Natural language processing techniques for the purpose of sentinel event information extraction

by

Neil Barrett
B.Sc., McGill University, 2001
M.Sc., Memorial University of Newfoundland, 2007

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Natural language processing techniques for the purpose of sentinel event information extraction

by

Neil Barrett
B.Sc., McGill University, 2001
M.Sc., Memorial University of Newfoundland, 2007

Supervisory Committee

Dr. Jens H. Weber-Jahnke, Co-Supervisor
(Department of Computer Science)

Dr. Francis Lau, Co-Supervisor
(School of Health Information Science)

Dr. William Wadge, Departmental Member
(Department of Computer Science)

Dr. Sandra Kirkham, Outside Member
(Department of Linguistics)

Supervisory Committee

Dr. Jens H. Weber-Jahnke, Co-Supervisor
(Department of Computer Science)

Dr. Francis Lau, Co-Supervisor
(School of Health Information Science)

Dr. William Wadge, Departmental Member
(Department of Computer Science)

Dr. Sandra Kirkham, Outside Member
(Department of Linguistics)

ABSTRACT

An approach to biomedical language processing is to apply existing natural language processing (NLP) solutions to biomedical texts. Often, existing NLP solutions are less successful in the biomedical domain relative to their non-biomedical domain performance (e.g., relative to newspaper text). Biomedical NLP is likely best served by methods, information and tools that account for its particular challenges. In this thesis, I describe an NLP system specifically engineered for sentinel event extraction from clinical documents. The NLP system's design accounts for several biomedical NLP challenges. The specific contributions are as follows.

- Biomedical tokenizers differ, lack consensus over output tokens and are difficult to extend. I developed an extensible tokenizer, providing a tokenizer design pattern and implementation guidelines. It evaluated as equivalent to a leading biomedical tokenizer (MedPost).
- Biomedical part-of-speech (POS) taggers are often trained on non-biomedical corpora and applied to biomedical corpora. This results in a decrease in tagging

accuracy. I built a token centric POS tagger, TcT, that is more accurate than three existing POS taggers (mxpost, TnT and Brill) when trained on a non-biomedical corpus and evaluated on biomedical corpora. TcT achieves this increase in tagging accuracy by ignoring previously assigned POS tags and restricting the tagger's scope to the current token, previous token and following token.

- Two parsers, MST and Malt, have been evaluated using perfect POS tag input. Given that perfect input is unlikely in biomedical NLP tasks, I evaluated these two parsers on imperfect POS tag input and compared their results. MST was most affected by imperfectly POS tagged biomedical text. I attributed MST's drop in performance to verbs and adjectives where MST had more potential for performance loss than Malt. I attributed Malt's resilience to POS tagging errors to its use of a rich feature set and a local scope in decision making.

- Previous automated clinical coding (ACC) research focuses on mapping narrative phrases to terminological descriptions (e.g., concept descriptions). These methods make little or no use of the additional semantic information available through topology. I developed a token-based ACC approach that encodes tokens and manipulates token-level encodings by mapping linguistic structures to topological operations in SNOMED CT. My ACC method recalled most concepts given their descriptions and performed significantly better than MetaMap.

I extended my contributions for the purpose of sentinel event extraction from clinical letters. The extensions account for negation in text, use medication brand names during ACC and model (coarse) temporal information. My software system's performance is similar to state-of-the-art results. Given all of the above, my thesis is a blueprint for building a biomedical NLP system. Furthermore, my contributions likely apply to NLP systems in general.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to acknowledge the Coast and Straits Salish Peoples on whose traditional territories I have had the privilege to live for the duration of my degree. I thank you for sharing your teachings.

I acknowledge my supervisors and their role in this work. More generally, I am grateful for my committee's feedback, both with respect to research and writing. I also thank my external examiner, Dr. Özlem Uzuner, for her role and her kind approach during my oral exam. I deeply appreciate Dr. Vincent Thai and his team's help, with a special thanks to Rachel. To my family and friends, thank you for your support.

"I don't understand," said the scientist, "why you lemmings all rush down to the sea and drown yourselves."
"How curious," said the lemming. "The one thing I don't understand is why you human beings don't."

– James Thurber

Chapter 1

Introduction

Natural language is an important communication tool and is widely used to disseminate knowledge and data within biomedical domains (e.g., medicine) [38, 58]. Although language is patterned and organized, its processing is often complex and difficult. An inability to process natural language biomedical data can exclude information from computer processing, such as computer systems supporting healthcare professionals and biomedical researchers [38]. Improving biomedical language processing will allow computer systems to better support healthcare professionals, biomedical researchers and other individuals in the biomedical domains [38].

Natural language enabled computer systems could read clinical documents and extract important information. This information could include changes in a patient's physical and mental state. These state changes could trigger clinical care guidelines and care protocols across disperse health and geographic environments. Clinical care guidelines and care protocols are evidence-based best practices for patient care [51]. Triggering clinical care guidelines and care protocols could be as simple as making these guidelines and protocols contextually available to healthcare professionals. For example, computer systems could recognize a potential treatment plan and automatically provide clinical care guidelines and care protocols related to the treatment plan. This relieves the healthcare professional from searching for and synthesizing a large quantity of information including potential treatments and their costs. Furthermore, certain clinical care guideline and care protocol details such as scheduling a procedure (e.g., X-ray) could be completed autonomously by computer systems, after health professional approval.

Natural language processing (NLP) is the computer processing of human language

[59]. It may span from speech to language understanding - from sounds to semantics.[1] NLP may be applied to biomedical texts. Biomedical texts are biological and medical texts, such as clinical notes and research papers. For example, NLP was applied to chest X-ray reports to identify new and expanding neoplasms (abnormal tissue growth) for the purpose of monitoring patient follow-ups [137] and to discharge summaries to determine the severity of a patient's community acquired pneumonia [36]. NLP has also automatically extracted obesity and related information from patient summaries [125] and identified smokers from patient records [126].

One approach to biomedical NLP[2] is to apply existing NLP solutions to biomedical texts such as clinical letters. Often, existing NLP solutions are less successful in the biomedical domain relative to their non-biomedical domain performance (e.g., relative to newspaper text) [38, 94]. For example, many NLP systems assume grammatically correct text whereas some biomedical texts such as clinical notes may be exceptionally concise, contain spelling mistakes and be ungrammatical. In other words, existing NLP solutions are often built and trained assuming non-biomedical domain input. Biomedical NLP is likely best served by methods, information and tools that account for its particular challenges.

This thesis is a blueprint for building a biomedical NLP system for sentinel event information extraction from clinical documents. A sentinel event is "*an unexpected occurrence involving death or serious physical or psychological injury, or the risk thereof*"[3]. The NLP system's design accounts for several biomedical NLP challenges. These challenges stem from biomedical NLP difficulties (Section 2.3). I address these challenges with novel methods, algorithms and information. I evaluate my contributions using free and publicly available corpora. This selection increases reproducibility and helps examine how suitable free public corpora are for biomedical NLP. I argue that my biomedical NLP system design and construction is effective for sentinel event information extraction from clinical documents. My proposed system and its components are supported through empirical evidence. In particular, Chapter 7 evaluates the system as a whole.

The specific challenges are as follows:

1. **Challenge:** Separating text into tokens (e.g., words or punctuation) is called

[1] Other terminology includes natural language understanding (NLU) and natural language generation (NLG)

[2] May also be referred to as medical language processing (MLP).

[3] `www.jointcommission.org/SentinelEvents`

tokenization. Biomedical tokenization is often problematic due to atypical use of symbols and other irregularities in biomedical text. Furthermore, tokenizer idiosyncrasies, a lack of guidance on how to adapt and extend existing tokenizers to new domains and inconsistencies between tokenizer algorithms limit tokenizer use.

Contributions: I develop a novel approach for tokenizing biomedical text and provided implementation guidelines for my method.

Benefit: My approach provides consistency in tokenizer creation and is extensible (handles new tokens). It performs as well as a hand-crafted biomedical tokenizer.

Main parts: Section 2.8 and Chapter 3

2. **Challenge:** There is limited training data for training biomedical NLP systems. Consequently, biomedical NLP systems may be trained on non-biomedical data, negatively impacting performance. This is the case for algorithms that assign part-of-speech tags (e.g., noun or adjective) to tokens.

   Contributions: I develop an algorithm that performs better than several leading algorithms in situations when training occurs on non-biomedical data.

   Benefit: My algorithm improved performance for situations in which only non-biomedical data is available. This performance improvement positively affects NLP components that rely on (correct) part-of-speech tags. My approach also provides insight on how to adapt existing algorithms to restricted training conditions.

   Main parts: Section 2.9 and Chapter 4

3. **Challenge:** Syntactic parsers output structures that a computer may use to interpret text semantics. Most parsers rely on correct part-of-speech tags during processing. Two broadly successful parsers have not been evaluated on imperfect input (e.g., incorrect part-of-speech tags).

   Contributions: I test the two parser on imperfect input and compare their results.

   Benefit: Researchers and developers are better informed on parser characteristics. This may help developers chose the best parser for their needs and

help researchers address performance weaknesses manifested during testing.

Main parts: Section 2.10 and Chapter 5

4. **Challenge:** Standardized biomedical terminologies may be promising tools for biomedical computing. A difficulty is correctly encoding biomedical text (e.g., phrases, words or chunks) to standardized terminology entities.

   Contributions: I develop an algorithm for encoding biomedical text to standardized terminology entities that uses the standardized terminology's topography. My algorithm performs better than a leading system (MetaMap) during evaluation.

   Benefit: Given standardized terminologies' use in biomedical domains (e.g., search, data communication, record classification), improvements could impact many biomedical services.

   Main parts: Section 2.11 and Chapter 6

In addition to my four research contributions above, I combined and extended my contributions to an information extraction task. I compared my software system's ability to extract important information from clinical letters to a physician and to physician collected information. The target information is currently under study. This demonstrates my software system's ability to tackle a novel biomedical information extraction task. Given all of the above, my software system and its evaluation is a novel research contribution.

My thesis is structured as follows:

- Chapter 2 introduces readers to NLP systems, biomedical NLP difficulties and previously successful biomedical NLP techniques. It then presents the challenges addressed in subsequent chapters.

- Chapter 3 addresses the tokenization challenge by presenting my tokenization approach, its evaluation and discussion.

- Chapter 4 addresses the part-of-speech tagging challenge by presenting my part-of-speech tagging algorithm, its evaluation and discussion.

- Chapter 5 addresses the syntactic parsing challenge by comparing syntactic parsing algorithms and discussing these results.

- Chapter 6 addresses the challenge of encoding biomedical text in standardized terminologies by presenting my encoding algorithm, its evaluation and discussion.
- Chapter 7 describes my software extension, its application to extracting important clinical information, the information extraction results and discussion
- Chapter 8 concludes this thesis.

Chapter 2

Background

This chapter introduces readers to NLP systems (Section 2.2), biomedical NLP difficulties 2.3, several successful biomedical NLP techniques (Section 2.4) and the current target information (Section 2.5) for my biomedical NLP system. It then presents the challenges (Section 2.8, 2.9, 2.10 and 2.11) addressed in subsequent chapters. The last section (Section 2.12) describes the information extraction scenario used to empirically evaluate my entire biomedical NLP system. Prior to these sections, the following section briefly discusses software engineering research in order to frame ensuing material.

2.1 Software engineering research

There is ongoing debate on the definition, methodologies, communication and quality standards of software engineering research [114, 91, 44]. An informal survey of Google's top ten search results for "software engineering research" describes software engineering research as research into the development and maintenance of software systems. As an analogy, consider engineering a bicycle. Given the previous description, software engineering research would be limited to improving bicycle production and maintenance. It would exclude improving the bicycle itself. This informal survey may reflect a common perspective on software engineering research but excludes broader perspectives such as those that follow.

The three subsequent examples demonstrate different perspectives and approaches to defining and describing software engineering research. Shaw [114] describes software engineering research by generalizing from past research questions. She describes

several research products: qualitative or descriptive models, empirical models, analytic models, notation or tools, a specific solution, a judgement and experience reports. Shaw categorizes software engineering research as

- a method or means of development (e.g., What is a better way to create X?)
- a method for analysis (e.g., How can I evaluate the quality of X?)
- the design, evaluation or analysis of a particular instance (e.g., What is a better design or implementation for application X?)
- the generalization or characterization (e.g., What are the important characteristics of X?)
- a feasibility assessment (e.g., Is it possible to accomplish X?)

Montesis et al. [91] reviewed software engineering publications and developed publication genres. Given their genres, software engineering research includes empirical research (observational studies, case studies, field studies, experimental research, and meta-analyses), experience reports, theoretical papers and synthesis papers.

Gregg et al. [44] describe software engineering research as phase-based. It begins with the conceptualization phase which is followed by a formal or developmental phase, or both. During conceptualization, researchers conceptualize ideas and define the theoretical grounding of the research's needs and requirements. Conceptualization is followed by formalization where concepts are specified using established standards (e.g., mathematical models) and by development where concepts are validated in prototypes. Developing prototypes is an iterative process where subsequent developments are built on successes (Figure 2.1). In other words, Gregg et al. [44] suggest that software engineering research produces formal models or prototypes.

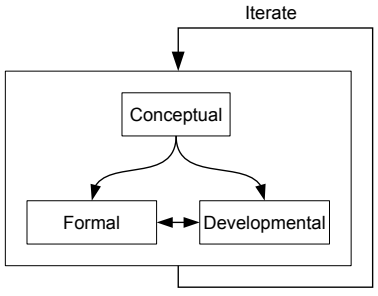

Figure 2.1: An iterative research method

In general, it may be difficult to find consensus among researchers on software engineering research since researchers rarely communicate their research paradigms [114]. Despite differing approaches to the definition and description of software engineering research, there is fair consensus on research quality. A preference exists for empirically evaluated software engineering research [44, 91, 114]. For example, Gregg et al. [44] define the highest quality software engineering research as research that creates novel conceptualizations which are formally defined or prototyped and that prototypes are verified and validated. I adopt Gregg et al.'s definition and description of software engineering research that includes, given the bicycle analogy, improving the bicycle itself.

2.2 NLP system components

When building a NLP system, the choice of system components depends on the system's goals. For example, extracting specifically formatted date information from text may only require pattern matching. On the other hand, building a computational understanding of text likely requires more complex linguistic structures and processing. Figure 2.2 presents a formulation of a NLP system. I use this formulation to introduce each system component and associated terminology. I present this formulation because it is the formulation followed by my proposed biomedical NLP system.

2.2.1 Sentence segmentation

Sentence segmentation is the act of separating text into sentences and sentence like segments. Separating text using periods, question marks and exclamation marks is an example of simple sentence segmentation. Text may also contain segments that should be treated as sentences. For example, section headings in documents should be considered individual segments rather than prepending or appending these segments to the preceding or following sentence.

2.2.2 Tokens and Tokenization

Tokenization is broadly defined as the segmentation of text into units for subsequent processing [129]. The segmented units are called tokens. For example, a tokenization

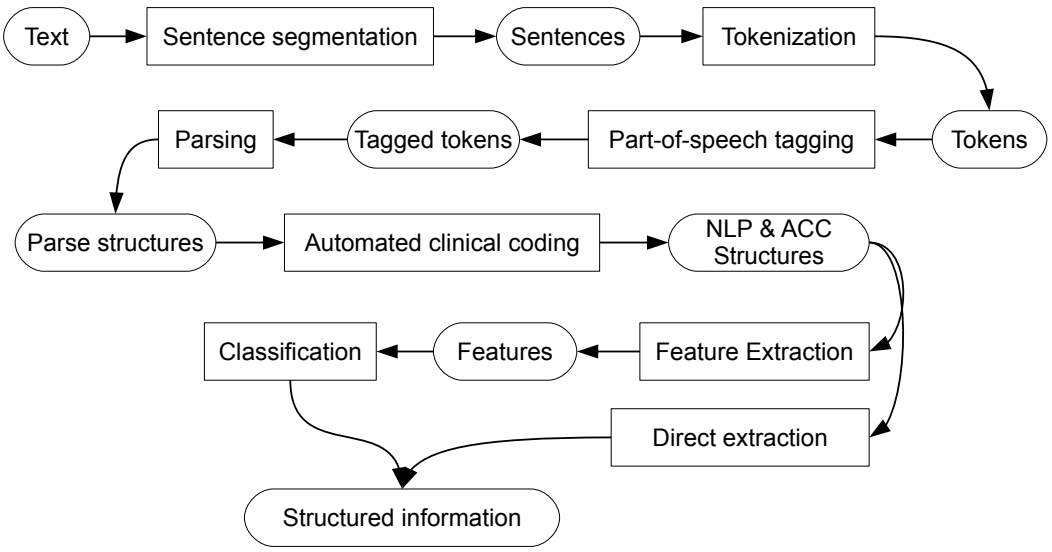

Figure 2.2: NLP system components

of "*The leg's fracture is long.*" could be "*The leg 's fracture is long .*". Tokenization is often a preliminary step in textual language processing and is an important component of language processing [129].

2.2.3 Part-of-speech tagging

Part-of-speech (POS) tagging assigns speech categories (as tags) to tokens [59], such as assigning the tag *noun* to the token *thorax*. A POS tag supplies information on its tagged word and on surrounding words [59]. For example, it is likely that a noun follows after the word *the* (e.g., the hands), whereas it is less likely that a verb follows *the* (e.g., the wrote). POS tags also affect word pronunciation in text to speech systems and improve information retrieval from textual documents, such as names, times, dates and other named entities [59].

2.2.4 Syntactic parsing

Syntactic parsing is the process of recognizing a sentence and assigning a structure to it [59]. Often, sentences are represented by a tree structure (e.g., Figure 2.3). Syntactic parsing may be split into shallow and deep parsing. Shallow parsing is a partial syntactic parse. It is useful in situations where a complete deep parse is unnecessary. For example, extracting people's names from text may be accomplished with shallow parsing of noun phrases because people's names are found in noun phrases. Deep parsing is a complete syntactic parse. Semantic analysis (semantic understanding) often depends on deep parse structures.

Syntactic parsing algorithms follow from two main grammar models: context free grammars (constituent structure or phrase-structure grammars) and dependency grammars. In context free grammars, phrase-structure rules state the components of a phrase. For example, a simple noun phrase such as "the severe fracture" may be represented by a determiner, an adjective and a noun, with a rule written as $NP \rightarrow DET\ ADJ\ NN$. In dependency grammars, dependency structure is described by binary relations between tokens, such as $the \leftarrow fracture$ and $severe \leftarrow fracture$. Subsequent syntactic parsing discussions centre on dependency parsing, consequently readers are referred to Jurafsky et al. [59] for further explanation of context free grammars. Hereafter parsing refers to syntactic parsing.

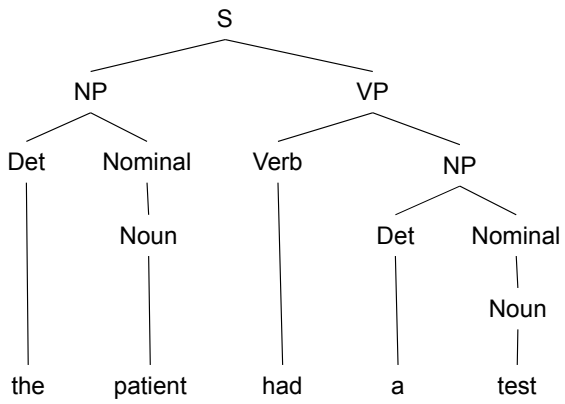

Figure 2.3: Example syntactic tree structure

2.2.5 Automated clinical coding

Stanfill et al. [117] describe automated clinical coding (ACC) as "*...computer-based approaches that transform narrative text in clinical records into structured text, which may include assignment of [unique identifiers] from standard terminologies, without human interaction*". For example, ACC could assign the unique identifier *123* to the phrase *structure of left lung*. An example of a standard terminology is Systematized Nomenclature of Medicine - Clinical Terms (SNOMED CT) [56]. SNOMED CT is described as a comprehensive clinical terminology. With respect to biomedical NLP, ACC may be used to normalize textual representations. For example, *structure of left lung* and *left lung structure* could be normalized to a single coded form.

