. Spyropoulos. Learning context-free grammars to extract relations from text. In M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. M. Avouris, editors, *Proceeding of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 303–307, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[151] D. Piliouras, I. Korkontzelos, A. Dowsey, and S. Ananiadou. Dealing with data-sparsity in drug named entity recognition. In *Proceedings of the IEEE International Conference on Healthcare Informatics (ICHI)*, pages 14–21, 2013.

[152] H. Poon and L. Vanderwende. Joint inference for knowledge extraction from biomedical literature. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2010.

[153] S. Pyysalo, A. Airola, J. Heimonen, J. Bjorne, F. Ginter, and T. Salakoski. Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics*, 9(Suppl 3):S6, 2008.

[154] S. Pyysalo, F. Ginter, J. Heimonen, J. Bjorne, J. Boberg, J. Jarvinen, and T. Salakoski. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(1):50, 2007.

[155] S. Pyysalo, T. Ohta, R. Rak, D. Sullivan, C. Mao, C. Wang, B. W. S. Sobral, J. Tsujii, and S. Ananiadou. Overview of the id, epi and rel tasks of bionlp shared task 2011. *BMC Bioinformatics*, 13(S-11):S2, 2012.

[156] D. Radev, S. Teufel, H. Saggion, W. Lam, J. Blitzer, H. Qi, A. Celebi, D. Liu, and E. Drabek. Evaluation challenges in large-scale document summarization. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 375–382. Association for Computational Linguistics, 2003.

[157] R. Rak, A. Rowley, and S. Ananiadou. Collaborative Development and Evaluation of Text-processing Workows in a UIMA-supported Web-based Workbench. In *LREC*, 2012.

[158] R. Rak, A. Rowley, W. Black, and S. Ananiadou. Argo: an integrative, interactive, text mining-based workbench supporting curation. *Database*, 2012, 2012.

[159] L. Ramshaw and M. Marcus. Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey, 1995. Association for Computational Linguistics.

[160] D. REBHOLZ-SCHUHMANN, A. J. J. YEPES, E. M. V. MULLIGEN, N. KANG, J. KORS, D. MILWARD, P. CORBETT, E. BUYKO, E. BEISSWANGER, and U. HAHN. Calbc silver standard corpus. *Journal of Bioinformatics and Computational Biology*, 08(01):163–179, 2010.

[161] S. Richard. Lextools: a toolkit for finite-state linguistic analysis, 2000.

[162] S. Riedel, H.-W. Chun, T. Takagi, and J. Tsujii. A markov logic approach to biomolecular event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, BioNLP '09, pages 41–49, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[163] S. Riedel and A. McCallum. Fast and robust joint models for biomedical event extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1–12, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[164] F. Rinaldi, K. Kaljurand, and R. Saetre. Terminological resources for text mining over biomedical scientific literature. *Artificial Intelligence in Medicine*, 52(2):107–114, 2011.

[165] F. Rinaldi, G. Schneider, K. Kaljurand, M. Hess, C. Andronis, O. Konstandi, and A. Persidis". Mining of relations between proteins over biomedical scientific literature using a deep-linguistic approach. *Artificial Intelligence in Medicine*, 39(2):127–136, 2007.

[166] T. Rindflesch, L. Tanabe, J. N. Weinstein, and L. Hunter. Edgar: Extraction of drugs, genes and relations from the biomedical literature. In *Pacific Symposium of Biocomputing*, pages 517–528, 2000.

[167] B. Rink, S. Harabagiu, and K. Roberts. Automatic extraction of relations between medical concepts in clinical texts. *Journal of the American Medical Informatics Association : JAMIA*, 18(5):594–600, 2011.

[168] A. Roberts, R. Gaizauskas, and M. Hepple. Extracting clinical relationships from patient narratives. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, BioNLP '08, pages 10–18, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[169] S. Robertson and D. Hull, A. TREC-9 filtering track collection. `http://trec.nist.gov/data/t9_filtering.html`, Dec. 2001.

[170] S. Robertson and D. Hull, A. Trec genomics track data. `http://ir.ohsu.edu/genomics/data.html`, Dec. 2001.

[171] Y. Sasaki, Y. Tsuruoka, J. McNaught, and S. Ananiadou. How to make the most of NE dictionaries in statistical NER. *BMC Bioinformatics*, 9(Suppl 11):S5, 2008.

[172] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. K. Schuler, and C. G. Chute. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *Journal of The American Medical Informatics Association*, 17:507–513, 2010.

[173] B. Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, JNLPBA '04, pages 104–107, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[174] B. Settles. ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.