2.2.6 Information extraction

Information extraction (IE) is the process of converting unstructured data such as biomedical text into structured information [59]. Although there are numerous IE methods reflecting numerous types of IE data and tasks, further discussion is restricted to the two methods relevant to my thesis. These methods are feature-based classification, and pattern-based extraction. Pattern-based IE in the context of my thesis refers to a simple algorithm that locates patterns in text and extracts information directly from these text chunks.

In feature-based classification, input data is modeled as features and classifiers assign labels (structured information) given features [2]. Features are measurements made on data, such as the existence (or not) of a specific key-word in a clinical document. Feature-based classifiers categorize feature models into specified categories. For example, a classifier could categorize a clinical letter as referring to patient that has delirium or as a patient without delirium. Feature-based classification may be used for IE by modeling data as features and having classifiers output specific structured information given the features.

2.2.7 Corpora

A corpus contains computer-readable text or speech [59]. Corpus content varies and may include newspaper or biomedical text. Often, a corpus is linguistically annotated. These annotations may include sentence segmentation, POS tags, syntactic parses or semantic information such as named entity roles (e.g., Canada, role=location).

Software components such as POS taggers and dependency parsers may train their linguistic models using corpora.

2.3 Biomedical NLP difficulties

Difficulties may be encountered when building and applying biomedical NLP systems. This section describes many of these difficulties. These difficulties influence the construction of my proposed biomedical NLP system. These difficulties also provide context for my research contributions. The following list summarizes the biomedical NLP difficulties described in the following sections.

Ambiguity: The uncertainty of meaning and syntax in language that often requires supplementary information to be resolved.

Character: The characteristics of biomedical text that frequently differentiate biomedical text from general text and complicate biomedical NLP.

Data: The hurdles in accessing and annotating biomedical data for system development.

Design and implementation: The issues reported when engineering a biomedical NLP system.

Diversity of language: The matter of multiple terms and phrases having identical or nearly identical semantics.

Domain knowledge: The need for biomedical NLP systems to include and understand domain knowledge.

Modifiers: The importance of processing modifiers such as certainty, quantitative, degree and temporal.

Relations: The importance of processing relations between phrases, sentences and documents.

Usability: The utility of biomedical NLP systems for users, mainly health professionals.

2.3.1 Ambiguity

Ambiguity is a difficulty in NLP [59] and manifests itself in biomedical NLP. It can be divided into two categories using Hripcsak et al.'s [53] notion of explicit and implicit vagueness. Thus, ambiguity may be categorized into explicit and implicit ambiguity. Explicit ambiguity is ambiguity created by a writer. For example, the phrase "about three weeks ago" is explicitly ambiguous due to the qualification *about*. Implicit ambiguity is ambiguity due to interpretation by a reader. Some examples of implicit ambiguity include the following:

- Abbreviations and acronyms may be ambiguous. For example, BPD could imply broncopulmonary dysplasia, borderline personality disorder, biparietal diameter, bipolar disorder, or biliopancreatic diversion [84]. It can also be difficult to differentiate between abbreviations and sentence boundaries [33, 55].

- A single word may be semantically ambiguous, such as discharge (e.g., discharge from hospital and discharge from wound) [37, 93, 111].

- Local terminology can be another cause of ambiguity [84]. For example, a facility named after a person.

- Medication names may be ambiguous when POS tagging since they may be tagged as a noun or a proper noun [105].

- Unknown words (undefined words, out-of-vocabulary items, new medical terms) are well reported as ambiguous [20, 35, 52, 95, 105]. For example,

Ambiguity is not restricted to tokens, relationships between words may be ambiguous. For example, "no acute infiltrate" may imply no infiltrate or, if there is an infiltrate, it is not acute [37]. The location of information within biomedical text may render it ambiguous. For example, "*pneumonia in the Clinical Information Section of a chest X-ray report may mean rule out pneumonia or patient has pneumonia, whereas the occurrence of pneumonia in the Diagnosis Section is not ambiguous*" [38]. Another example involves temporal relationships: last year implies 2002 uttered in 2003 and 1995 uttered in 1996 [135]. Similarly, biomedical NLP systems may have difficulty differentiating between findings and their interpretations [133].

2.3.2 Character

General English texts such as novels and news articles differ from biomedical text [16, 39, 48, 130, 135]. This difference has been described as a difference in vocabulary and grammar [16], a non-standard language model [48], and as a sublanguage following research by Zellig Harris [39].

> According to Harris, the languages of technical domains have a structure and regularity which can be observed by examining the corpora of the domains and which can be delineated so that the structure can be specified in a form suitable for computation. Whereas the general English grammar theory primarily specifies well-formed syntactic structures only, Harris' sublanguage grammar theory also incorporates domain-specic semantic information and relationships to delineate a language that is more informative than English because it reflects the subject matter and relations of a domain as well as the syntactic structure. [39]

No matter how biomedical text is described, many researchers agree that there are characteristic differences between it and general English texts.

Characteristic of biomedical text is the widespread use of abbreviations and acronyms [38, 48, 55, 84], use of Latin and Greek terminology [48], increased use of compound words (e.g., otorhinolaryngology) [70], and an unusual use of certain symbols [133] (e.g., > for implies). Furthermore, biomedical text may contain an above average use of measurement units [48], lists and enumerations [8, 48] and tabular data [133].

Tabular data are an example of a non-paragraph formatting that is characteristic of some biomedical text (e.g., clinical text) [105]. This formatting might occur as sections in a single report [38] or fragmented text [55]. Friedman et al. [38] describe this characteristic as heterogeneous formats and Xu et al. [133] as special formatting.

It is also characteristic of biomedical text to contain misspellings [20, 38, 48, 105, 110]. These may vary from a mix of lower case and capitalized medication names [105] to typographical errors such as the word *hyprtension* [38]. Inconsistencies such as typographical errors may lead to contradictory medical reports [53, 135]. Furthermore, biomedical text may be ungrammatical [47, 55]. Errors range from run-on-sentences [47, 137] and omission of punctuation [38, 133] to sentences missing subjects and verbs [39].

Biomedical text may also be terse due to information omission [39, 137]. For example, "infiltrate noted" might represent "inltrate in lung was noted by radiologist"

[39]. Other examples include the following. The phrase "the patient had spiked to 101.4" makes an implicit reference to temperature [135]. The word ruptured in the phrase "she had been ruptured times 25 1/2 hours" implies rupture of membranes [38]. Biomedical NLP systems may be required to recover omitted information in order to process text correctly.

2.3.3 Data

Biomedical data such as biomedical text allow developers to gain practical insight and experience on domain data. For example, biomedical corpora may be employed in training linguistic models for biomedical text. A difficulty in biomedical NLP is accessing and annotating biomedical data. Hurdles may include the ethical approval required to gain access to private biomedical data (e.g., patient information), the effort required to anonymize texts to protect parties, training experts (either training linguists for the medical domain or medical experts in linguistics), time and cost [38, 105]. Furthermore, each biomedical subdomain may require separate data [33].

2.3.4 Design and implementation

A first step in constructing a biomedical NLP system is determining what information to capture and its granularity [38, 53]. For example, temporal information may be important to a system's task. If temporal information is important, it may be sufficient to capture event sequences. On the other hand, exact dates and times may be required. Once information is processed, biomedical NLP systems may be required to exchange information with internal and external systems. That is, system developers must consider system interoperability and intraoperability [38, 84].

Many biomedical NLP systems are narrowly focused on particular tasks [103, 33]. Despite this narrow focus, each implementation may have specific difficulties such as finding concept negation [93]. Generalizing a systems function for the purpose of reuse is both important and difficult because generalization may lead to additional problems for the system to resolve [34]. For example, if a word is employed in two medical domains with unique semantics then a generalized biomedical NLP system must accurately distinguish between the semantics in order to process biomedical text correctly.

System design can facilitate maintenance and system improvement. For example, creating software with a manually managed grammar and semantic rules requires

significant time, background data and is likely expensive [33, 84]. Unfortunately, designing a modular biomedical NLP system is difficult [103].

2.3.5 Diversity of language

Several terms may have identical or nearly identical semantics. For example, numerous words may make reference to a single concept such as hepatic and liver [9, 111, 112]. There is also diversity in phrases [37, 84, 135]. For example, congestive heart failure, heart failure and CHF may reference the same concept. Dates and values may be expressed diversely as well, such as 01/03/02, 010302, 01032002 and Jan 3, 02. Biomedical NLP systems unable to process (e.g., normalize) language diversity risk excluding important narrative concepts.

2.3.6 Domain knowledge

Understanding biomedical text often requires an understanding of medical domain knowledge [48, 41]. Domain knowledge provides context under which the biomedical NLP system can process and interpret biomedical text. For example, domain knowledge may improve a systems ability to recover information or disambiguate phrases. Tools such as standardized terminologies provide partial domain knowledge that may be integrated into biomedical NLP systems.

2.3.7 Modifiers

Ignoring modifiers such as negation can result in an inaccurate picture of a patients chart and other biomedical text [40]. For example, there is a substantial difference between "no HIV present" and "HIV present". Modifiers may be ignored if biomedical NLP systems process only sentence segments rather than entire sentences [35].

Friedman et al. [40] suggest four types of modifiers: certainty (e.g., may represent cancer), quantitative (e.g., 2 pills), degree (e.g., severe burn), and temporal (e.g., past fracture). Modifiers may have unequal value allowing biomedical NLP systems to focus on certain modifiers while ignoring others. Negation is an important modifier [8, 17, 18, 52, 53, 55, 93]. For example, Mutalik et al. [93] found that 95 to 99% of statements within a variety of medical documents are negated.

Temporal modifiers have also received a great deal of attention [33, 36, 53, 82, 135, 136]. Temporal information may describe an intermittent event such as pain,

or a periodic event such as a prescription. Temporal information processing is often difficult. For example, Zhou et al. [136] have noted such problems as integrating domain knowledge with temporal reasoning.

2.3.8 Relations

The concept of modifiers (Section 2.3.7) may be extended to phrases with one phrase modifying another, to sentences with one sentence modifying another and to documents. For example, in the text "She fractured her right femur. The fracture is causing complications." the word *she* refers to the patient and the phrase *the fracture* refers to the fractured right femur. These complex relationships in biomedical text and a need to process them have been acknowledged [46, 135]. For example, Hahn et al. [46] argue for the importance of resolving and reasoning on referential relationships between sentences.

2.3.9 Usability

Users of biomedical NLP systems may report these systems as lacking usability. Concerns include computational performance (i.e., speed) [103], domain performance [38] and system accessibility [33] (e.g., interface or output). Perhaps the greatest concern is that systems will add error to the medical processes in which they participate [31, 84] and decrease the quality of medical care [41]. Consequently, those involved in biomedical NLP systems may need to quantify a system's error [31]. Attention should be placed on making biomedical NLP systems accurate and robust [33]. These characteristics contribute to success and acceptance by users within real environments.

2.4 Successful techniques and guidelines

The previous section described several difficulties that may be encountered when building and applying biomedical NLP systems. This section summarizes successful techniques employed in biomedical NLP systems. These techniques may be applied to address biomedical NLP difficulties when constructing a biomedical NLP system.

Stanfill et al. [117] conducted a systematic review of ACC and classification systems. They reviewed systems similar to my proposed biomedical NLP system. They concluded that "*automated coding and classification systems themselves are not generalizable, and neither are the evaluation results in the studies*". In other words,

previous work may provide poor guidance on solving my biomedical information extraction task. This suggests reviewing previous work for the purpose of providing general guidance, rather than reviewing previous work for implementation and design specifics. Furthermore, this suggests that an exhaustive review of previous work would provide little additional applicable guidance compared to a focused review of biomedical NLP systems conceived with similar information extraction objectives.

Given the above, this section summarizes Informatics for Integrating Biology and the Bedside's (i2b2) NLP shared tasks. i2b2 is an NIH-funded National Center for Biomedical Computing. Their main goal is to develop "*scalable informatics framework that will enable clinical researchers to use existing clinical data for discovery research*"[1]. NLP shared-tasks challenge NLP researchers and interested individuals to design NLP systems for processing clinical documents. Each NLP shared task has unique processing output goals. The NLP shared tasks referenced in this section are

- Smoking [126]: "*automatically determining the smoking status of patients from information found in their discharge records*"
- Obesity and comorbidities [125]: "*automatically extracting information on obesity and fifteen of its most common comorbidities from patient discharge summaries*"
- Medication information [127]: "*identication of medications, their dosages, modes (routes) of administration, frequencies, durations, and reasons for administration in discharge summaries*"
- Concepts, assertions and relations [128]: "*a concept extraction task focused on the extraction of medical concepts from patient reports; an assertion classification task focused on assigning assertion types for medical problem concepts; and a relation classification task focused on assigning relation types that hold between medical problems, tests, and treatments*"

Table 2.1 summarizes winning system characteristics across NLP shared tasks. It speaks to Stanfill et al.'s conclusions and to system variability across winning systems. That is, each task saw a variety of systems perform well using different approaches including different levels of NLP from simple pattern matching to deep parsing. Below are broad guidelines synthesized from i2b2's NLP shared tasks. These guidelines inform my biomedical NLP construction.

[1] `www.i2b2.org`

- Select NLP techniques depending on the biomedical NLP system's goals and overall design (e.g., [131] versus [107]).
  NLP techniques and components should be selected based on system goals and overall design. For example, if information is explicitly stated within biomedical documents then key words and pattern recognition may be sufficient to extract the desired information. On the other hand, if information is implied then information extraction may require deep parsing and reasoning.

- Include some sense of statement certainty (e.g., [21]).
  Clinical documents often contain a range of certainty. Processing all statements as certain fact misrepresents clinically relevant information. A test confirming lung cancer versus no lung cancer has dramatically different consequences for a patient and may equally affect software systems. Negated and non-negated statements are a simple case of statement certainty. Most systems handle some degree of negated and non-negated statements.

- Remove or segregate unneeded content (e.g., [116]).
  Some content may be unneeded given task objectives. For example, a patient identifier is likely irrelevant to a software system attempting to predict whether or not a patient is obese. Unneeded content should be removed or segregated such that sub-systems (e.g., classifiers) are not confused by irrelevant information.

- Encode medical concepts using standard or local terminologies to leverage medical domain knowledge (e.g., [73]).
  Medical knowledge is often important when processing biomedical text (e.g., medical knowledge in rule-based systems). Encoding to standard or local terminologies provides for a consistent representation of medical information. In the simplest case, standardized terminologies may be used as vocabulary and phrase thesauri. Unfortunately, encoding to standardized terminologies does not guarantee increased performance (see selecting NLP techniques above).

- Consider rule-based systems based on expert knowledge (e.g., physician knowledge) or support vector machine classifiers (SVM) when a rule-based system is infeasible (e.g., [24]).
  Most systems include a sub-system that transforms NLP output to task answers. These transformers may include human devised rules or may be auto-

mated learning and classification systems. Expert knowledge appears to benefit rule-based transformers. These latter transformers seem to perform best overall. In other words, modeling medical knowledge and intelligence has distinct performance advantages. SVM classifiers also perform well across various transformation tasks. In the absence of an expert or during early system development, SVMs are good candidates for transformation sub-systems.

- Contextualize content (e.g., [116]).
  Semantic understanding often depends on context. Contextualizing content enables sub-systems (e.g., classifiers) to better assess meaning. For example, a patient's past infection may have little bearing on whether or not the patient currently has an infection. Contextualizing information (e.g., the infection as historical) could help software systems disambiguate semantics.

- Combine systems to exploit qualities from each (e.g., [87]).
  Most systems have positive and negative qualities. For example, a classifier may perform well on one task and poorly on another. The goal in combining systems is the construction of a final system that manifest only the positive qualities of its sub-systems. An approach to combining systems is a voting scheme where each system casts a vote on a final answer. The answer with the most votes wins. In voting systems, sub-systems performing poorly are expected to be out-voted by those performing well.

2.5 Sentinel events and knowledge representation

The information to be extracted by a biomedical NLP system influences its design and implementation (see Section 2.3.4). This section describes the information to be extracted by my proposed biomedical NLP system and examines the information in context of knowledge representation. In Section 2.5.5, the knowledge representation discussion informs design of biomedical NLP systems for extracting sentinel events.

2.5.1 Sentinel events

A sentinel event is "*an unexpected occurrence involving death or serious physical or psychological injury, or the risk thereof*"[2]. For example, the occurrence of sepsis (a

[2]This definition appears in the introduction and is repeated here for readability.

Table 2.1: Summary of winning system characteristics across i2b2 NLP shared tasks

Characteristic	Smo. [126]	Ob. [125]	Medi. [127]	Con. [128]
Linguistic structures	sentences, key words	sentences, phrases, key words	sentences, phrases, key words	sentences, phrases
NLP	pattern matching, statistical NLP	pattern matching, shallow parsing	pattern matching, shallow parsing, deep parsing	pattern matching, shallow parsing, deep parsing
Uncertainty	negation	negation	negation	negation and simple uncertainty
Encoding	-	local, UMLS	UMLS	UMLS
Classification	SVM, rule-based, decision tree, kNN, AdaBoost, Okapi-BM25	SVM, rule-based, decision tree, ME	SVM, rule-based, Adaboost CRF	SVM, rule-based, ME, CRF
A system applied heterogeneous format handling	yes	yes	yes	yes
A system applied content filtering	yes	yes	yes	yes
Some hybrid systems	yes	yes	yes	yes

bacterial infection of the blood) in palliative care patients correlates with a decrease in patient survival [121, 122]. Clinical recommendations for patients and families depend on an understanding of disease trajectory. Sentinel events may provide usable information about disease trajectory. In other words, sentinel events are likely important in palliative patient care and to patient quality of life. Awareness of sentinel events may help palliative care teams provide better patient care and improve patient quality of life. For example, if sentinel events indicate that a patient has few weeks to live then an informed patient and family may better decide whether to care for the patient at home or in hospice.

I collaborated with a palliative care team in Edmonton led by Dr. Vincent Thai. Thai et al. [121, 122] are interested in sentinel events as predictors of acute palliative care patient survival. Appendix E includes the data collection instrument used by Thai et al. and consequently supplies a brief overview of the collected sentinel event data. The sentinel events selected for automated software extraction are those of interest to Dr. Thai. The sentinel events are listed below in accordance with their specification in Appendix E:

- Dyspnea
- Dyspnea at rest
- Delirium
- Brain or leptomeningeal metastases, and date diagnosed
- Sepsis in the last 4 weeks (suspected or documented), and onset date
- Infection in the last 4 weeks, and onset date
  - Chest infection, aspiration related
  - Infection site (urinary tract, intra-abdominal, skin or other)
- IV antibiotic use in the last 4 weeks, response (no, partial or complete), and onset date
- Oral antibiotic use in the last 4 weeks, response (no, partial or complete), and onset date
- Serum creatinine level and level's date

- Dysphagia in the last 2 weeks, and onset date
- Previous venous thromboembolism (VTE), and onset date
- VTE in the last 4 weeks, and onset date
- Intensive care unit (ICU) stay in the last 4 weeks and length of stay in days

The medical nature of the sentinel events are as follows. Descriptions are sourced from The Gale Encyclopedia of Medicine [68] (except for the last description):

- Dypsnea: feeling of laboured or difficult breathing that is out of proportion to the patient's level of physical activity (may be acute or chronic); feeling of increased effort or tiredness in moving the chest muscles; a panicky feeling of being smothered; sense of tightness or cramping in the chest wall
- Delirium: delirium is a state of mental confusion that develops quickly and usually fluctuates in intensity; disturbance in the normal functioning of the brain
- Brain and leptomeningeal metastases: cancer of the brain originating from another part of the body; follows the nerve pathways to the brain; most likely from a breast, colon, kidney, lung, melanoma, nasal passage and throat; meninges are membranes that enclose the brain and spinal cord
- Sepsis: bacterial infection of the blood
- Organ infection: the pathological state resulting from an invasion of the body by pathogenic microorganisms, with the infection or invasion located at an organ.
- Intravenous antibiotic use: intravenous use of a sub-group of anti-infectives used to treat a bacterial infection
- Oral antibiotic use: oral use of a sub-group of anti-infectives used to treat a bacterial infection
- Creatinine level: level of a compound produced by the body used in skeletal muscle contractions; depends on muscle mass which fluctuates little; related to renal function

- Dysphagia: swallowing disorders (difficulty swallowing); difficulty in passing food or liquid from the mouth to the stomach
- Venous thromboembolism (VTE), recent/history of: arterial embolism is a blood clot, tissue or tumor, gas bubble or other foreign body that circulates in the blood stream before it becomes stuck; deep vein thrombosis is a blood clot in a major vein (e.g., leg or pelvis)
- Intensive care unit (ICU) stay: a stay in a hospital department for patients in need of intensive care

2.5.2 Sentinel event representation in abstract

I use Friedman et al.'s [40] four types of modifiers to discuss sentinel event knowledge representation in abstract. The four modifier types are certainty (e.g., may represent cancer), quantitative (e.g., 2 pills), degree (e.g., severe burn), and temporal (e.g., past fracture). The following list indicates which modifiers apply to each sentinel event. The listing follows the sentinel event specification from Section 2.5.1. The sentinel events in the list are represented by a basic form without textual modifier descriptions.

- Dyspnea: none
- Dyspnea at rest: none
- Delirium: none
- Brain or leptomeningeal metastases: temporal
- Sepsis: temporal
- Infection: temporal
- Chest infection, aspiration related: none
- Infection site: none
- IV antibiotic use and response: degree, temporal
- Oral antibiotic use and response: degree, temporal

- Serum creatinine: quantitative, temporal
- Dysphagia: temporal
- VTE: temporal
- ICU stay: quantitative, temporal

Three modifiers (degree, temporal, quantitative) are explicitly stated in the sentinel event descriptions. *Certainty* is implicit to the sentinel events. That is, certainty is not explicitly specified (see Appendix E) but may be conveyed through context such as "likely represent a chest infection". Consequently, all modifiers are required to represent the sentinel events above.

The representation granularity differs among all modifiers. The certainty modifier's granularity depends on context. For example, text may include only three levels of certainty such as positive, possible, negative. The quantitative modifier's granularity depends on the target information. For example, half day stays in the ICU may only be represented in full days given the sentinel event, ICU stay. The degree modifier's granularity is specified as none, partial and full (e.g., response to antibiotic). The temporal modifier is as both absolute and relative. For example, an onset date is an absolute statement of day, month and year, whereas *recent* or "last four weeks" is a relative statement. In both cases, the granularity is a date specified by day, month and year.

2.5.3 SNOMED CT

Standard terminologies are a mechanism that may be used to represent biomedical information such as clinical text and sentinel events. Since Canada has adopted SNOMED CT as a standard clinical coding terminology[3], I have employed it for knowledge representation in ACC. This section covers SNOMED CT and Section 2.5.4 discusses sentinel event representation in SNOMED CT.

SNOMED CT is a product of the International Health Terminology Standards Development Organization (IHTSDO[4]). SNOMED CT is described as a comprehensive clinical terminology, with an objective of "*precisely representing clinical information across the scope of health care*" [56]. SNOMED CT contains approximately 390,000

[3] `sl.infoway-inforoute.ca/content/disppage.asp?cw_page=snomedct_e`
[4] `www.ihtsdo.org`

concepts, 1.4 million relationships and 1.1 million additional concept descriptions. SNOMED CT is a standard mechanism for representing clinical information.

SNOMED CT concepts represent medical and medically related concepts. SNOMED CT concept descriptions and relations establish a concept's semantics.

Several example descriptions are "temperament testing (procedure)", "adverse reaction to gallamine triethiodide" and "thymidine (FLT)F^18^". Descriptions may be divided into three types: fully specified name (FSN), preferred term and synonyms. A FSN is an unambiguous way to name a concept given all SNOMED CT descriptions. A preferred term is a description commonly used by health professionals. Synonyms include any description other than a FSN and preferred terms.

Relations link two concepts in a relationship. A relationship may be defining, qualifying, historical or additional. *Defining* relationships logically and semantically define concepts. The following example of *defining* relationships originates from the SNOMED CT User Guide [56]. The concept "fracture of tarsal bone (disorder)" is defined as:

- is a (subtype relation of) "fracture of foot (disorder)"
- and has a finding site (relation to the) "bone structure of tarsus (body structure)"
- and has an associated morphology (relation to the) "fracture (morphologic abnormality)"

The primary *defining* relationship between concepts is a hierarchical relationship called "is a". For example, SNOMED CT states that the concept *pyemia* is a type of "systemic infection", or that a "fractured femur" is a type of *fracture*. This relationship permits the use of *parent* (*ancestor*) and *child* (*descendants*) terminology. With respect to the example above, "fracture of tarsal bone (disorder)" is a child of "Fracture of foot (disorder)". Conversely, "fracture of foot (disorder)" is a parent of "fracture of tarsal bone (disorder)". A concept may have many parents and children. The root concept has no parents. All concepts, but the root concept itself, are descendants of the root.

Qualifying relationships refine a concept. For example, the severity of "fracture of tarsal bone (disorder)" may be qualified as *severe* or *mild*.

A pre-coordinated concept refers to a single SNOMED CT concept that represents a medical concept. A post-coordinated expression refers to the use of two or

more concepts to represent a clinical concept. If a medical concept is not represented by a pre-coordinated concept then the medical concept may be represented by a post-coordinated expression. Concepts are not arbitrarily combined to form post-coordinated expressions. Post-coordinated expressions adhere to restrictions established by the IHTSDO. These restrictions stipulate how concepts may be combined and how post-coordinated expressions may be combined with concepts and other post-coordinated expressions.

Below is an example post-coordinated expression. Concepts are represented by an identifier followed by a description, where the description is delimited by the | character (e.g., 64572001|Disease|). A concept is defined and refined by attribute-value pairs following the colon. For example, the attribute "finding site" has a value "meninges structure".

64572001|Disease|:
363698007|Finding site|=1231004|Meninges structure|,
116676008|Associated morphology|=79282002|Carcinoma, metastatic|

2.5.4 Sentinel event representation in SNOMED CT

Section 2.5.2 characterized knowledge representation for sentinel events as a base form with modifiers. This section examines how the sentinel event base form and modifiers are represented in SNOMED CT (July 2011).

Sentinel event base forms may be represented using SNOMED CT. The representations are as follows:

- Dyspnea:
  267036007|Dyspnea|
- Dyspnea at rest:
  161941007|Dyspnea at rest|
- Delirium:
  2776000|Delirium|
- Brain or leptomeningeal metastases:
  94225005|Metastasis to brain|
  or

64572001|Disease|:
363698007|Finding site|=1231004|Meninges structure|,
116676008|Associated morphology|=79282002|Carcinoma, metastatic|

- Sepsis:
  91302008|Systemic infection|

- Infection:
  40733004|Infectious disease|

  - Aspiration related chest infection:
    40733004|Infectious disease|:
    363698007|Finding site|=302551006|Entire thorax|,
    47429007|Associated with|=14766002|Aspiration|

  - Infection site (urinary tract):
    68566005|Urinary tract infectious disease|

  - Infection site (intra-abdominal):
    128070006|Infectious disease of abdomen|
    or
    40733004|Infectious disease|:
    363698007|Finding site|=361294009|Entire abdominal cavity|

  - Infection site (skin):
    108365000|Infection of skin|

  - Infection site (other):
    child of 301810000|Infection by site|
    or post-coordinated

- IV antibiotic use:
  281790008|Intravenous antibiotic therapy|

- Oral antibiotic use and :
  281791007|Oral antibiotic therapy|

- Antibiotic response (no, partial, complete):
  405177001|Medication response|
  102471002|Absence of therapeutic response|

399204005|Partial therapeutic response|
399056007|Complete therapeutic response|

- Serum creatinine level:
  365757006|Finding of serum creatinine level|

- Dysphagia:
  40739000|Dysphagia|

- VTE:
  429098002|Thromboembolism of vein|

- ICU stay:
  305351004|Admission to intensive care unit|

Several representations inadequately capture the intended semantics. Their base forms are intra-abdominal infection site and IV/oral antibiotic response. For intra-abdominal infection site, the concept "infectious disease of abdomen" is too broad in that abdomen is more general than intra-abdominal. Similarly, its post-coordinated expression (see above) could imply that the entire abdominal cavity is infected rather than some structure in the abdominal cavity. Antibiotic response's individual semantics are represented in SNOMED CT. However, there is no expression relating these semantics. For example, an expression relating these concepts could answer a questions such as "A complete therapeutic response to what?".

Contrary to base forms, most modifiers are poorly represented by SNOMED CT:

Certainty: SNOMED CT does not contain a consistent mechanism for expressing certainty. Some concepts express certainty directly. For example, "no active muscle contraction" expresses negation.

Quantitative: Quantitative values occur in unique circumstances such as "30 mg tablet". Aside from unique circumstances, SNOMED CT includes an incomplete hierarchy for encoding quantitative values. For example, Arabic values are limited to 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 17, 19, 23, 26, 29, 31, 32, 33, 38, 43, 55, 65, 68, 69, 71, 97 and 100.

Temporal: SNOMED CT represents time relative to the present as *current* or *past*. *Current* may be refined to "specified time" and *past* may be refined to "past

specified", "past unspecified" and "recent past". This coarse temporal representation is available through post-coordination. Individual concepts may also encode time. For example, the concepts "number of sexual partners in past year" and "number of sexual partners in past 5 years" encode time in years from a starting date.

Degree: Of all four modifiers, degree is best represented in SNOMED CT. Degree roughly corresponds to qualifying relationships and subcomponents of the qualifier hierarchy. For example, pain may be qualified in degree by the concept *severe*. To refine a concept in degree, the degree qualifier must exist, the degree qualifier must be related to the concept of interest and the refinement must be sanctioned by IHTSDO.

2.5.5 Knowledge representation and biomedical NLP system design

SNOMED CT is generally capable of representing sentinel event base forms but is unable to properly represent most sentinel event modifiers. This suggests that biomedical NLP systems should represent sentinel event modifiers (certainty, quantitative and temporal) in structures other than SNOMED CT. Assuming that sentinel events are described in biomedical text as base forms with modifiers then biomedical NLP systems may be required to process base forms and modifiers to accurately extract sentinel events from biomedical text.

2.6 Additional background information

This section covers additional background information relevant to my thesis. It discusses corpora and evaluation metrics used throughout the thesis to evaluated my research products.

2.6.1 Corpora

Several corpora were repeatedly employed throughout this thesis for training and evaluation. I selected these corpora because they are free and publicly available. This selection increases reproducibility and helps examine how suitable free public corpora are for biomedical NLP. The corpora are presented below.

Natural Language Toolkit-Penn Treebank and Natural Language Toolkit-Wall Street Journal

Natural Language Toolkit[5] (NLTK) is a language processing toolkit written in Python[6]. NLTK [69] includes approximately 10% of the Penn Treebank's (PTB) [75] Wall Street Journal (WSJ) newspaper text. Throughout this thesis, NLTK-PTB refers to the original NLTK sample and NLTK-WSJ refers to a version of the corpus with linguistic meta-data matching that of other corpora after normalization (see Section 2.6.1). An example sentence is "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.".

MedPost corpus

The MedPost corpus[7] was developed to train and evaluate the MedPost POS tagger [115]. The MedPost corpus contains biomedical abstracts from MEDLINE. MEDLINE is accessed through PubMed[8]. An example sentence is "The protein expression profiles were almost equivalent, but the MEK expression was slightly advanced, suggesting that the observed up-regulation of MEK was not due to that of ERK.".

Medpost is used throughout this thesis to refer to either the POS tagger or the corpus. In cases where the meaning might be ambiguous the tagger and corpus are explicitly differentiated.

Penn BioIE

Penn BioIE[9] (version 0.9) [64] contains MEDLINE/PubMed abstracts on oncology and cytochrome proteins. An example sentence is "P2-450 was, therefore, purified from isosafrole - treated DBA/2N liver microsomes having negligible amounts of contaminating P1-450 and P3-450.".

GENIA

GENIA[10] (version 1) [60] contains MEDLINE/PubMed abstracts on transcription factors in human blood cells. An example sentence is "Glucocorticoid resistance in

[5] `www.nltk.org,version2.0b8`
[6] `www.python.org`
[7] `www.ncbi.nlm.nih.gov/staff/lsmith/MedPost`
[8] `www.ncbi.nlm.nih.gov/pubmed`
[9] `bioie.ldc.upenn.edu`
[10] `www-tsujii.is.s.u-tokyo.ac.jp/GENIA/home/wiki.cgi`

the squirrel monkey is associated with overexpression of the immunophilin FKBP51.".

Normalizing

Corpora varied in annotation content (Table 2.2) and style. Corpora were normalized to render their annotation styles compatible for the purposes of machine learning and evaluation. The following describes the normalization process.

Table 2.2: Corpus annotation content.

Annotation	NLTK WSJ	MedPost	Penn BioIE	GENIA
POS tags	yes	yes	yes	yes
Syntax trees	yes	no	yes	yes

MedPost is only linguistically annotated with POS tags. Consequently, MedPost's use was restricted to tokenization (Chapter 3) and POS tagging (Chapter 4). MedPost's POS tags are based on the Penn Treebank's. A script provided in the MedPost download converted MedPost POS tags to PTB POS tags. Aside from a set of specific word-bound conversions between tag sets, MedPost's tag set is a specialization of the PTB tag set. Table 2.3 contains the mapping between POS tags.

NLTK PTB, BioIE and GENIA contained sufficient linguistic information for dependency parser training and evaluation. The Stanford parser[11] converted their natural PTB format to CoNLL-X dependency format [14]. I configured the Stanford parser using the arguments "*-basic -conllx -keepPunct*" to keep punctuation and produce basic dependency trees rather than dependency graphs. Dependency graphs include trees but are not restricted to them. The Stanford parser also created dependency labels [27]. Some parsing algorithms are unable to train on or output dependency graphs. The dependency parsers discussed in Chapter 5 assume the selected dependency tree format.

BioIE and GENIA use the PTB's tag set but differ in POS labeling. For example, "*COMMA*" is used rather than ",". I created a script that normalized the POS labels following the conversions in Table 2.4. The script also normalized bracket tokens by reversing the PTB conventions (Table 2.5).

[11] `nlp.stanford.edu/index.shtml`

Table 2.3: Conversion between MedPost POS tags and the PTB's.

MedPost	PTB	MedPost	PTB	MedPost	PTB	MedPost	PTB
VVGN	NN	JJT	JJS	VBB	VBP	VVD	VBD
CS	IN	RR	RB	VHB	VBP	VBN	VBN
CSN	IN	RRR	RBR	VDB	VBP	VHN	VBN
CST	IN	RRT	RBS	VVB	VBP	VDN	VBN
II	IN	PN	PRP	VBI	VB	VVN	VBN
DD	DT	PND	PRP	VHI	VB	VBG	VBG
DB	PDT	PNG	PRP$	VDI	VB	VHG	VBG
MC	CD	PNR	WDT	VVI	VB	VDG	VBG
GE	POS	VM	MD	VBD	VBD	VVG	VBG
VVNJ	JJ	VBZ	VBZ	VHD	VBD	VHZ	VBZ
VVGJ	JJ	VDZ	VBZ	VDD	VBD	VVZ	VBZ

Table 2.4: POS tag normalization map for BioIE, GENIA and PTB

Original	Final	Original	Final	Original	Final
$	SYM	:	COLON	-LSB-	-LRB-
#	SYM	“	LQT	-RSB-	-RRB-
LRB	-LRB-	”	RQT	-LCB-	-LRB-
RRB	-RRB-	WPP	WP$	-RCB-	-RRB-
.	PERIOD	PRPP	PRP$	,	COMMA

Table 2.5: Bracket simplification map

Original	Final	Original	Final
-LRB-	(	-RRB-	)
-LSB-	[	-RSB-	]
-LCB-	{	-RCB-	}

Characteristics

To better understand the similarities and differences between corpora, Table 2.6, Table 2.7 and Table 2.8 present several corpus statistics. Some notable characteristics are

- GENIA contains the longest and deepest parse trees,
- BioIE contains the shortest and shallowest parse trees,
- GENIA and BioIE share the greatest percentage of tokens
- and NLTK-WSJ and GENIA share the least percentage of tokens.

Corpora characteristics are particularly relevant to Chapter 5's Discussion section.

Node breadth is defined as the number of dependents per word for words having dependents. A stem is the base component of a word to which an inflectional affix is added [101]. For example, the stem of *legs* is *leg*. Token stems were computed using the Lancaster stemmer available in NLTK.

2.6.2 Evaluation metrics

Several evaluation metrics were used throughout my thesis. They are defined here:

- Accuracy: true positives / total items
- Precision: true positives / (true positives + false positives)
- Recall: true positives / (true positives + false negatives)
- F-measure: uniformly weighted harmonic mean, $hm(a_1, \ldots, a_n) = \frac{1}{1/a_1 + \ldots + 1/a_n}$ (e.g., f-measure of precision and recall is $2pr/(p+r)$)

2.7 Challenges

The previous sections have discussed a number of challenges in processing biomedical text, such as biomedical NLP difficulties (Section 2.3) or biomedical knowledge representation (Section 2.5). Subsequent sections discuss the specific challenges addressed in this thesis by my biomedical NLP and IE system. Each subsequent section (e.g., Section 2.8) will introduce a challenge, discuss related work and briefly state my

Table 2.6: Corpus characteristics

Characteristic	NLTK-WSJ	MedPost	BioIE	GENIA
Sentences	3914	6695	9165	18541
Tokens	97998	182360	207061	505174
Tokens per sentence	25.0	27.2	22.6	27.2
Unique tokens	11969	19994	18143	22548
Unique lowercase stems	7267	13795	13457	16368
Avg. tree depth per sentence	6.6	∅	5.5	7.2
Avg. node breadth	2.2	∅	2.3	2.1

Table 2.7: Shared characteristics for NLTK-PTB and MedPost

	NLTK-PTB·MedPost
Total unique	
Tokens	28369
Lowercase stems	18532
Duplicated	
Tokens	3593
Stems	2530
Verbs	810
Adverbs	203
Nouns	1432
Adjectives	536
Other	612
Duplicated as %	
Tokens	12.7
Stems	13.7
Verbs	2.9
Adverbs	0.72
Nouns	5.0
Adjectives	1.9
Other	2.2

Table 2.8: Shared characteristics for NLTK-WSJ, BioIE and GENIA

	NLTK-WSJ·BioIE	NLTK-WSJ·GENIA	BioIE·GENIA
Total unique			
Tokens	27061	31381	34416
Lowercase stems	20724	23635	29825
Duplicated			
Tokens	3049	3134	6273
Stems	2445	2220	4038
Verbs	635	857	1101
Adverbs	196	246	329
Nouns	1313	1083	3067
Adjectives	411	493	1185
Other	494	455	591
Duplicated as %			
Tokens	11.3	10.0	18.2
Stems	13.4	10.4	15.7
Verbs	2.4	2.7	3.2
Adverbs	0.72	0.78	0.96
Nouns	4.9	3.5	8.9
Adjectives	1.5	1.6	3.4
Other	1.8	1.5	1.7

contribution. Details of each contribution are presented in subsequent chapters. On occasion, subsequent sections discuss particular techniques such as maximum entropy POS tagging or support vector machines. These are discussed because my contributions are compared to these techniques or because my contributions employ these techniques.

To determine which challenges to address when building my biomedical NLP and IE system, I began with the system's textual input and considered each processing module iteratively. This follows Gregg et al.'s [44] notion of system prototyping: *Prototyping is an iterative process wherein subsequent developments are based on initial successes.*

The first module considered was sentence segmentation. My biomedical NLP system employs the Punkt [61] sentence segmentation system. It has been evaluated on different text genres and languages including scientific articles, where it performed well (e.g., overall mean error of 1.26%). Given that Punkt is an unsupervised learning approach to sentence segmentation, the Punkt system could be trained on clinical documents without the need for manual training annotations. I employed the Punkt implementation provided in NLTK [69] and was not required to retrain Punkt.