[175] H. Shatkay, F. Pan, A. Rzhetsky, and W. J. Wilbur. Multi-dimensional classification of biomedical text: Toward automated, practical provision of high-utility text to diverse users. *Bioinformatics*, 24(18):2086–2093, 2008.

[176] D. Shen, J. Zhang, G. Zhou, J. Su, and C.-L. Tan. Effective adaptation of a Hidden Markov Model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*, volume 13 of *BioMed '03*, pages 49–56, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[177] L. Si. Boosting performance of bio-entity recognition by combining results from multiple systems. In *BIOKDD: Proceedings of the 5th international workshop on Bioinformatics*, pages 76–83. ACM Press, 2005.

[178] B. Sigurd. From numbers to numerals and vice versa. In *Proceedings of the 5th Conference on Computational Linguistics - Volume 2*, COLING '73, pages 429–455, Stroudsburg, PA, USA, 1973. Association for Computational Linguistics.

[179] M. Simpson and D. Demner-Fushman. Biomedical Text Mining: A Survey of Recent Progress. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, pages 465–517. Springer US, 2012.

[180] L. Smith, L. Tanabe, R. Ando, C.-J. Kuo, I.-F. Chung, C.-N. Hsu, Y.-S. Lin, R. Klinger, C. Friedrich, K. Ganchev, M. Torii, H. Liu, B. Haddow, C. Struble, R. Povinelli, A. Vlachos, W. Baumgartner, L. Hunter, B. Carpenter, R. Tsai, H.-J. Dai, F. Liu, Y. Chen, C. Sun, S. Katrenko, P. Adriaans, C. Blaschke, R. Torres, M. Neves, P. Nakov, A. Divoli, M. Mana-Lopez, J. Mata, and W. J. Wilbur. Overview of BioCreative II gene mention recognition. *Genome Biology*, 9(Suppl 2):S2, 2008.

[181] M. Q. Stearns, C. Price, K. A. Spackman, and A. Y. Wang. SNOMED clinical terms: Overview of the development process and project status. In *Proceedings of the AMIA Annual Symposium*, pages 662–666, 2001.

[182] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii. brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the*

Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics, Stroudsburg, PA, USA, Apr. 2012. Association for Computational Linguistics.

[183] A. Stubbs. MAE and MAI: Lightweight Annotation and Adjudication Tools. In *Proceedings of the 5th Linguistic Annotation Workshop*, LAW V '11, pages 129–133, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[184] D. R. Swanson. Fish oil, Raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18, Jan. 1986.

[185] K. Takeuchi and N. Collier. Bio-medical entity extraction using support vector machines. *Artif. Intell. Med.*, 33(2):125–137, Feb. 2005.

[186] J. M. Temkin and M. R. Gilder. Extraction of protein interaction information from unstructured text using a context-free grammar. *Bioinformatics*, 19(16):2046–2053, 2003.

[187] P. Thomas, T. Bobic, U. Leser, M. Hofmann-Apitius, and R. Klinger. Weakly labeled corpora as silver standard for drug-drug and protein-protein interaction. In *Proceedings of the Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM) on Language Resources and Evaluation Conference (LREC)*, pages 63–70, Instanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[188] C. A. Thompson, M. E. Califf, and R. J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 406–414. Morgan Kaufmann Publishers Inc., 1999.

[189] P. Thompson, S. Iqbal, J. McNaught, and S. Ananiadou. Construction of an annotated corpus to support biomedical information extraction. *BMC Bioinformatics*, 10(1):349, 2009.

[190] K. Tomanek, J. Wermter, and U. Hahn. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *In Proc. of EMNLP/CoNLL07*, pages 486–495, 2007.

[191] R. Tsai, W.-C. Chou, Y.-S. Su, Y.-C. Lin, C.-L. Sung, H.-J. Dai, I. Yeh, W. Ku, T.-Y. Sung, and W.-L. Hsu. BIOSMILE: A semantic role labeling system for biomedical verbs using a maximum-entropy model with automatically generated template features. *BMC Bioinformatics*, 8(1):325, 2007.

[192] J.-J. Tsay, B.-L. Wu, and C.-C. Hsieh. Automatic extraction of kinetic information from biochemical literatures. In *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery - Volume 5*, FSKD'09, pages 28–32, Piscataway, NJ, USA, 2009. IEEE Press.

[193] Y. Tsuruoka, J. McNaught, J. Tsujii, and S. Ananiadou. Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. *Bioinformatics*, 23(20):2768–2774, 2007.

[194] Y. Tsuruoka and J. Tsujii. Boosting precision and recall of dictionary-based protein name recognition. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*, volume 13 of *BioMed '03*, pages 41–48, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[195] Y. Tsuruoka and J. Tsujii. Probabilistic term variant generator for biomedical terms. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 167–173, New York, NY, USA, 2003. Association for Computational Linguistics.

[196] Y. Tsuruoka, J. Tsujii, and S. Ananiadou. Accelerating the annotation of sparse named entities by dynamic sentence selection. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, BioNLP '08, pages 30–37, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[197] O. Tuason, L. Chen, H. Liu, J. A. Blake, and C. Friedman. Biological Nomenclatures: A Source of Lexical Knowledge and Ambiguity. *Pacific Symposium on Biocomputing*, 9:238–249, 2004.

[198] K. Tymoshenko and C. Giuliano. FBK-IRST: Semantic relation extraction using Cyc. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 214–217, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[199] University of Sheffield. GATE. `http://gate.ac.uk/`.

[200] University of Stanford. stanfordNLP. `http://nlp.stanford.edu/`.

[201] Y. Usami, H.-C. Cho, N. Okazaki, and J. Tsujii. Automatic acquisition of huge training data for bio-medical named entity recognition. In *Proceedings of BioNLP 2011 Workshop*, BioNLP '11, pages 65–73, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

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

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

[204] O. Uzuner, I. Solti, and E. Cadag. Extracting medication information from clinical text. *Journal of the American Medical Informatics Association*, 17(5):514–518, Sept. 2010.

[205] O. Uzuner, B. R. South, S. Shen, and S. L. DuVall. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 2011.

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

[207] A. Vlachos and C. Gasperin. Bootstrapping and evaluating named entity recognition in the biomedical domain. In *Proceedings of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis*, BioNLP '06, pages 138–145, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[208] J. Šarić, L. J. Jensen, R. Ouzounova, I. Rojas, and P. Bork. Extraction of regulatory gene/protein networks from medline. *Bioinformatics*, 22(6):645–650, Mar. 2006.

[209] Z. Wang, S. Kim, S. K. Quinney, Y. Guo, S. D. Hall, L. M. Rocha, and L. Li. Literature mining on pharmacokinetics numerical data: A feasibility study. *Journal of Biomedical Informatics*, 42(4):726–735, Aug. 2009.

[210] T. Wattarujeekrit, P. Shah, and N. Collier. PASBio: predicate-argument structures for event extraction in molecular biology. *BMC Bioinformatics*, 5(1):155, 2004.

[211] D. S. Wishart, C. Knox, A. C. Guo, D. Cheng, S. Shrivastava, D. Tzur, B. Gautam, and M. Hassanali. Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Research*, 36(suppl 1):D901–D906, 2008.

[212] U. Wittig, R. Kania, M. Golebiewski, M. Rey, L. Shi, L. Jong, E. Algaa, A. Weidemann, H. Sauer-Danzwith, S. Mir, O. Krebs, M. Bittkowski, E. Wetsch, I. Rojas, and W. Mller. Sabio-rkdatabase for biochemical reaction kinetics. *Nucleic Acids Research*, 40(D1):D790–D796, 2012.

[213] H.-Y. Wu, S. Karnik, A. Subhadarshini, Z. Wang, S. Philips, X. Han, C. Chiang, L. Liu, M. Boustani, L. Rocha, S. Quinney, D. Flockhart, and L. Li. An integrated pharmacokinetics ontology and corpus for text mining. *BMC Bioinformatics*, 14(1):35, 2013.

[214] K. Yamamoto, T. Kudo, A. Konagaya, and Y. Matsumoto. Protein name tagging for biomedical annotation in text. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine*, volume 13 of *BioMed '03*, pages 65–72, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[215] Y. Yang, T. Ault, and T. Pierce. Combining multiple learning strategies for effective cross-validation. In *International Conference on Machine Learning*, pages 1167–1174, 2000.

[216] A. Yeh, A. Morgan, M. Colosimo, and L. Hirschman. BioCreAtIvE Task 1A: gene mention finding evaluation. *BMC Bioinformatics*, 6(Suppl 1):S2, 2005.

[217] A. Zanasi. *Text Mining and its Applications to Intelligence, CRM and Knowledge Management*. WIT Press, 2007.

[218] G. Zhou, D. Shen, J. Zhang, J. Su, and S. Tan. Recognition of protein/gene names from text using an ensemble of classifiers. *BMC Bioinformatics*, 6(Suppl 1):1–7, 2005.

[219] P. Zweigenbaum and D. Demner-Fushman. Advanced Literature-Mining Tools. In *Bioinformatics*, pages 347–380. Springer New York, 2009.

[220] P. Zweigenbaum, D. Demner-Fushman, H. Yu, and K. B. Cohen. Frontiers of biomedical text mining: current progress. *Briefings in Bioinformatics*, 8(5):358–375, 2007.