2.8 A tokenization challenge

Hassler et al. [49] report that tokenization is often perceived as a solved problem and Tomanek et al. [123] report that tokenization can be perceived as "*unsophisticated clerical work*". In contrast to this perception, tokenization is often complex. For example, Spanish and English words can be considered to flow across whitespace boundaries (e.g., sin embargo [42] and New York). An example of tokenization complexity in the biomedical domain includes biomedical substance names because they often contain special characters such as slashes or brackets [57] (e.g., *2-(2,4-Dichlorophenoxy)propionic*). The following reviews previous work on biomedical tokenization.

2.8.1 Related work

He and Kayaalp [50] applied 13 tokenizers to 78 biomedical text abstracts. Only 3 of the 13 tokenizers produced identical results and the results varied widely. Given the latter, He and Kayaalp advocate awareness of a tokenizer's details without clearly

defining or specifying which tokenizer details are important. Tokenizer details are expected to influence whether a tokenizer is well suited or adaptable to a particular language processing task. A poor choice of tokenizer is expected to cause (unintentional) information loss [50].

Tokenization strategies can vary depending on language [129, 6], software goals [57] and other criteria. Several tokenizers examined by He and Kayaalp [50] used simple rule-based tokenization methods (e.g., regular expressions). Jiang and Zhai's [57] empirical study of rule-based tokenization supports the use of rule-based tokenizers on specific texts. Although rule-based tokenization methods may perform well for specific texts, these methods appear to generalize poorly [50, 57].

Other than rule-based techniques, tokenization in the biomedical domain has been regarded as a classification task [3, 76, 123, 132]. For example, a classification-based tokenizer could assign a token-separator label to the space character.

Biomedical classification-based tokenization can be divided into two approaches: classifiers that classify textual objects as a token boundary (or not) and classifiers that reassemble primitive tokens. In other words, classifier-based tokenizers either split or join textual objects through classification. Split-join based tokenization approaches have applied a variety of machine learning methods with success.

In a *split* approach, a classifier identifies symbols such as a space or a period as token separators [3]. The approach performed well on named entity data (e.g., person, organization) and poorly on named entities in MEDLINE abstracts (see Section 2.6.1). The approach is restricted to a manually constructed set of single symbols (e.g., punctuation).

Tomanek et al. [123] trained two classifiers (*split* approach) to identify sentence and token boundaries using a corpus derived from the BioIE and GENIA corpora (see Section 2.6.1). Input text was split into sentences and sentences were split into tokens. The token-splitting classifier used preset token boundary symbols and corpus-based training to identify token boundaries.

Wrenn et al. [132] used transitional entropy and conditional probability to detect token boundaries (*split* approach). They compared their tokenization method to human specified sentence boundaries and a rule-based tokenizer that segmented sentences by whitespace. The authors acknowledge that the lack of a gold standard is the most important limitation of their work. An example of this limitation is that their method is not evaluated on whether punctuation such as a comma is indicative of a token boundary.

McDonald et al. [76] applied multi-label classification techniques to tokenization. Their classifier assigns beginning (B), inside (I) and outside (O) labels to primitive token sequences. The segments labeled with a B followed by consecutive I labels represented a single large token. This *join* approach might also be considered as *over-segment and repair* because their classifier reassembled incorrectly segmented tokens.

2.8.2 Summary

There is no single widely accepted tokenization method for English texts, including biomedical texts [123, 57, 3, 129, 50]. Previous approaches to biomedical tokenization lack guidance on how to modify and extend existing tokenizers to new domains. Furthermore, their idiosyncratic nature (e.g., discarding or keeping dashes and hyphens) complicates selection, modification and implementation. To address this challenge, I developed a novel approach for tokenizing biomedical text and provided implementation guidelines for my method (Chapter 3).

2.9 A POS tagging challenge

POS taggers are typically trained on linguistically annotated corpora created by human experts. A difficulty with POS tagging of biomedical text is accessing and annotating appropriate training corpora. Hurdles may include the ethical approval required to gain access to private biomedical text (e.g., patient information), the effort required to anonymize the texts to protect parties, training experts (either training linguists for the medical domain or medical experts in linguistics), time and cost [38, 105]. These difficulties may be magnified for languages with few corpora (e.g., [102]). Even if only a small biomedical training corpus is required, constructing such a corpus may still be difficult. An alternative to creating a specialized biomedical training corpus is to train POS taggers on corpora that differ from the tagger's target biomedical text (cross-domain tagging). Cross-domain tagging occurs when the training and test data are from different domains [62]. Prior to reviewing work related to cross-domain tagging on biomedical text, frequently applied POS tagging methods are discussed. They are transformation based, hidden Markov model and maximum entropy POS taggers.

2.9.1 Transformation based taggers

Transformation based taggers, such as the Brill tagger [13], work by successively refining assigned tags [59]. Each refinement applies a more specific transformation rule with the goal of assigning a more appropriate tag. Transformation rules may be specified or learned from templates and training corpora.

An example template is "change tag A to B when the preceding word is tagged Z". A possible transformation rule learned from this template is "change noun to verb when the previous tag is TO (word *to*)". This rule is applied in the following example drawn from Jurafsky and Martin [59]. Given the word *race* is more likely to be a noun than a verb and the two fragments "to race tomorrow" and "the race for", a transformation based tagger would first tag *race* as a noun and then apply the previous rule. This would result in *race* tagged as a verb in "to race tomorrow" and as a noun in "the race for".

Transformation-based taggers learn transformation rules by testing and selecting rules that reduce training errors, until no error reduction is possible or some threshold is met. The first step in training a transformation-based tagger is an initial tagging of the training data. This can be as simple as assigning noun to every token in the training data. During each learning iteration, a transformation rule is learned that reduces POS tagging error most. Transformation rules are assessed by testing their error-reduction affect on the training data. Error reduction may be calculated as the difference in correctly assigned tags between the data with and without applying a transformation rule. Training results in a list of transformation rules that may be applied, in order, to unseen data for POS tagging the data.

2.9.2 Hidden Markov model taggers

Hidden Markov Model (HMM) taggers, such as TnT [12], generally[12] work by assigning a tag given previous tags and the current token [59]. These probabilities are respectively named the prior probability and token probability. HMM taggers combine prior and token probabilities to generate a probability distribution over POS tags for the current token. Zero-order (unigram) HMM taggers ignore prior probabilities. First-order (bigram) HMM taggers consider one previous POS tag, and so on.

An HMM tagging model considers the POS tag as a hidden state and the token as visible output caused by the hidden state. Consequently, the token probability is

[12]In some cases, other tokens and tags may be used.

modeled as $P(t|c)$ instead of $P(c|t)$, where t is a token and c is a POS tag. Computing the most likely POS tag is done by computing $P(t|c)$ for all known POS tags and selecting the POS tag associated with highest probability (ignoring the issue of ties).

A zero-order HMM tagger assumes that each token is independent of any surrounding tokens or POS tags. This simplifying assumption results in a token probability calculated as

$$P(t|c) = \frac{P(c|t)P(t)}{P(c)} = \frac{C(c,t)}{C(c)}$$

where P is the probability given the training data and C is function that counts a condition (argument) in training data.

First-order HMM taggers calculate a POS tag probability as

$$P(t_i|c_i, c_{i-1}) = P(t_i|c_i)P(c_i|c_{i-1}) = \frac{C(c_i, t_i)C(c_{i-1}, c_i)}{C(c_i)C(c_{i-1})}$$

where i is the current token in a token sequence. The first-order HMM tagger assumes that c_{i-1} is fixed (decided). Computing the most likely POS tag is similar to the zero-order HMM tagger. Higher order HMM models follow from the first-order model. To tag an entire sequence, tag likelihoods are integrated into a dynamic programming algorithm often referred to as the Viterbi algorithm [59] (see Section 2.9.4).

2.9.3 Maximum entropy taggers

A major difference between maximum entropy (ME) taggers, such as mxpost [109], and the previous two taggers is the ME formalism [109]. ME taggers assign a tag to a token based on prior tags, prior words and future words. Implementations may use different tokens or sequence features however all ME taggers learn using ME learning and classification.

ME taggers define a history, h, as a tuple of elements (e.g., tokens and POS tags) surrounding the token to be POS tagged. The probability of the history and a POS tag is calculated for each POS tag t, with the most likely t selected. Positive model parameters, μ and α_j, contribute probability towards a POS tag. These parameters are enabled by features, f_j, that are triggered by a history and POS tag tuple. The model parameters are learned from training data. π is a normalization constant.

Formally the probability of some history and tag is

$$P(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)}$$

A feature could be

$$f_e(h,t) = \begin{cases} 1 & \text{the token to be tagged in } h \text{ ends with 'ing',} \\ 0 & \text{otherwise.} \end{cases}$$

2.9.4 Viterbi algorithm

In POS tagging, the Viterbi algorithm determines the most likely sequence of POS tags for a token sequence [59]. It is often paired with HMM or ME models to perform POS tagging.

The Viterbi algorithm maintains for each token and tag, a sequence probability and a back pointer. The back pointers mark the most likely sequence, of all possible sequences. The most likely sequence over an input (e.g., sentence) is found by locating the last token's highest probability sequence and repeatedly following back pointers to the first token.

The Viterbi algorithm computes from the first token to the last. It computes the likelihood of each POS tagged sequence ending with a token and POS tag. Assuming there are p POS tags then there are p previous sequences and $p \times p$ possibilities to compute for the current token. Probabilities may be computed by applying the equations from Section 2.9.2 or 2.9.3.

2.9.5 Pre-trained taggers

An alternative to training an existing tagger is using a pre-trained tagger. MedPost [115] is a pre-trained HMM tagger for the biomedical domain. It uses a special POS tag set and is enhanced with a 10,000 word lexicon, mostly biomedical words. The lexicon maps words and word endings to POS tags.

2.9.6 Related work

Researchers have examined whether existing taggers trained using non-biomedical corpora (e.g., newspaper text) accurately tag biomedical corpora. Their research is

summarized in tables 2.10, 2.11 and 2.12. Table 2.9 describes the acronyms used in tables 2.10, 2.11 and 2.12. Readers should refer to the corresponding research paper for specific details. Readers are also cautioned against direct comparison of results since corpora and methods often differ.

Table 2.9: Acronyms used in tables 2.10, 2.11 and 2.12

Acronym	Description
E	English
G	German
N	Norwegian
P	Portuguese
Brown	General English
CN	Clinical notes
EHR	General electronic health record data
FrAMed	German electronic health record data
GENIA	Biomedical research abstracts
HDS	Hospital discharge summaries
TL	Tagged lexicon
MAC-Morpho	Portuguese news
Mayo MED	Mayo electronic health record data
MedPost	Biomedical research abstracts
NEGRA	German news
Penn BioIE	Biomedical research abstracts
SPR	Surgical pathology reports
TB, TB-2	Penn Treebank
ws	words indicating corpus size
WSJ	Wall Street Journal text (in TB)

Table 2.10 summarizes intra-corpus tagging results. Intra-corpus is where taggers are trained and tested using a single corpus. Intra-corpus taggers average 95.3% accuracy. Accuracy is the portion of correctly tagged tokens given all tokens. All tagging methods perform well when trained and tested on the same corpus. Intra-corpus tagging accuracy closely matches inter-human POS tagging agreement and thus corpus POS tag consistency [75, 109].

Table 2.11 summarizes cross-domain tagging results. Tagging accuracy decreases by about 10% when a tagger is trained on non-biomedical corpora and applied to biomedical corpora. A notable exception is discussed by Wermter and Hahn [48]. They trained TnT and Brill on German news text and retrained these taggers with

Table 2.10: Intra-corpus results

Paper	Tagger	Lang.	Training	Testing	Accuracy
[23]	HMM	E	TB-2	TB-2	96
[23]	HMM	E	GENIA	GENIA	97
[23]	HMM	E	Mayo MED	Mayo MED	92
[55]	HMM	N	EHR (1000 ws)	EHR	79.1*
[55]	HMM	N	EHR (64000 ws)	EHR	94.5
[28]	HMM+TL	E	MedPost	MedPost	95.1
[115]	HMM+TL	E	MedPost	MedPost	97.4
[16]	Brill	E	HDS	HDS	96.9
[120]	ME	E	Fruit fly	Fruit fly	93.1
				Average	93.5
				(omit *)	95.3

biomedical text. Their cross-domain tagging accuracy differed by about 2% rather than 10%. Under both conditions, their taggers were above 90% accurate.

Table 2.11: Cross-domain results

Paper	Tagger	Lang.	Training	Testing	Accuracy
[55]	HMM	N	News	EHR	76.9
[23]	HMM	E	TB-2	Mayo MED	88
[23]	HMM	E	TB-2	GENIA	85.1
[104]	TnT	E	TB	CN	89.8
[48]	TnT	G	NEGRA	FrAMed	95.2
[16]	Brill	E	Brown, WSJ	HDS	89
[48]	Brill	G	NEGRA	FrAMed	91.9
[120]	ME	E	WSJ	GENIA	85
[120]	ME	E	WSJ	Penn BioIE	86
[120]	ME	E	WSJ	Fruit fly	88.7
[67]	ME	E	WSJ	SPR	79
[124]	ME	E	WSJ	GENIA	80.8
[102]	OpenNLP	P	MAC-Morpho	HDS	75.3
				Average	85.4

Table 2.12 summarizes tagger results of taggers retrained for the biomedical domain. Tagging accuracy often returns to intra-corpus levels upon retraining. Occasionally, retraining with a biomedical corpus may not return tagging accuracy to intra-corpus levels. In some cases, tagging accuracy decreases [23].

Existing non-biomedical corpora have been supplemented with manually derived biomedical lexicons, automatically derived biomedical lexicons, additional biomedical corpora and portions of the target biomedical corpus to return tagging accuracy to intra-corpus levels. Tateisi et al. [120] and Liu et al. [67] suggest that supplementing an existing non-biomedical corpus with a biomedical corpus improves tagging more than a lexicon. However, a supplemental biomedical lexicon can also return tagging accuracy to intra-corpus levels [102].

Causes of tagging error (CTE) may be grouped into two broad categories: lexical and syntactic. Lexical CTE is restricted to a language's vocabulary and syntactic CTE is the general arrangement and patterns of individual words in a language.

Unknown words are considered the major lexical CTE [67, 104]. Another lexical CTE is increased POS tag ambiguity [23, 67]. This occurs when a word is associated with additional tags given new training data. For example, the word *implant* may be

Table 2.12: Retrained tagger results

Paper	Tagger	Lang.	Training	Testing	Accuracy
[23]	HMM	E	TB-2, GENIA	MED	88
[23]	HMM	E	TB-2, GENIA	GENIA	97
[23]	HMM	E	TB-2, Mayo MED	Mayo MED	84.4
[88]	HMM	E	WSJ, TL	GENIA	95.8
[88]	HMM	E	WSJ, TL	MedPost	94.6
[55]	HMM	N	News, EHR (1000 ws)	EHR	84.5
[55]	HMM	N	News, EHR (64000 ws)	EHR	94.1
[67]	HMM+TL	E	MedPost	SPR	84.2
[104]	TnT	E	TB, CN	CN	94.7
[48]	TnT	G	NEGRA, TL	FrAMed	96.7
[48]	TnT	G	NEGRA, TL	FrAMed	96.8
[48]	Brill	G	NEGRA, TL	FrAMed	93.4
[48]	Brill	G	NEGRA, TL	FrAMed	93.6
[67]	ME	E	WSJ, auto-SPR	SPR	84
[67]	ME	E	WSJ, filter-SPR	SPR	92.7
[67]	ME	E	WSJ, SPR	SPR	93.9
[120]	ME	E	WSJ, Penn BioIE, GENIA	Fruit fly	96.4
[120]	ME	E	WSJ, Penn BioIE, GENIA, Fruit fly	Fruit fly	97.9
[124]	ME	E	WSJ, web	GENIA	83.7
[102]	OpenNLP	P	MAC-Morpho, TL	HDS	90
[119]	CRF	E	WSJ, MEDLINE	Penn BioIE	87.6
				Average	91.5

tagged as a verb in newspaper corpora and as a noun in biomedical corpora [23].

More researchers [88, 23, 67, 104] discuss syntactic CTE compared to lexical CTE. Syntactic CTE may be due to differences in tagging style between corpora [88] or models over-fitting training data (tri-gram model trained with TB-2 and MED) [23]. In other words, syntactic CTE is likely a greater source of cross-domain tagging error than lexical CTE.

2.9.7 Summary

Biomedical POS taggers may be trained on non-biomedical data, negatively impacting performance. An alternative to creating a specialized biomedical training corpus is cross-domain tagging. I developed an algorithm that performs better than several leading algorithms in cross-domain tagging of biomedical text (Chapter 4).

2.10 A parsing challenge

Dependency parsers have been applied to various biomedical text processing tasks. For example, Miyao et al. [90, 89] investigated the impact of various parsers and training on protein-protein interaction identification in biomedical text. My interest in dependency parsers is motivated by their transparent[13] encoding of linguistic structures and their fast processing speed relative to other parsers (Section 2.10). Processing speed has been critical to the usability and acceptance of software by health professionals (Section 2.3.9).

Dependency parsers have the potential to quickly map text input to linguistic structures for semantic understanding purposes. The effect that POS tagging errors have on two state-of-the-art dependency parsers (MST and Malt) is unknown. Consequently, I compare the effect of POS tagging errors on MST and Malt. The following subsections introduce dependency parsing, MST, Malt, and discuss previous comparisons of these two parsers.

2.10.1 Dependency parsers

Dependency parsers stem from the theoretical constructs of dependency grammar [63]. Fundamental to dependency grammar is the idea that syntactic structure is composed

[13] simple, direct

of tokens joined by asymmetrical binary relations called dependencies. A dependency relation binds a dependent token (syntactically subordinate token) with its head token (the token on which the dependent relies). For example, the phrase *malignant cells* might be represented by an edge between the dependent word *malignant* and its head *cells*, where this edge is labelled *adjective modifier*. Dependency parse structures may be used to extract text semantics (e.g., extracting the modifier severe in severe foot fracture).

Dependency parsing automatically derives a labeled or unlabeled dependency graph from an input sentence, or partial sentence [63], such as the depicted in Figure 2.4. In other words, dependency parsing maps a sentence's tokens to a dependency graph. A labelled edge in the dependency graph specifies the relationship between a head and dependent token. Most dependency parsers learn from linguistically annotated data prior to parsing, often making substantial use of POS tags assigned by POS taggers. Learning and parsing are defined as:

- "*Learning: Given a training set D of sentences (annotated with dependency graphs), induce a parsing model M that can be used to parse new sentences*" [63].
- "*Parsing: Given a parsing model M and a sentence S, derive the optimal dependency graph G for S according to M*" [63].

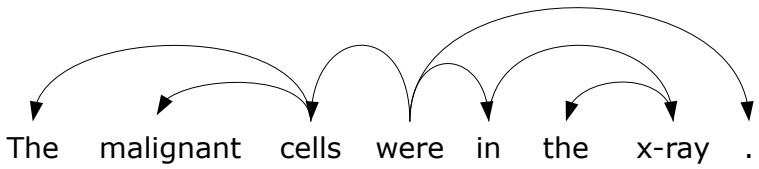

Figure 2.4: Simple dependency graph (arrow points at dependent)

Advantages attributed to dependency parsing include transparent encoding of linguistic structures, suitability for free or flexible word order languages and accurate parsing for multiple languages [63]. Dependency parsers are often theoretically faster than other[14] parsing methods [99]. Miyao et al. [90, 89] have found that dependency parsers are generally faster in practice as well.

Formal definitions are borrowed from McDonald et al. [79]. An input sentence is defined as $x = w_0, w_1, \ldots, w_n$ where w_k is a token and w_0 is a special root token. A

[14]grammar and constituent based

dependency graph, G, is defined as a set of nodes, V, and a set of labeled directed edges, E. If $(i, j, l) \in E$ then there is an edge in G from node i to node j with label $l \in L$. For unlabeled dependency graphs, l may be set to a placeholder label. Node $k \in G$ represents token w_k. Consequently, each edge may be interpreted as a syntactic relationship l between a head, w_i, and a dependent w_j. Furthermore, a dependency graph must:

- span the entire input. $E = \{0, 1, \ldots, n\}$.
- guarantee that w_0 is the graph's root. If $(i, j, l) \in E$ then $j \neq 0$.
- guarantee that no node has two head nodes. If $(i, j, l) \in E$ then for all $(i', j, l') \in E$, $i = i'$ and $l = l'$.
- have a directed path from the root to any node except the root. For any $j \in V - \{0\}$, there exist $(0, j, l) \in E$ or a non-empty sequence of nodes $i_1, i_2, \ldots, i_m \in V$ and labels $l_1, l_2, ..., l_{m+1} \in L$ such that $(0, i_1, l_1), (i_1, i_2, l_2), ..., (i_m, j, l_{m+1}) \in E$.

A further restriction assumes projective dependency graphs. The definition above is for non-projective dependency graphs. Projective dependency graphs have no crossing edges, Figure 2.4. In other words, an edge (i, j, l) is non-projective if one or more nodes between i and j is not a descendent of i. English is generally projective [81]. For example, the Penn Treebank is exclusively projective [81]. I restrict parsing to projective dependency graphs because most English is projective.

2.10.2 MST and Malt

MST and Malt are two dependency parsers[15]. MST [81], in its fastest configuration, runs in quadratic time and Malt [98] runs in linear time, with respect to input words of a sentence. Both parsers have been evaluated on multiple languages. For example, MST [81] and Malt [98] were the two best dependency parsers in the "*CoNLL-X shared task on Multilingual Dependency Parsing*" [14]. This task measured dependency parsing performance on 13 languages. MST placed first. The difference in results between the first place and second place parser were not statistically significant [14]. They also demonstrate state-of-the-art performance on English [77, 99, 79].

[15]MST and Malt are also called parser generators because they include an approach to creating a new parser based on training data, as well as a parsing method once trained.

MST

MST is a graph-based parser. Graph-based parsers build upon existing knowledge of graphs and graph algorithms. MST integrates several algorithms and their variations into its formal parsing framework. The framework is as follows. Let $s : V \times V \times L \rightarrow \Re$ be a function that scores dependency edges. Given s, a dependency graph, G, is scored as the sum of its edge scores (Equation 2.1). Parsing is equivalent to finding the highest scoring dependency graph given all possible dependency graphs, D, for some input (Equation 2.2). All possible dependency graphs are contained in a directed dense graph over the input. In this case, a directed dense graph contains all edges in $V \times V \times L$.

$$s(G = (V, E)) = \sum_{(i,j,l) \in E} s(i, j, l) \tag{2.1}$$

$$G^* = argmax_{G \in D}\ s(G) = argmax_{G \in D} \sum_{(i,j,l) \in E} s(i, j, l) \tag{2.2}$$

Selecting a best dependency graph from a directed dense graph is equivalent to finding a highest scoring directed spanning tree (maximum spanning tree). Several algorithms exist for finding a maximum spanning tree in a directed dense graph. MST's fastest configuration uses a quadratic time (relative to input tokens) algorithm to find a maximum spanning tree.[16] This configuration is hereafter referred to as MST.

Given training data, MST learns a function using large-margin models and algorithms that scores edges. To parse input, MST scores all dense graph edges and selects the highest scoring edge for each node. If the result is a tree then the algorithm stops otherwise it stores all subtrees, eliminates the arcs that are no longer possible given the subtrees and recursively applies itself to the graph's remaining cycles.

Malt

Malt is a transition-based parser. Transition-based parsers model parsing as an abstract machine that transitions between machine states, where transitions map an input to a dependency graph. Transition-based parsing defines a set of states, C, and a set of transitions T. Each state $c \in C$ defines a partially built dependency graph and each $t \in T$ is a partial function $t : C \rightarrow C$. Transition-based parsers learn a scoring function, $s : C \times T \rightarrow \Re$, that models the likelihood of transition t leading to the optimal dependency graph, given state c. Parsing finds a terminal state (a

[16] This configuration is a first order model.

dependency graph over the input) starting from an initial state (e.g., input data and no dependency graph) by selecting the best transition (Equation 2.3) for each parse state.

$$t* = argmax_{t \in T}\ s(c, t) \tag{2.3}$$

Malt includes several configurations. These configurations specify data structures, transitions and other parsing details. The English configuration uses the *stack projective* algorithm, a stack, an input buffer and three transitions. This configuration will hereafter be referred to as Malt.

Malt's start state consists of a non-empty input buffer, an empty dependency graph and a (fictitious) root token on the stack. Malt attaches all sentences and fragments to the root. An end state consists of an empty input buffer and a dependency graph.

The three pre-defined transitions change the Malt's state. For example, a *shift* transition moves a token from the input to the stack and a *left-arc* creates an arc between the next input token and the stack's top token (and removes the stack's top token). Transitions include restrictions that guarantee construction of a well-formed dependency graph. For example, a restriction would disallow an arc between a dependent token and a head token if an arc already exists between the dependent token and some other head token. Malt also uses large-margin models and algorithms (SVM) [98] to learn s.

2.10.3 Related work

This section summarizes the work comparing MST and Malt, done by McDonald, Nivre et al. [78, 99, 79]. It focuses on major differences between MST and Malt. An example of omitted content is a comparison between MST and Malt on Arabic linguistics structures.

The parsing configurations employed in my comparison differ from those employed by McDonald, Nivre et al. In their comparison, MST uses a second-order model (cubic processing time) that is know to improve parsing (e.g., less than 1% on dependencies and a larger improvement on perfectly parsed sentences) [80], and Malt handles non-projective edges. Despite these differences, the discussion below is expected to apply when discussing results (Section 5.2) because English is mostly projective and the second-order model is similar to the first-order model.

McDonald et al. [78] highlight the differences between MST and Malt. These

differences correlate with correct parsing and parsing errors. The differences are:

- Inference: MST exhaustively searches a dense graph to find the maximum scoring dependency graph. Malt greedily selects the best transition based on a trained classifier and the parser's current state.
- Training: MST trains a model to maximize a global score whereas Malt trains a classifier to choose the best local transition.
- Feature representation: MST is restricted to fewer features in its feature selection in order to make the overall parsing problem tractable. Malt may use a feature rich representation.

The parsing behaviour described in the remainder of this section results from parser differences. Malt is slightly better at parsing shorter sentences (10 tokens or less). Aside from short sentences, the two parsers commit an equivalent number of errors as sentence length grows.

Dependency arc length is the number of tokens between the dependency arc's head and dependent token. MST is better than Malt at parsing longer dependency arcs whereas Malt is better with shorter dependency arcs. MST's errors are not related to arc length.

Parser behaviour on sentence and dependency arc length is expected given that Malt makes local decisions based on a rich feature set and MST makes globally exhaustive decisions. Malt creates shorter arcs first and longer arcs last. Malt's longer arcs are influenced by errors in shorter arcs.

MST is better at parsing dependency arcs close to the root whereas Malt is better with dependency arcs far from the root. The distance of an arc from the root is the number arcs (including the arc itself) on the path between the root and the arc. Malt's focus on creating short dependency arcs implies that it creates dependency arcs furtherest from the root first. An error's influence is felt most at the root. MST's errors are evenly distributed except for dependency arcs close to and furtherest from the root. MST behaves opposite to Malt and appears to push difficult decisions further from the root.

Sibling dependency arcs are arcs that syntactically modify the same word (dependents of the same head). There is little difference between parsers with respect to sibling dependency arc errors.

Errors were separated using seven coarse POS categories: verbs, nouns, pronouns, adjectives, adverbs, adpositions and conjunctions. MST performed better in all categories except nouns and pronouns. These results and their explanation mirrored previous results and explanations. For example, verbs are close to the root whereas nouns are far from the root. Malt was expected to perform better on adpositions and adjectives but did not, with no explanation for this unexpected inconsistency.

MST performs better for root node dependents (mostly verbs) whereas Malt performs better on subject and object dependents. Malt uses previously assigned dependency arc labels in decisions. These labels are presumed to be important in disambiguating subjects and objects.

To summarize, Malt is best at correctly parsing noun phrases whereas MST is best at correctly parsing verb phrases. This reflects Malt's local and MST's global parsing styles.

2.10.4 Summary

McDonald et al. [78] and Nivre et al. [99] analyzed and compared MST and Malt's parsing errors. They did not examine, nor compare, the effect that POS tagging errors have on MST and Malt's dependency parsing. Clegg et al. [22] investigated the consequences of training NLP systems on non-biomedical corpora and applying them to biomedical corpora. They claimed that POS tagging errors cause a large portion of dependency parsing errors. Others support this result (e.g., [108, 10]). If MST and Malt parsers are applied to biomedical text-processing tasks then these parsers will likely be subject to imperfectly POS-tagged input. Consequently, I tested (Chapter 5) MST and Malt on imperfectly tagged biomedical input and compared their results.

2.11 An ACC challenge

The American Health Information Management Association view ACC as a means to make clinical coding more efficient, accurate, consistent and productive [30]. Since the need for coded data is increasing as the health care industry integrates coded data into their services [117], the need for ACC solutions is increasing as well. ACC could supplement, enhance or replace current industry wide manual coding processes.

ACC may also enhance biomedical NLP. For example, ACC may be used to normalize diverse textual representations. Careful selection of a standardized terminol-

ogy and specialized algorithms may further improve biomedical NLP. For example, SNOMED CT's characteristics such as logic-based relations (Section 2.5.3) could enable machine processable content that disambiguates ambiguous natural language input. Similarly, there may be relationships between linguistic structures and concept relationships such as prepositional forms (metastases *in* the leg) and SNOMED CT relationships such as "finding site" or "associated morphology".

The remainder of this section reviews related ACC work. It begins with related systems and follows with additional related work.

2.11.1 Related systems

This section draws on Stanfill et al.'s [117] systematic review of ACC. As part of Stanfill et al. [117] systematic review, they counted the number of publications attributed to a particular ACC system. This section discusses all systems published three or more times.

MedLEE [40] encodes narrative terms and their modifiers. In the example *enlarged spleen*, *spleen* is the main term and *enlarged* is the modifier. A coding table maps the MedLEE form to a code. The coding table is automatically built following four steps: term selection, term preparation, MedLEE parsing, table generation. Term selection and preparation reduce and clean prospective terms resulting in a terminology for use. In this terminology, several raw descriptions may encode as the same code. In other words, the terminology contains synonyms. In subsequent steps, the descriptions are parsed using MedLEE and MedLEE's parsed form is associated to each code instead of the original textual description. To automatically code a document, MedLEE matches parsed forms from the document with parsed forms in the coding table.

ONYX (SymText/SPRUS/MPLUS) [19, 20, 86] uses specialized (adapted) Bayesian networks [20] to transform terms and modifiers into coded form. A Bayesian network is a directed acyclic graph with nodes representing words or concepts and relationships between these concepts modelled by links. In a simple case, the Bayesian network captures conditional statements such as if *location=left* and *organ=lung* then codes *left lung* as having code X. These networks are manually created and their probabilities learned from manually assembled training examples.

MetaMap (MMTx) [5, 4] applies shallow parsing to extract noun phrases from text. From the extracted noun phrases, variations are computed for matching against concept descriptions. For example, *ocular complications* has variants such as *opti-*

cal complications or *ocular*. Variants are matched with UMLS (Metathesaurus) [5] descriptions and ranked based on four dimensions called centrality, variation, coverage, and cohesiveness. These dimensions rank exact matches as best, with scores decreasing as matches are less exact.

NegEx [18] is an algorithm for identifying negated statements in English biomedical text. Negex does not perform automated coding but may participate in systems that perform automated coding.

ChartIndex's (Saphire) [71, 54] automated coding component uses MMTx with specialized enhancements. Instead of parsing using MMTx, ChartIndex employs the Stanford parser to identify noun phrases and NegEx to identify negation. Thus, ChartIndex is linguistically more complex than MMTx but employs MMTx's coding (code ranking) functionality.

2.11.2 Additional related work

Bashyam et al. [7] apply an approach similar to ACC as MedLEE. Their software parses narrative phrases and concept descriptions into a computational linguistic structure called a dependency tree. Modified dependency trees are compared to code narrative descriptions as concepts. The difference between Bashyam and Taira's approach and MedLEE's is the parsing techniques and match scoring functions.

Stenzhorn et al. [118] use a natural language toolkit called OpenNLP to extract noun phrases. These are segmented into morphemes as identified by an external system. A morpheme is the smallest unit of meaning such as *-s* indicating plural [59]. Morphemes are the internal representation used when matching concept descriptions with narrative text. A major difference in their work is their encoding of Portuguese in SNOMED CT since SNOMED CT was not available in Portuguese (at the time).

MedTex [96, 97] employs a language processing framework called GATE. MedTex's GATE pipeline includes NegEx and MMTx. Rather than applying negation to any concept, MedTex restricts negation to four SNOMED CT concepts and their descendents (e.g., restrict to clinical findings).

2.11.3 Summary

Approaches to ACC may be generalized as follows. Current methods transform code descriptions and narrative text into an internal representation. Text is matched to

codes based on the similarity between the text's and the code's internal representation. Internal representations normalize raw forms and generally capture linguistic information used in matching and scoring. In other words, internal representations facilitate mapping between a variety of textual representations and account for semantics captured by linguistic structures. These methods make little or no use of the additional semantic information available through topology. Additional semantic information is expected to improve ACC accuracy. I devised a token based (non-phrase based) ACC approach that begins to exploit additional semantic information available in SNOMED CT (Chapter 6).

2.12 An IE challenge

Natural language is an important communication tool that is widely used to disseminate knowledge and data within biomedical domains (e.g., medicine) [38, 58]. Natural language enabled computer systems could read clinical documents and extract important information to better support healthcare professionals, biomedical researchers and other individuals in the biomedical domains. Under this premise, my contributions from the previous challenges are extended and applied to the task of extracting important information from clinical letters. The domain is palliative care (end-of-life care, Section 2.12.1), the clinical letters are palliative care consult letters and the important information is sentinel events (Section 2.5).

Palliative care teams could benefit from software tools that identify sentinel event information. Software tools could identify sentinel event information for the purpose of quality assurance. Quality assurance is "*the systematic monitoring and evaluation of the various aspects of a project, service, or facility to ensure that standards of quality are being met*".[17] Software could read consult letters to identify sentinel event information similar to a physician highlighting and annotating a consult letter's sentinel event information. In so doing, the software provides feedback to the palliative care team with the goal of ensuring a standard of care. For example, software could remind geographically dispersed physicians and related health care professionals to commence and adhere to appropriate care protocols. Care protocols are somewhat like algorithms for health care teams. Care protocols may state how care should take place, when and by whom [51].

[17] `www.merriam-webster.com/dictionary/quality+assurance`

Building software tools to provide quality assurance in palliative care is a step towards more complex software tools for palliative care. Software tools could automatically suggest tests (e.g., an x-ray), automatically coordinate palliative care team members or enable intelligent interfaces. For example, software could read and identify sentinel event and other information, using the information to find relevant clinical guidelines for palliative care teams or to suggest treatment plans.

Software could also facilitate secondary analysis and data mining for health research. Large quantities of health data exist in paper charts and electronic databases. Digitized charts and databases could be analyzed to discover novel health facts or provide evidence for research hypotheses. Having qualified health professionals undertake manual data analysis is often expensive and time consuming. Software is expected to be quick and semi-autonomous. There are many potential applications for software tools in palliative care.

The remainder of this section discusses palliative care in more depth, defines palliative care consult letters and describes two machine learning approaches (SVMs and decision trees) used during sentinel event information extraction.

2.12.1 Palliative care

Palliative care is "*an approach that improves the quality of life of patients and their families facing the problem associated with life-threatening illness, through the prevention and relief of suffering by means of early identification and impeccable assessment and treatment of pain and other problems, physical, psychosocial and spiritual*".[18] Below, Lanuke et al. [65] describe a patient's encounter and treatment by a palliative care team. Their description infuses a humanity and reality into the definition of palliative care.

> A 70-year-old retired man was transferred from a local hospital with increasing dyspnea and a large pneumothorax. He had idiopathic pulmonary fibrosis. ... He lived in rural Alberta, was married and four of their five children lived in the immediate vicinity. During the admission he was tearful, anxious, and depressed. ...
>
> The palliative care consultation team was asked to see the patient 4 weeks after admission. He had continuous dyspnea, coughing, chest pain, and

[18] `www.who.int/cancer/palliative/definition/en/`

asthenia, which he found intolerable. The patient agreed to a trial of oral morphine, in addition to his oxygen, in attempt to relieve his dyspnea.

He expressed a desire to have oxygen therapy discontinued, because he no longer wished to live in his present condition. He knew that the pulmonary fibrosis was not curable and that he was not improving with corticosteroids. He scored 5/10 on the numerical analogue scale for depression. The patient could remove his own oxygen but was concerned about suffering prior to loss of consciousness. ... [W]e negotiated for time to try to work through the adjustment and depression and maximize all potential treatment options prior to considering sedation. The attending physician followed the patient daily. He continued on paroxetine for his depressive symptoms and received psychosocial and emotional support from pastoral care, social work, and the palliative care team on a daily basis. The spiritual dimension of his distress was explicitly acknowledged and addressed. Pharmacy, nutrition, rehabilitation medicine, respiratory therapy, and the resident from medical ethics provided additional support. Transfer to a palliative care unit for longer term care was also discussed with the patient and his family.

... He denied ongoing depression and anxiety and expressed a fear of living rather than of dying. As the patient's asthenia increased, he continued to discuss the possible use of sedation with palliative care staff and his family. A family conference was held with the palliative care physician, the patient, his wife, and children to discuss his wish to have therapy with-drawn 2 weeks after the initial palliative care consultation. The family did accept the patient's decision to receive sedation, followed by the withdrawal of his oxygen therapy, because he could no longer tolerate living in his compromised physical state. The attending physician and unit staff were also involved in the decision and application. ... Midazolam was titrated to 12 mg/hr to achieve sedation, after which the oxygen was removed. The patient died peacefully 6 hours later.

2.12.2 Palliative care consult letters

An encounter between a palliative care team and a patient is summarized in a consult letter (see example in Appendix D). This letter captures sentinel events explicitly (e.g.,

the statement *patient has sepsis*) and implicitly (e.g., the medication list includes medication to treat sepsis such as Cefuroxime). Explicit and implicit information contribute to automated sentinel event IE.

2.12.3 Support vector machines

Support vector machines (SVM) classify data into classes, such as parsing operations in Chapter 5 or clinically important information in Chapter 7. An SVM is trained using class-data pairs. SVMs assume that data and class labels are transformed into numeric values prior to training. An SVM learns a linear separation (e.g., line) between two classes, given training data. Figure 2.5 depicts a linear separation of data. Formally, a classifier is a function $f(x) : R^d \rightarrow \Re$ and is linear when the classifier assumes the form $f(x; w, b) =< w, x > + b$ [32]. The symbol w is a weight vector and b a bias vector with identical dimension to x. The operation <> is the inner product. The remainder of this section will discuss SVM concepts without corresponding mathematical formalism since SVMs are employed as blackboxes. The libsvm library [32] provides SVM implementations and tools.

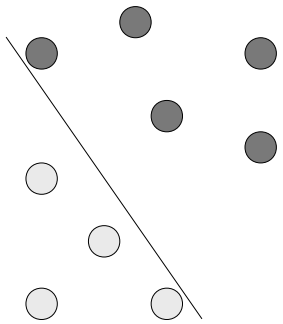

Figure 2.5: A linear separation between binary class data

Various methods exist for situations where data is separated into three or more classes [32]. For example, k classifiers are trained for k classes. Each classifier learns the separation between a class and all other classes. New data is assigned to the class that scores highest over all classifiers. In other words, binary classifiers such SVMs may be generalized to multi-class classification.

SVMs are known as large-margin classifiers because they maximize the margin (space) separating classes. Figure 2.6 illustrates a separation with larger margin, m, than Figure 2.5. It is known that classification errors are inversely proportional to margin size, thus a large margin decreases classification errors [32].

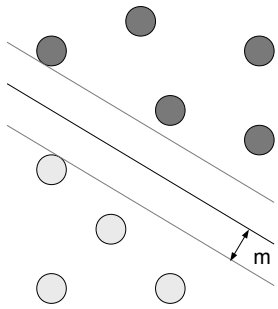

Figure 2.6: A larger margin separation between binary class data compared to Figure 2.5

Linear classifiers assume data is separable by a hyperplane such as a line in two dimensional space. Cases, such as Figure 2.7, result in misclassification errors. The "kernel trick" maps data from its original form to a new form with the expectation that the new form is linearly separable. A kernel function performs the mapping. For example, a polynomial kernel with degree two could transform Figure 2.7 to Figure 2.6. Consequently, non-linear classifiers may be constructed from linear classifiers and a kernel function. SVM's employ kernel functions and are thus capable of non-linear classification.

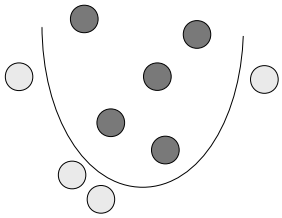

Figure 2.7: A non-linear separation of binary class data

When data is linearly non-separable, data are misclassified. SVMs handle non-separable data by introducing slack variables. Slack variables penalize a hyperplane for misclassifying data. Learning the correct separating hyperplane becomes a tradeoff between errors and margin size. The larger the margin the more likely that non-separable data are misclassified and the greater the penalization by slack variables.

The kernel and parameters that produce the best classifier given some data are found by trial and error. Parameter values are found using grid search. Grid search tests evenly distributed points in parameter space and reports the best result. Parameter space may be subsequently refined and new points tested. An example refinement is to center the new parameter space on the previous best result and decrease the spacing between points, consequently searching a localized space around the previous result. Alternate parameter optimization methods are less exhaustive and more directed. Directed methods are generally faster but may find local optimizations (stuck in local minima or maxima) rather than global optimizations.

2.12.4 Decision trees

A decision tree is a hierarchical structure whereby regions in decision space (e.g., regions representing class labels) are identified by recursive splits of decision space [2]. Decision trees consist of decision nodes and outcome leaves. Each decision node

represents a test that results in a followed branch. Each leaf represents an outcome. An outcome is computed by starting at a tree's root decision node, applying the decision node's test and following the determined branches until a leaf and outcome are reached. An example decision tree is depicted by Figure 2.8 along with its decision space.

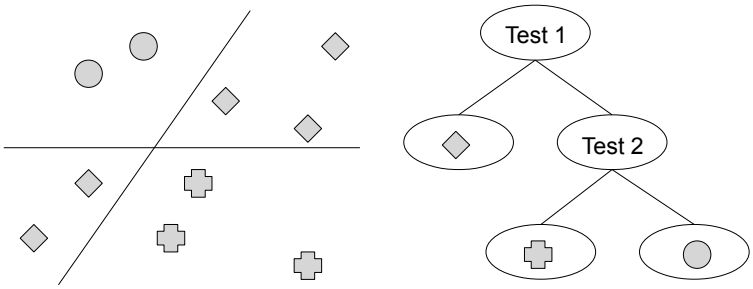

Figure 2.8: An example decision space and accompanying decision tree

When a decision tree is used for classification, the goodness of a test is quantified by an impurity measure [2]. The goal is to select tests that purify the data by splitting the data into single class groups. Thus, each decision node and test should create a better class-wise separation of the data. This may be formalized as follows (from Alpaydin [2]).

The probability for a particular class C_i at a decision node m with training instances N_m^i, representing class C_i, of total instances N_m may be expressed as

$$P(C_i|x,m) = p_m^i = \frac{N_m^i}{N_m} \tag{2.4}$$

where x represents data. In other words, the probability of a class at a node is its proportion of all instances. A node is pure if all instances of a class reach a node or do not: p_m^i for all i are either 0 or 1. A possible impurity measure is entropy:

$$I_m = -\sum_{i=1}^{K} p_m^i \; log_2 \; p_m^i \tag{2.5}$$

Building a decision tree is a greedy process. Tests (decision nodes) are greedily selected that minimize impurity. Impurity after a split is calculated as follows:

$$P(C_i|x,m,j) = p_{mj}^i = \frac{N_{mj}^i}{N_{mj}} \tag{2.6}$$

$$I_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p_{mj}^{i} \; log_2 \; p_{mj}^{i} \tag{2.7}$$

N_{mj} represents an outcome's instances. Given n outcomes, impurity after a split is a weighted sum of the outcome impurities. A test's impurity minimization is the difference between the impurity before (Equation 2.5) and after (Equation 2.7) a split.

2.13 Summary

This chapter introduced several biomedical NLP difficulties and biomedical knowledge representation issues. Biomedical NLP and IE challenges stem from these difficulties and issues. During my iterative biomedical NLP and IE system development, several challenges were addressed. The resulting research contributions are discussed in subsequent chapters. In addressing each challenge, biomedical NLP guidelines and successful techniques (Section 2.4) were considered as well as biomedical NLP difficulties (Section 2.3) and biomedical knowledge representation issues (Section 2.5). In other words, design, development and implementation decisions were influenced by the knowledge presented in this chapter.

Chapter 3

Tokenization

There is no single widely accepted tokenization method for biomedical text, nor is biomedical tokenization trivial. Previous approaches to biomedical tokenization vary. Existing biomedical tokenizers lack guidance on how to modify and extend existing tokenizers to new domains. Given the performance decreases seen when porting POS taggers to the biomedical domain (Section 2.9), tokenization methods may experience a similar performance drop. Furthermore, the idiosyncratic nature of previous approaches (e.g., discarding or keeping dashes and hyphens) further complicates selection, modication and implementation.

To address the above difficulties, I identified and completed a novel tokenizer design pattern and suggest a systematic approach to tokenizer creation. In so doing, I provide a definition of tokenization and describe software components to accompany the proposed definition. I implemented a tokenizer based on my design pattern that combines regular expressions and machine learning. My machine learning approach differs from the previous split-join classification approaches. I evaluated my approach against three other tokenizers on tokenizing biomedical text and demonstrate that high likelihood POS tagging correlates with correct tokenization. As a result, my biomedical tokenizer may leverage cross-domain POS tagging work, may eliminate training of a tokenization model and employs an approach based on natural linguistic POS tag models that is likely less idiosyncratic than current tokenizers.

3.1 Algorithm and implementation

According to Buschmann et al. [15], "*a design pattern provides a scheme for refining elements of a software system or the relationships between them. It describes a commonly-recurring structure of interacting roles that solves a general design problem within a particular context.*" In this chapter, I propose a design pattern for building a biomedical tokenizer. A biomedical tokenization design pattern may allow NLP implementors to quickly and systematically change and adapt a tokenizer for a particular context of use. My tokenizer design pattern, named the *token lattice design pattern*, defines a tokenizer's input, output and software components. The software components separate token identification and normalization from candidate token disambiguation.

3.1.1 Input and output

Current tokenizers generally compute on raw text (e.g., Wrenn et al. [132]) or sentences (e.g., McDonald et al. [76]). The token lattice design pattern restricts a tokenizer's input to raw text. If the text contains well formed sentences then it may be possible to use existing software to segment text into sentences with few errors (e.g., Punkt [61]).

A *POS-token* is defined as a token that includes a probability distribution over POS tags. The token lattice design pattern, given several sequences and some language model, restricts a tokenizer's output to the most likely POS-token sequence. In other words, tokens are restricted to POS categories such as noun or adjective. Consequently, a tokenizer must implement a defined and documented POS tag set such as the Penn Treebank's [75] and must produce the most likely sequence of POS-tagged tokens. For example, an appropriately trained tokenizer is unlikely to segment a chemical substance such as *3,4-epoxy-3-methyl-1-butyl-diphosphate* into (space delimited) "*3 , 4 epoxy 3 methyl 1 butyl diphosphate*" because such a sequence is less likely than the single token. These restrictions blur the conventional boundary between tokenizers and POS-taggers. The tokenizer could easily tag tokens during tokenization.

The notion of a POS tokenizer can be formalized as $T := (\Sigma, L_m, \Gamma)$

- Σ is a finite set of symbols called the alphabet.

- S is the set of all finite strings over Σ and $S' := S + \{\epsilon\}$, includes the empty string ϵ.
- $L_m : (E, C)$ is a language model (e.g., a probabilistic model for parsing sentences) that includes a finite set of POS tags and a finite set of tokenization contexts.
- $E(L_m) := E$ is a finite set of POS tags.
- $C(L_m) := C$ is a finite set of contexts where a context is a tuple of information specific to a tokenizer instance. For example, a context could contain the previous sentence's parse or simply the previous token.
- T_t is the set of all tuples with elements from $S \times E$. These tuples represent sequences of tagged tokens, excluding empty tokens.
- $\Gamma : C \times S' \rightarrow T_t$ is a tokenization function that given some context and text segments the text into POS tokens.

A good tokenizer is a tokenizer that selects the most likely sequence of tagged tokens for a given context, input and language model. A good tokenizer satisfies:

- $\forall c \; \epsilon \; C, s \; \epsilon \; S' \quad \Gamma(c, s) = argmax \; _{t_t \epsilon T_t} \; P(t_t | c, s, L_m)$.
- where argmax is (customarily) defined as a function that, given an expression resulting in a real-value and a set of elements, returns the subset of elements that maximize the expression's value.

3.1.2 Components

The token lattice design pattern separates a tokenizer into a token lattice and lattice constructor, a best lattice-path chooser and token transducers. Token transducers create candidate tokens from text. These candidate tokens are assembled into a token lattice by the lattice constructor. The best path (tokenization) is selected from the token lattice. These components are illustrated in Figure 3.1 and explained below.

Token lattice

Text may have multiple segmentations caused by ambiguous token boundaries. For example, the phrase "*The patient's 10mg tablet.*" (Figure 3.2) segments into eight

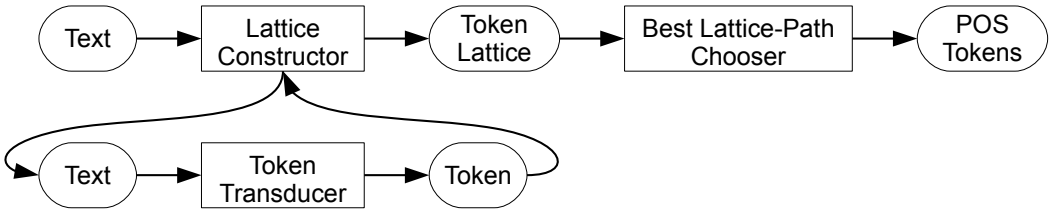

Figure 3.1: A tokenizer's components and the information flow through these components

token sequences given that *patient's*, *10mg* and *tablet.* could also be interpreted as (space delimited) "*patient 's*", "*10 mg*" and "*tablet .*". The symbols *' m* and *.* ambiguously act as token boundaries in English (e.g., *tablet.* versus *2.3*).

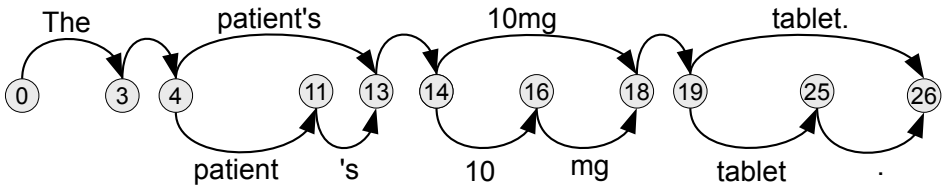

Figure 3.2: A bounded lattice representing a phrase's segmentations

A bounded lattice [26] can represent a text's segmentations. In this context, a bounded lattice is a partially ordered set of segmentations with a least and greatest element (e.g., Figure 3.2). Such a lattice is referred to as a token lattice. Conceptualizing a sentence's segmentations as a bounded lattice has been suggested previously [45, 43, 72].[1]

Token transducer

When converting text to a token lattice, it may be necessary to transform a text's candidate tokens into candidate tokens that increase the text's POS-token sequence likelihood. For example, it may be necessary to transform the token *mg* into *milligrams* to increase the POS-token likelihood of the sentence "*The patient's 10mg tablet.*". Increasing POS-token likelihood partially satisfies the tokenizer output definition, that of likely POS-token sequences.

Token transducers identify and transform a text into candidate token sequences for the token lattice. The candidate token sequences are inserted into the token lattice

[1] A token lattice had not been applied to biomedical text or used in biomedical tokenization until now.

by the lattice constructor.

A token transducer is formally defined as: $T_{transducer} := (\Sigma, L_m, \tau)$

- Σ is a finite set of symbols called the alphabet.
- S is the set of all finite strings over Σ and $S' := S + \{\epsilon\}$, includes the empty string.
- L_m is a language model (e.g., a probabilistic model for parsing sentences) that includes a finite set of tokenization contexts.
- $C(L_m) := C$ is a finite set of contexts where a context is a tuple of information specific to a tokenizer instance.
- T_s is the set of all tuples over S. These tuples represent token sequences.
- $\tau : C \times S' \rightarrow \mathbb{N}_0 \times T_s$. The transduce function returns the length of text used and a corresponding sequence of tokens.

A transduce function implementation applied to the example string "*10mg of*" could result in: $\tau_{impl}($ null, "10mg of" $) = (4,$ ("10", "milligrams")). The quantity of text used by a transducer is bounded by the length of the input, $l \in [0, \ length(s)]$, given $(l, t_s) \in \mathbb{N}_0 \times T_s$ and some $s \in S'$. A value of $(0, \emptyset)$ indicates that a transducer could not be applied.

The token transducer formalization assumes that the token transducer operates from the input string's beginning. An alternate formalization includes an index into the input string specifying the location on which to apply the transducer. Token transducer pseudocode is not supplied because token transducer code is specific to a token transducer's objective.

Lattice constructor

A lattice constructor applies every transducer to each character position in the input text, resulting in a token lattice (e.g., Figure 3.2). Given:

- Σ is a finite set of symbols called the alphabet.
- S is the set of all finite strings over Σ.
- $G := (V, E)$ is a directed graph consisting of a finite set of vertices and a finite set of labelled edges, $E \subseteq V \times S \times V$.

The token lattice G is constructed for some text $s \in S$ as follows:

- Let $L := \{i : i \in \mathbb{N}_0,\ 0 \leq i \leq length(s)\}$.
- s' is a substring of s; $s' := s[i : length(s)]$ given an $i \in L$.
- $v_i \in V$ for $i \in L$. These vertices represent a position between characters in s.
- For every slice of s and corresponding token transducer output $\tau(c,\ s') = (l,\ (t_0, \ldots, t_m))$, a path of edges, $(e_0, e_1, \ldots, e_m)$, in the token lattice, G, is constructed where the first and last vertices of the path correspond to a position between characters, $e_0[0] = v_i$ and $e_m[2] = v_{i+l}$, and an edge is associated with a token by $label(e_j) = t_j$.

Algorithms 1, 2 and 3 describe the lattice constructor in pseudocode.

Algorithm 1 The following algorithm, *constructLattice*, constructs the token lattice using token transducer output.

```
Input: context, text, transducers
  G ← createDirectedGraph()
  for index = 0 to len(text) do {nodes represent position between symbols}
    createNode( G, index )
  end for
  for index = 0 to len(text) − 1 do
    if index = 0 or nodeInDegree( G, index ) > 0 then {path through lattice must
    exist}
      for td in transducers do
        updateContext( context, G, td )
        l, token_seq = td( C, s[index:len(text)] )
        updateLattice( G, index, l, token_seq )
      end for
    end if
  end for
  cleanup( G, length(text), true )
  return G
```

Best lattice-path chooser

A best lattice-path algorithm chooses the best path (tokenization) from the token lattice. The token lattice's best path is the most likely path through the token lattice given some language model. The adapted Viterbi algorithm [42] is one possible best

Algorithm 2 The following algorithm, *updateLattice*, updates the token lattice given token transducer output (required by Algorithm 1).

```
Input: graph, index, l, sequence
  if l = 0 then {transducer could not identify token sequence}
    return
  end if
  currentNode ← index
  lastNode ← index + l
  if length(sequence) = 0 then {empty edge}
    if missingEmptyEdge( graph, currentNode, lastNode ) then
      addEmptyEdge( graph, currentNode, lastNode )
    end if
    return
  end if
  sequenceIndex ← 0
  while sequenceIndex < length(sequence) do
    token ← sequence[sequenceIndex]
    nextNode ← None
    if sequenceIndex + 1 ≥ length(sequence) then
      nextNode ← lastNode
    end if
    edge ← findEdge( graph, currentNode, nextNode, token ) {edge could exist}
    if edge = None then {edge missing, create edge}
      addEdge( graph, currentNode, nextNode, token )
    end if
    sequenceIndex ← sequenceIndex + 1
    currentNode = nextNode
  end while
```

Algorithm 3 The following algorithm, *cleanup*, removes unneeded edges from the token lattice (required by Algorithm 1).

```
Input: graph, node, isStart
  if isStart then
    beforeLastNode ← node − 1
    for n = beforeLastNode to 0 do {work backwards through the nodes}
      cleanup( graph, n, false )
    end for
  else
    if node in graph AND outDegree( graph, node ) = 0 then {lattice path does
    not reach last node}
      preds ← predecessors( graph, node )
      removeNode( graph, node )
      for p in preds do
        cleanup( graph, p, false )
      end for
    end if
  end if
```

lattice-path algorithm. The adapted Viterbi algorithm extends the Viterbi algorithm (Section 2.9.4) to consider several previous tokens rather than one previous token.

3.2 A systematic approach to creating a biomedical tokenizer

For a tokenizer that follows the token lattice design pattern, there are three principal steps in tokenizer creation. The following subsections discuss the third step since the first two steps depend on personal preference, project specifics or other matters specific to each implementation.

1. Choosing a set of documented POS tags such as the Penn Treebank's.
2. Choosing a best path selection algorithm. Implement the algorithm, if necessary.
3. Identifying the token transducers. Implement the transducers, if necessary.

3.2.1 Token transducer identification

Token transducers are dependent on the tokenizer's input language. For example, the token transducers required for English will likely differ from the token transducers required for Spanish. This section describes a systematic approach to token transducer construction. The approach, as guidelines, is as follows:

1. Select a set of documented POS tags such as the Penn Treebank's.
2. Collect text segments (e.g., sentences) from the input texts that are representative of the input texts' diversity. This may be via random sampling or another method.
3. For each text segment, identify its tokens.
    - Adhere to POS tag definitions
    - Ensure that each token corresponds to at least one POS tag.
    - Do not segment text when segmentation results in an unlikely POS-tag sequence such as segmenting "*di-trans,poly-cis-Undecaprenyl-diphosphate*"

into (space separated) "*di trans , poly cis Undecaprenyl diphosphate*". This can be captured as $P(t_t|c, s, L_m) > t$ using the introduced notation (the probability of a sequence of POS-tagged tokens given some context, input string and language model is greater than a threshold).

- Segment text when text ambiguously maps to multiple POS tags and segmenting establishes a single POS tag per token (e.g., "2.4kilograms" becomes "2.4" and "kilograms")

4. Categorize the identified tokens into token classes (e.g., '*1*, *6.2*, *10 000* and *III* are numerical).
   - Each class should be unique.
   - A categorization with few multi-class tokens is better than one with many.
5. Create a token transducer for each class of token.

3.2.2 Example identification

The following is a token transducer identification example that follows my token transducer identification guidelines (Section 3.2.1). The example uses the Penn Treebank's POS tag set, my experience as a native English speaker and the following sample descriptions. Segmentations are space delimited.

1. Entire upper dental arch (body structure)
   Segmentation: Entire upper dental arch (body structure)
2. Royal Navy - non-commissioned personnel (occupation)
   Segmentation: Royal Navy - non-commissioned personnel (occupation)
3. Primidone 50mg tablet
   Segmentation: Primidone 50 mg tablet
4. Primary Sjogren's syndrome with organ/system involvement (disorder)
   Segmentation: Primary Sjogren 's syndrome with organ or system involvement (disorder)
5. Posterior cervical spinal cord injury, without spinal injury, C1-4
   Segmentation: Posterior cervical spinal cord injury , without spinal injury , C1 to 4

6. Precorrin-3B C17-methyltransferase
   Segmentation: Precorrin-3B C17-methyltransferase

7. Salmonella III arizonae 47:k:1,5,7
   Segmentation: Salmonella III arizonae 47:k:1,5,7

Item 1 is an example of a simple segmentation.

Item 2 includes two uses of the symbol -. The first use is assigned the POS tag ":" whereas the second use, a hyphen in the token *non-commissioned*, is more difficult to assess. The hyphen could have been removed resulting in two tokens. Since *non* is a prefix and is unlikely to be associated with a particular POS tag, hyphen removal would decrease POS tag sequence likelihood. Thus, *non-commissioned* was segmented as one token.

The text *50mg* of Item 3 is segmented because segmenting establishes a single POS tag per token. The text would otherwise be a partial match to at least two POS category descriptions. For similar reasons, *C1-4* of Item 5 is segmented into multiple tokens.

The Penn Treebank specifies possessives as a separate POS category. Given this definition, the possessive *'s* is split from *Sjogren's*.

Items 4, 5, 6 and 7 are segmented to maintain likely POS tag sequences. That is, *47:k:1,5,7*, *Precorrin-3B* and *C17-methyltransferase* remain as one token, whereas *organ/system* and *C1-4* are modified.

Given these segmentations the resulting token transducers are:

- Alphabetic (dental)
- Possessive ('s)
- Independents (- ,)
- Numeric (50)
- Abbreviations (- for *to* and / for *and*)
- Functional names (C1)
- Substances (Precorrin-3B, C17-methyltransferase, 47:k:1,5,7)

3.3 Evaluation

I applied the design pattern and the token transducer identification guidelines in the creation of a tokenizer for biomedical concept descriptions and compared my tokenizer to three other tokenizers. This evaluation demonstrates that my tokenizer, design pattern and token transducer identification guidelines are equivalent to leading biomedical tokenizers.

3.3.1 Test data

Biomedical concept descriptions were extracted from SNOMED CT. SNOMED CT contains descriptions that range across the scope of health care. The concept descriptions were extracted from the January 2010 release's current concepts (as opposed to historical concepts). A SNOMED CT tokenization corpus provided a breath of biomedical tokenization cases and specifically targeted tokenization required for my ACC method, discussed in chapter 6.

I randomly selected 2781 current SNOMED CT concept descriptions[2] to create the ground truth (gold standard) tokenizations. An example concept description is "*Posterior cervical spinal cord injury, without spinal injury, C1-4*". The ground truth data contains a total of 15103 tokens. Tokenizing the MedPost corpus was not attempted since it's raw non-segmented data could not be located.

The concept descriptions were manually segmented as per token transducer identification guidelines (Section 3.2.1). A second individual also segmented the concept descriptions after reading instructions and practicing on several examples. The instructions and examples can be found in Appendix B. The second individual has a health sciences background and is not a native English speaker. Knowledge of health terminology was expected to benefit segmentation by reducing over segmenting of health terminology (e.g., substance names). Their non native-English speaker status was not expected to affect segmentation because the individual is proficient in English.

The second segmentor was provided with open-ended segmenting instructions and five examples. The segmentor read the instructions and segmented the examples, after which the preferred segmentations were presented. This was sufficient for the segmentor to conclude that segmentation "*separated units of meaning*". The seg-

[2]The goal was to select 2000 but due to a copy and paste mistake 3000 were selected. The mistake was discovered on item 2781.

mentor was encouraged to develop their own segmentation strategy given that this strategy included the two rules provided in the instructions.

An effect of my segmentation definitions and guidelines was to expand closed-class words to their normal form. For example, plus and slash separated lists were converted to regular lists (e.g., "paracetamol + caffeine" became "paracetamol and caffeine"). Similarly, dashes representing the word "to" were replaced (e.g., "C1-4" becomes "C1 to 4") and slashes representing the word "per" were replaced (e.g., "ml/g" becomes "ml per g"). Knowing that these abbreviated forms were generally absent in the training data, their expansion was to satisfy the requirement of likely POS tag sequences.

Agreement was measured with Cohen's Kappa (CK) [25] - a statistic that accounts for chance agreement. The probability of chance agreement was calculated as 0.5. CK is typically applied to categorical agreement (e.g., POS taggers agree that a word is an adjective). In this research, agreement was defined as both segmentors producing identical segmentations for a given concept description. A coin toss modeled chance agreement, where one side represents agreement and the other disagreement. For each concept description, a conceptual coin toss determined whether the segmentations would agree or not, by chance. The expected probability of chance agreement is 0.5.

There was weak preliminary agreement (CK 0.139, see Table 3.1) because descriptions ending with a parenthesized word such as "*(finding)*" were considered one segment by the second segmentor. She judged these parenthesized endings to have a single meaning and thus a single segmentation. In other words, she considered parentheses to have no explicit semantics. When the second segmentor encountered descriptions ending with several words within parentheses, she opted for segmentation consistency (not separating parentheses) rather than changing completed segmentations (changing single parenthesized words).

The parentheses were segmented from their adjoining token and agreement was recalculated. This single change resulted in a CK of 0.888. Further minor corrections to both segmentor's results, such as segmenting missed possessives, resulted in a CK of 0.916. My corrected segmentations were adopted for testing. These segmentations appear to be reasonable segmentations given a CK between 0.888 and 0.916 with another segmentor.

Table 3.1: Inter-segmentor agreement on SNOMED CT concept description segmentations

Description	Percent Agreement	Cohen's Kappa
Preliminary	56.9	0.139
Parentheses corrected	94.4	0.888
Final corrected	95.8	0.916

3.3.2 Tokenizers

I compared three tokenizers to my token-lattice tokenizer. The three tokenizers were a baseline whitespace-only tokenizer and two tokenizers selected from the list provided by He and Kayaalp [50]. I selected Specialist [100] and MedPost [115] because these tokenizers were specifically designed for biomedical text. Furthermore, He and Kayaalp reported no concerns regarding the MedPost tokenizer, unlike many of the other tokenizers they discussed.

Specialist is written in Java. Specialist considers a continuous run of alphanumeric characters bounded by white space as a token, as well as individual punctuation. Specialist over-segments and, at a latter stage, repairs the segmentation into meaningful tokens. For example, *2.4* is segmented as (space delimited) "*2 . 4*" and corrected post-tokenization. Specialist was run using the following command: *java -classpath nlpProject.jar gov/nih/nlm/nls/utils/Tokenize –inputType=freeText –tokens*.

MedPost is written in C++ and uses 33 interdependent heuristics to segment biomedical text. It segments text for further processing which includes POS tagging. MedPost's POS tag set is based on the Penn Treebank's POS tag set. MedPost was run using the command: *medpost -text*.

I implemented the adapted Viterbi algorithm [42] to choose a best-path (tokenization) from the token lattice. I created two variants of the algorithm's HMM: a zero-order and first-order HMM. I programmed the token-lattice tokenizers in Python (www.python.org) and used NLTK (www.nltk.org, version 2.0b8) [69], a natural language toolkit library.

Token-lattice tokenizers were trained on the NLTK-PTB corpus (see Section 2.6.1 for corpus details). Token-lattice tokenizers were also trained on the NLTK-PTB corpus augmented with the MedPost corpus [115]. NLTK-PTB was augmented with a random 10% of the MedPost corpus and in a separate instance all of the MedPost

corpus.

Table 3.2 summarizes my tokenizer's token transducer classes. I obtained these classes by segmenting concept descriptions by whitespace, constructing a set from these segmentations, randomly selecting 1900 items from the set and by separating each item into tokens. The manual segmentation of tokens followed my token transducer identification guidelines (Section 3.2.1) and used the Penn Treebank's POS tag set. Several items were segmented in context of their complete description because the item contained insufficient information to perform segmentation (e.g., the + in *Paracetamol + caffeine*).

Table 3.2: Token transducer classes derived from SNOMED CT concept descriptions

Class	Examples
Whitespace	
Independents	[?)
Dash and Hyphen	ACHE - Acetylcholine
Alphabetic	Does or dental
Numeric	1500 1.2 10,000 III 1/2
Possessive	's
Substances	2-chloroaniline
Serotypes	O128:NM
Abbreviations	L.H. O/E
Acronyms and initialisms	DIY
Lists	Paracetamol + caffeine
Range	C1-4
Functional names	H-987

3.3.3 Results

The tokenizers segmented the ground truth data. Ambiguous token boundaries occurred in 45.5 percent of the data. A segmentation identical to the ground truth's was considered successful and any other segmentation was considered in error. Table 3.3 summarizes the results. MedPost and my adapted Viterbi tokenizer performed best with a 92.9% and 92.4% accuracy respectively. Confidence intervals (CI) (95% confidence) were calculated using the normal approximation method of the binomial confidence interval [83].

Table 3.3: Tokenizer results

Tokenizer	Accuracy (%)	CI for 95%
Whitespace-baseline	53.9	52.0, 55.8
Specialist	47.7	45.8, 49.6
MedPost	92.9	91.9, 93.9
Adapted Viterbi, 0-order HMM	70.8	69.1, 72.5
Adapted Viterbi, 1st-order HMM (AV-1)	84.6	83.3, 85.9
AV-1 + train 10% MedPost	92.4 (5 run avg)	91.4, 93.4
AV-1 + train MedPost	92.7	91.7, 93.7

3.4 Discussion

Specialist performed poorly because of its over-segment and repair approach to tokenization. Specialist also removes symbols from the output, such as brackets, resulting in poorer performance than the baseline whitespace-only tokenizer.

MedPost's most consistent error was leaving a quantity and its unit joined rather than segmenting them. For example, MedPost would leave *10mg* as a token whereas the ground truth contained *10mg* segmented as *10* and *mg*.

The AV-1 tokenizer consistently and erroneously segmented decimal numbers. For example, it would separate *0.123* into "*0 . 123*" (space separated). An explanation could be that the training data contained an insufficient quantity of decimal numbers. Unless the tokenizer had been trained with the exact decimal number then the token was unknown to the tokenizer. Training the tokenizer using token features as well as the token itself would likely improve on AV-1's accuracy.

The AV-1 tokenizer performed as well or better than current biomedical text tokenizers (AV-1 92.4%, MedPost 92.9%). This suggests that the token-lattice design pattern and guidelines are a viable alternative to current biomedical tokenization methods.

Chaining arbitrary tokens together is unlikely to form a valid (English) sentence. Accordingly, knowing a token's POS tag indicates which POS tags and tokens are likely to occur in the token's vicinity [59]. POS-tokens inherit language characteristics that are likely to increase tokenization accuracy given that these characteristics have been successfully exploited in the past (e.g., Viterbi algorithm [59]).

POS tag sequences and training data have a significant impact on proper text tokenization. The 0-order HMM disregards transition probabilities and consequently

POS tag sequences, whereas the 1st-order HMM considers one previous state. One previous state improves tokenization by approximately 15%. A further improvement of approximately 10% is achieved by training the HMM on data that has greater resemblance to the testing data. In other words, ambiguous tokenizations can be disambiguated through POS tagging.

Inter-annotator agreement can be measured for POS tagging. This is a measure of agreement between people performing manual POS tagging of text. For example, the Penn Treebank's inter-annotator agreement for POS tagging is above 90% [75]. Since algorithms can mimic human behaviour when assiging POS tags to tokens (e.g., Brants' POS tagger [12]), tokenizers that output POS-tokens are expected to produce valid POS-token sequences and consequently mimic human performance. For example, two tokenizers adhering to Penn Treebank POS tags should segment sentences with over 90% agreement given individually successful implementations. POS-tokens should foster consistent human-like tokenization behaviour. Such behavior is expected to increase tokenizer reuse.

Dividing software into well-defined components can increase software extensibility and reuse [106]. My design pattern increases tokenizer extensibility and reusability. For example, token transducers can be reused in other token-lattice tokenizers. As an example of extensibility, consider applying a token-lattice tokenizer to new text. This should consist of identifying the new text's token transducers, including these transducers in the existing tokenizer and possibly training the tokenizer with additional data.

3.5 Summary

This chapter discussed the token lattice design pattern and the associated token transducer identification guidelines. The token lattice design pattern addresses a tokenizer's input, output and internal components. The components are a token lattice and lattice constructor, a best lattice-path chooser and token transducers. The token lattice design pattern provides a framework where biomedical text processing components such as named entity recognizers can interact cooperatively (as token transducers). The tokenization approach differs from previous split-join classification approaches. A tokenizer implementing the token lattice design pattern and guidelines was evaluated as equivalent to a leading biomedical tokenizer. Given the results, the token lattice design pattern is a viable approach to tokenization.

The results also demonstrate that ambiguous tokenizations can be disambiguated through POS tagging. In doing so, POS tag sequences and training data have a significant impact on proper text tokenization. If accurate POS tagging may be achieved on biomedical text (Section 2.9) then accurate tokenization may be achieved as well. Furthermore, the token lattice design pattern and guidelines provide a systematic recipe for tokenizer creation, result in an extensible tokenizer (handles new tokens), and are expected to produce consistent token output across differing tokenizer implementations. All of these latter qualities address biomedical NLP difficulties (*character* and *design and implementation*) and validate the use of my tokenizer for my biomedical NLP system.

Discussion has been restricted to the biomedical domain since the token lattice design pattern has been evaluated solely in this domain. However, the token lattice design pattern is expected to apply to domains other than the biomedical domain. The next chapter discusses a POS tagging algorithm that processes tokens such as those supplied by the token lattice design pattern tokenizer.

Chapter 4

Cross-domain Part-of-speech Tagging

In general, training POS taggers on non-biomedical corpora and applying these trained taggers to biomedical corpora results approximately in a 10% decrease in tagging accuracy (Section 2.9). Even in cases where a tagger is trained on one biomedical corpus and applied to a different biomedical corpus, a decrease in tagging accuracy can occur [23]. In situations such as this thesis work where linguistically annotating a training corpus is infeasible or difficult (e.g., cost), it would be beneficial to develop POS tagging methods that perform well across differing corpora. It is desirable for my POS tagger to perform well in cross domain tagging since the free public training corpora differ from the target palliative care consult letters.

I developed a POS tagger that is more accurate than three POS taggers when trained on a non-biomedical corpus and evaluated on biomedical corpora. I achieved this statistically significant increase in tagging accuracy by ignoring previously assigned POS tags (except in two verb cases) and restricting the tagger's scope to the current token, previous token and following token. This differs from the three other POS taggers that make significant use of POS tags and generally consider POS tags over an entire sentence through dynamic programming methods. In other words, ignoring generalization and focusing on detail improves cross-domain tagging. Intuitively one might expect the contrary, that generalized models would bridge cross-domain corpora best. Given my results and analysis, I suggest how to reduce the difference in accuracy observed when training and application corpora differ across domains. In particular, my results suggest that biomedical POS tagging algorithms

may be modified to improve their robustness.

4.1 A token centric tagger

This section describes my token centric POS tagger, TcT. TcT estimates a tag's likelihood for a token by combining sequence probabilities (probabilities given previous and future words) and token probabilities (probabilities based on the current token). This is similar to HMM and ME taggers. The two main differences between TcT and HMM and ME taggers is that TcT does not use dynamic programming methods and focuses on tokens rather than POS tags to estimate tag likelihood. Consequently, TcT runs in linear time relative to the number of tokens.

TcT calculates token probabilities using token features and a Naive Bayes classifier [59]. Naive Bayes classifiers apply Bayes' theorem (Equation 4.1) to calculate the most likely classification label or label distribution given feature values (e.g., POS tags). I selected a Naive Bayes classifier because it is conceptually simple, computationally fast, applicable to diverse data sets and classifies well [29].

A Naive Bayes classifier's characteristics may be particularly important as conveyed by the following example of physicians at a busy hospital. If a software system with POS tagging is to help physicians with patient care, then the software system may be required to respond immediately to physician queries due to user needs or critical patient needs. Physician queries may span diverse data from lab results to clinical notes. Software systems applicable to diverse data sets are likely beneficial. Perhaps the most important characteristic in a clinical environment is software system correctness. Physicians will not employ software systems in patient care if these systems provide incorrect answers and place patients at risk. Naive Bayes classifiers are potentially well suited to clinical environments or other similar biomedical environments.

A Naive Bayes classifier is computationally fast because it generally learns by seeing each feature set and associated label once and classifies by considering each label and feature-value pair once. In the case of POS tagging, a Naive Bayes classifier sees each token's features and tag once while learning and classifies by considering each POS tag and token feature-value pairs once. Domingos and Pazzani [29] present evidence supporting robustness of Naive Bayes classifiers and their applicability to diverse data sets. In particular, Naive Bayes classifiers generally perform well even when classification features have some degree of dependence. This contradicts the

assumption that Naive Bayes classifiers require independent features to perform well and partially explains why Naive Bayes classifiers are applicable to diverse data sets.

Given some POS tag, C, and independent features $F_1, \ldots, F_n$ (e.g., token features), Bayes Theorem (Equation 4.1) [29] restates a POS tag's probability given features as the probability of the features given the POS tag and the probability of the POS tag itself (normalized by the probability of the features). Naive Bayes classifiers learn by computing $P(C)$, $P(F|C)$ and $P(F)$ from training data. To classify features, a Naive Bayes classifier computes the right-most expression of Equation 4.1 for each POS tag. The selected POS tag has the greatest probability.

$$P(C|F_1, \ldots, F_n) = \frac{P(C)P(F_1, \ldots, F_n|C)}{P(F_1, \ldots, F_n)} = P(C)\prod_{i=1}^{n}\frac{P(F_i|C)}{P(F_i)} \tag{4.1}$$

My Naive Bayes classifier implementation differs from a typical implementation (e.g., [29]) to account for infinite features. Words are an example of an infinite feature since new words are created regularly.

A Naive Bayes classifier assigns a probability of 0 to each POS tag given an unknown feature value such as a new token. This occurs because Naive Bayes classifiers rely on $P(F|C)$ which is 0 if the feature value was not seen in training. The consequence of this behavior on tagging is erroneous tag likelihoods and incorrect tagging. My Naive Bayes classifier uses log probabilities and represents the $log_2(0)$ as a value close to zero. Reformulating Equation 4.1 using log probabilities results in Equation 4.2. A value close to zero in Equation 4.2 is essentially ignored. This is equivalent to ignoring features that were not seen in training. In other words, our Naive Bayes classifier handles infinite features and their unknown feature values by ignoring them.

$$log(P(C|F_1, \ldots, F_n)) = log(P(C)) + \sum_{i=1}^{n} log(P(F_i|C)) - log(P(F_i)) \tag{4.2}$$

TcT uses my Naive Bayes classifier implementation to estimate a probability distribution over POS tags for tokens, using the following token features:

1. Numeric - the token represents a number.
2. Capitalized - the token is capitalized.
3. Closed class word - the token is a known closed class word (defined below).
4. Suffix - the feature value is the token's suffix.

5. Lower case token - the token's lower case string.

The first two features were implemented using regular expressions. The numeric feature identifies numeric values which are typically tagged as noun or number. An advantage of the numeric feature is that it generalizes over all numbers. The capitalization feature generally identifies proper nouns and tags that occur at the beginning of sentences. Capitalization has been useful previously in determining a word's POS category [11] and semantic category in biomedical text [61].

Feature three is a word from a list of closed class words or *None*. Closed class words such as pronouns (e.g., he, all) or prepositions (e.g., between, onto) rarely change as a language evolves [1]. Feature three biases the Naive Bayes classifier into quickly adopting POS tags from training examples rather than relying on other features.

Suffixes (word endings) suggest the word's POS because English employs an affixational process that forms new words with a category or meaning distinct form the word's base form [101]. For example, *-s* in *legs* suggests plural[1] or *-ment* in *treatment* suggests a noun. A token's suffix was approximated and extracted using the Lancaster stemmer provided in NLTK [69]. NLTK is a natural language processing toolkit written in Python. If a suffix is longer than three characters then only the last three characters are used.

Feature five is the token's lower case string. This feature also biases the Naive Bayes classifier into adopting POS tags from training examples rather than relying on other features such as capitalization.

The token features were selected because they linguistically determine POS tags and bias tag assigned on tokens. Other token features and methods equivalent to Naive Bayes may prove more successful as further research is conducted.

Sequence probabilities were estimated based on token collocation. Collocation describes a token's influence on its surrounding tokens [59]. Statistically, collocation describes a significant difference from chance in sequential word pairings. For example, a person is considered tall rather than high when describing their height (tall person, not high person). Tea is thought to be strong rather than powerful (strong tea, not powerful tea). For tagging, a token is associated with its collocated POS tags rather than tokens. For example, the word *the* is often followed by an adjective or noun. TcT considers one collocated token to each side of the current token to be tagged.

[1]POS categories such as the Penn Treebank's include plural as a separate category. For example, *NN* tag for nouns and *NNS* for plural nouns.

I chose to model sequence probabilities using collocation because I conceptualized closed class words as anchors between small sequences of potentially unknown tokens, and I wanted to exclude POS tag feedback errors. Taggers often use previously assigned POS tags to help assign the current POS tag. In cases where these taggers have assigned incorrect tags, the tagging errors are fed back into the tagging system. TcT's input is restricted to tokens to avoid such feedback errors.

Previous research suggests that taggers over-fit sequence data leading to a decrease in cross corpus performance. I employed additive smoothing (Laplace smoothing) [59] to avoid over fitting sequence data. Additive smoothing modifies a categorical estimator by adding some equal probability to all categories. TcT punishes (below average) distribution uniformity by computing the average sample size and adding this quantity to all distributions.

4.1.1 Formal description

The main concepts described above are summarized in Equation 4.3, where t_i is a token and i is the token's position within the input stream, P is a conditional probability, LP is a conditional probability with smoothing and F_j is a function that maps t_i to a feature value. Equation 4.3 computes a probability distribution over POS tags in S, given token position i.

$$\forall C \epsilon S, Prob(C) = LP(C|t_{i-1}) * P(C|F_1(t_i), \ldots, F_n(t_i)) * LP(C|t_{i+1}) \tag{4.3}$$

4.1.2 Minor enhancements

The learned association between tokens with digits such as numbers and identifiers (e.g., *T3*) and POS tags does not generalize well. For example, the token *1* differs from *2* yet these often assume the same POS tag. Similarly, the identifiers *T1* and *T2* often assume the same POS tag. To generalize over such tokens, numbers are replaced[2] with an abstract number (e.g., $< num >$) and digits in identifiers are replaced with zero (e.g., *T1* becomes *T0*).

A common difficulty when tagging is assigning the correct POS tag when a token is associated with multiple tags. TcT has the most difficulty with verb-related POS tags because verbs are often preceded by nouns. Since there are a large number of

[2]Replaced in features and not in text

nouns the noun-verb sequence is often unknown to the tagger. To partially correct such errors, TcT punishes tags for base form verbs and past participle verbs. Infinitive form verbs are confused with the non-third person singular forms (e.g., *to write* confused with *we write*) and past participle verbs are confused with adjectives (e.g., *he has written* confused with *the written word*). The tags representing these verbs are punished when their conditional probability given the previous tag is less than their probability given the entire learned text. That is, when $P(VBTag|PreviousTag)/P(VBTag) < 1.0$ then punish $VBTag$ by this computed ratio.

4.2 Evaluation

I compared cross-domain and intra-biomedical domain POS tagger accuracy by training several taggers on one corpus and applying them to another. Taggers were compared on tagging accuracy because this metric has been employed in previous research, providing a context in which to interpret my results, and this metric is representative of proper tagging behavior.

I also compared TcT to MedPost [115] on the MedPost corpus. This comparison positions TcT and my results relative to a specialized biomedical tagger. The MedPost tagger was evaluated using 1000 reserved sentences. Similar to the original evaluation, I set aside 1000 random sentences for testing during each tagging run and trained on the remaining sentences.

Since tagging speed may be a concern in medical environments, I recorded the duration of time required to tag the MedPost corpus (excludes training time). Brill, TnT and TcT are written in Python whereas mxpost is written in Java. To exclude differences in speed due to programming languages and runtime environments, I compared only the tagging methods written in Python. These were run on an Apple XServe with two 2.26 GHz Quad-Core Intel Xeon processors and 12 GB of ram. Tagging was restricted to one processor core such that times were not improved through parallelization.

I compared TcT to Brill's tagger [13], TnT [12] and mxpost (ME tagger) [109]. Brill's tagger and TnT are available in NLTK's tagging demos. The mxpost implementation was downloaded from `www.cis.upenn.edu/~dbikel/software.html`. The baseline tagger assigns the most likely tag to each token based on the likelihoods learned in training. For unknown tokens, it assigns the most likely tag given the

entire training corpora (e.g., noun).

4.2.1 Corpora

I employed one non-biomedical corpus (variants NLTK-PTB and NLTK-WSJ) and three biomedical corpora (MedPost, BioIE, GENIA). There existed two training and testing scenarios. The first was to train using NLTK-PTB and test on MedPost. The second scenario was to train using NLTK-WSJ and test on BioIE and GENIA. A detailed description of these corpora is found in Section 2.6.1.

4.2.2 Results

TcT was most accurate in cross-domain tagging and mxpost was most accurate in intra-domain cross-corpus tagging (for complete results see Table 4.1). Confidence intervals for 95% confidence were calculated using the normal approximation method of the binomial confidence interval [83]. All confidence intervals are less than ± 0.002, or 0.2% accuracy. Non-intersecting confidence intervals indicate statistical significance. If the difference in POS tagging accuracy is greater than 0.4% then there is statistical significance at $p \leq 0.05$.

TcT is statistically equivalent to MedPost (Table 4.2). For the MedPost evaluation, confidence intervals for 95% confidence are less than ± 0.012, or 1.2% accuracy.

I evaluated tagging speeds for all Python based taggers since speed could be a concern in medical environments (Table 4.3). The Brill tagger was fastest at assigning POS tags and TcT followed in speed. TnT was significantly slower.

Table 4.1: Tagging accuracy

Train	Test	TcT	mxpost	TnT	Brill	Baseline
NLTK-PTB	MedPost	$\star$ 81.7	78.2	78.2	77.5	76.6
NLTK-WSJ	BioIE	$\star$ 84.6	82.1	77.2	76.4	75.3
NLTK-WSJ	GENIA	$\star$ 85.4	82.6	83.2	82.2	81.5
GENIA	BioIE	84.8	$\star$ 87.0	84.8	84.8	82.8
BioIE	GENIA	92.8	$\star$ 95.1	92.4	92.2	89.7
	Avg.	85.9	85.0	83.2	82.6	81.2

Table 4.2: Tagging accuracy on the MedPost corpus

Tagger	Accuracy
MedPost	96.9
TcT (3 runs)	96.7, 96.8, 96.6

Table 4.3: Tagging duration on the MedPost copus

Tagger	Duration (hour:min:sec)
Brill	0h:0m:10s
TnT	1h:1m:21s
TcT	0h:4m:09s

4.3 Discussion

TcT focuses on recognizing each token in context of the preceding and following tokens rather than focusing on the POS tag sequence. This is accomplished by computing a probability distribution over the POS tags for the current token and modifying the distribution's probabilities given the predictive power of the preceding and following tokens. This contrasts with other methods that use the two previous POS tags to help predict the current POS tag. For example, TnT uses a dynamic programming algorithm to select the most likely sequence of POS tags for a sequence of tokens.

There are several possible reasons for TcT's strategy being more effective in cross-domain tagging. Training and testing corpora may differ significantly in POS tag sequence probabilities decreasing tagging accuracy for taggers that place greater weight on POS tag sequences. Focusing on words may allow TcT to recover more quickly from errors. With a focus on POS tag sequences an incorrect POS tag is likely to affect subsequent tags and overall tagging accuracy. Another possible reason for TcT's effectiveness is its capacity to predict a token's POS tag from its features. Taggers missing this capacity may be at a significant disadvantage. Given my experience, it is unlikely that a single reason underlies the difference in tagging accuracy between TcT and those it was compared to.

4.3.1 Difference by example

I illustrate the difference in tagging between TcT and mxpost by presenting two examples where TcT assigns the correct tag and mxpost does not. The examples reinforce the value of omitting recently assigned POS tags during cross-domain parsing. The examples originate from the run where TcT and mxpost were trained on NLTK-WSJ and tested on GENIA.

In the first example, TcT correctly assigns the comparative adjective tag (JJR) to *stronger*, rather than the proper noun tag (NNP). The training data suggest otherwise by associating only the correct JJR tag to *stronger* five times. However, the probability of starting a sentence with JJR is 0.003 whereas NNP is 0.198. The POS tag probability likely dominates.

- Correct (TcT): Stronger/JJR c-myc/NNP mRNA/NNP expression/NN was/VBD ...
- Incorrect (mxpost): Stronger/NNP c-myc/NNP mRNA/NNP expression/NN was/VBD ...

The second example sees TcT correctly categorize *FKBP51* as a noun (NN) and *13-fold* as an adverb (RB), rather than a number (CD) and an adjective (JJ) respectively. The feature "token contains a number" within mxpost likely dominates when mxpost incorrectly assigns CD to *FKBP51*. This tagging error results in the tag sequence CD-VBD which predicts an RB with 0 probability, given the training data. Again, the POS tag probability likely dominates in the case of *13-fold*. This is an example of an incorrect POS tag affecting subsequent tags and overall tagging accuracy.

- Correct (TcT): The/DT immunophilin/NN FKBP51/NN was/VBD 13-fold/RB higher/JJR in/IN ...
- Incorrect (mxpost): The/DT immunophilin/NN FKBP51/CD was/VBD 13-fold/JJ higher/JJR in/IN ...

4.3.2 Coverage metrics

To better understand my results and cross-domain POS tagging, I compiled coverage metrics. I measured how often tokens and POS tag sequences learned in a training

corpus reoccurred in a testing corpus – how the training corpus covered the testing corpus. For example, I collected a set of tokens given a training corpus and separated a testing corpus' tokens into those that had been seen in training and those that had not. I computed coverage for tokens, POS tag trigrams (sequences of three POS tags) and token/tag bigrams (sequences of two token and associated tag pairs). These coverage metrics are meant to model how suitable a training corpus is for a testing corpus and which training corpus characteristics are similar to the testing corpus'.

The NLTK implementations of the TnT and Brill tagger have no default mechanism for handling unknown words (e.g., computing POS tags from token features). Unknown words are words that exist in the test corpus but not in the training corpus. The TnT and Brill taggers routinely performed worse than TcT and mxpost, two taggers that handle unknown words. Taggers unable to handle unknown words are completely dependent on learned POS tag sequences to assign correct POS tags to unknown words. The latter dependency is supported by TnT and Brill's improved cross-domain performance on GENIA relative to MedPost and BioIE. NLTK-PTB/WSJ's POS trigram coverage of GENIA is about 10% higher than its POS trigram coverage of MedPost and BioIE, whereas the token coverage is similar for all three corpora (Table 4.4).

Given the coverage metrics in Table 4.4, low token coverage in cross-domain tagging appears to be the dominant factor in decreasing tagging accuracy. Mxpost seems more susceptible to low token coverage than TcT. The strongest evidence for this claim derives from the instances where GENIA and BioIE are the test corpus. In these instances, the POS tag trigram coverage is similar for both training corpora whereas token coverage differs more substantially, as does tagging performance. In other words, POS tag sequence coverage appears to have less effect on tagging accuracy than token coverage.

An expected general trend is that better coverage results in better tagging performance. Within this general trend, it appears that token coverage produces the greatest performance increase and that POS sequence coverage is required to augment performance further. This implies that measuring token coverage between training and application data should act as a rough gauge on tagger performance.

Lexical CTE appears to have more effect on tagging accuracy than originally considered by previous biomedical research work. My results do not imply that a lexicon is sufficient for returning tagging performance to intra-corpus levels. Instead, it appears that additional training on a lexicon may be an effective preliminary step

Table 4.4: Training corpus coverage of test corpora as percentages

Train	Test	Tokens	POS Trigrams	Token/Tag Bigrams
NLTK-PTB	MedPost	64.9	83.7	17.6
NLTK-WSJ	BioIE	63.5	86.8	17.6
NLTK-WSJ	GENIA	65.6	95.0	19.4
GENIA	BioIE	83.5	89.4	43.9
BioIE	GENIA	87.7	98.7	48.5

in returning tagging performance to intra-corpus levels and that a domain corpus is required for high accuracy tagging. This supports previous work by, for example, Liu et al. [67]. When discussing POS tagger retraining, they noted that "the highest gain in accuracy was obtained when [they] selected sentences that contained the most frequent unknown words". On the other hand, my results differ from previous work in that I demonstrate the value in investigating new or modified POS tagging solutions to overcome the performance gap when cross-domain tagging.

4.3.3 Effect of errors

The cross-domain tagging differences between TcT and the next best tagger are between 2 and 3 percent (Table 4.1). Manning [74] argues that a 3 percent difference in tagging between 97% and 100% is relevant. Performance differences in tagging accuracy for accuracies between 70 and 90 percent may be even more relevant. For example, algorithms that use several POS tags as input may have sufficiently correct input (e.g., between 97 and 100 percent) that output is unaffected. In contrast, these algorithms may improve significantly when input errors decrease for noisy inputs (e.g., 77 to 80 percent). The relevance of a 2 to 3 percent improvement in tagging accuracy will depend on how the assigned POS tags are subsequently used.

As an example of the effect of POS tagging errors, I compared dependency parsing on perfectly tagged corpora to dependency parsing on imperfectly tagged corpora. I examined the best parser from the CoNLL-X challenge [14] – MST [81]. I selected TnT to generate the imperfect tagging because it is an accepted POS tagging technique and was pragmatically simple to employ in this experiment. Chapter 5 contains a more complete treatment of the effect of tagging errors on dependency parsing.

I trained and evaluated MST similarly to how I trained and evaluated the taggers. I trained MST using the same setup reported to have successfully parsed section 23

of the PTB. In particular, I passed the arguments "training-k:5 loss-type:nopunc decode-type:proj" to MST.

The CoNLL-X challenge evaluated parsers using labeled attachment score (LAS). LAS is the percentage of tokens that have a correctly predicted head and dependency label. McDonald, Crammer and Pereira [77] report unlabeled attachment scores (UAS). UAS is similar to LAS but excludes the dependency label. I adopt UAS and LAS to evaluate parsing accuracy. I also report the percentage of perfect sentences (PS). A parser outputs a perfect sentence if the sentence has a perfect LAS.

MST's dependency parsing results are found in Table 4.5. Confidence intervals for 95% confidence were calculated using the normal approximation method of the binomial confidence interval [83]. All UAS and LAS confidence intervals are less than ± 0.0022, or 0.22%. All PS confidence intervals are less than ± 0.012, or 1.2%.

The difference between perfect tagging and imperfect tagging is summarized in Table 4.6. Although each case produces unique results, dependency parsing metrics decrease given imperfect tagging (Table 4.6). An average decrease in tagging accuracy of about 3% results in a statistically significant decrease of about 1% in UAS and LAS scores. This example demonstrates significant value for tagging accuracy improvements similar to the TcT cross-domain tagging accuracy improvements reported in this chapter.

Table 4.5: Perfect-tag and TNT-tag parsing results (% accuracy)

Training	Testing	UAS	LAS	PS
NLTK-WSJ	BioIE	63.5	59.4	8.3
NLTK-WSJ	GENIA	81.4	71.4	0
GENIA	BioIE	69.3	57.3	14.1
BioIE	GENIA	84.0	72.3	17.1
NLTK-WSJ	BioIE	56.4	48.1	4.3
NLTK-WSJ	GENIA	70.3	55.3	0
GENIA	BioIE	66.6	54	12.7
BioIE	GENIA	80	65.7	13.7

4.3.4 Guidelines

My results and analysis imply the following for building effective cross-domain taggers:

- An effective method for handling unknown words is required.

Table 4.6: Difference in tagging and parsing results (% accuracy)

Training	Testing	Tag Acc.	UAS	LAS	PS
NLTK-WSJ	BioIE	-22.8	-7.1	-11.3	-4.0
NLTK-WSJ	GENIA	-16.8	-11.1	-16.1	0
GENIA	BioIE	-15.2	-2.7	-3.3	-1.4
BioIE	GENIA	-7.6	-4.0	-6.6	-3.4
	Avg.	-15.5	-6.2	-9.3	-2.2

- The current token should filter the POS tags and any remaining ambiguity should be resolved through the token's context (e.g., surrounding tokens or tags). For example, allow only the top five POS tags to be disambiguated through context.

- Greater weight should be placed on deriving POS tags from tokens and sequences of tokens relative to POS tag sequence information (e.g., backoff to POS tags when required).

4.3.5 Context of results

This section presents some non-algorithmic aspects, such as corpora, that affect tagging in order to relate my results to previous research and, to some degree, future research. I focus on three aspects: training data, tagger implementation and experimental method.

Experimental method is known to affect results. Methodological differences such as 5-fold and 10-fold cross validation or use of a reserved test set may create discord between research results. These methodological choices and their effect appears to be undocumented for biomedical POS tagging.

Training corpora size [23, 67] and content [23, 67, 88, 48] also effect tagging results. Attention has generally focused on the lexical and syntactic similarity and differences between training and evaluation data, and how these similarities and differences affect evaluation metrics. The observed rule being the greater the similarity between training and evaluation data the better the tagging method performs. Similarity is frequently maximized when training and evaluation data come from one corpus.

In cross-domain tagging, large training corpora are more likely to include lexical and syntactic content similar to the selected evaluation data. In other words, tagging methods trained on large corpora are expected to perform better than those trained

on a small corpora. Assuming a robust tagger is one that requires little training and performs well when applied to many different corpora then large training corpora may hide a tagging method's weaknesses.

Language underpins all corpora and their domains. Tagging methods well adapted to a language (e.g., English or German) are likely to maintain performance across corpora. Hahn and Wermter's [48] tagging experiments are the only tagging experiments reported in this thesis to maintain performance across domains (and corpora). This may suggest that the German language by nature reduces inter-domain differences to the benefit of tagging performance.

Subtle implementation differences between equivalently branded taggers (e.g., English and German Brill [48]) may adversely affect results. Implementation differences may only manifest under specific conditions to the benefit or detriment of tagger performance.

Consider Hahn and Wermter's research [48] as a concrete example of the discussion above. They apply German implementations of TnT and Brill to German news and medical corpora. Their research differs from my research in language, corpora content, corpora size, tagger implementation and experimental methodology. All of these differences and possibly others factor into the differences seen between my results and theirs. More broadly, the non-algorithmic aspects discussed above should be considered when interpreting my results in relation to previous and future research.

4.4 Summary

I built and evaluated a token-centric POS tagger called TcT. TcT differs from the work to date that has focused on retraining existing POS taggers or training taggers on large data sets such as web-based data sets. Instead, TcT investigates the feasibility of developing a biomedical NLP system using a small quantity of free public corpora. Since the free public corpora differ from the target palliative care consult letters, it is desirable for my POS tagger to perform well in cross domain tagging. TcT performed better than three frequently used biomedical text taggers (mxpost, TnT and Brill) in cross-domain POS tagging.

TcT ignores previously assigned POS tags to avoid feedback errors, supporting syntactic CTE (Section 2.9) as a major source of cross-domain POS tagging error. Since some biomedical text such as clinical documents may be ungrammatical (e.g., run-on-sentences, missing punctuation), the token-centric focus of my POS tagger

will likely make it less susceptible to syntactic sequences that differ between training and application data. TcT also provides insight on how to adapt existing algorithms to restricted training conditions (e.g., avoid feedback errors and focus on tokens).

TcT may provide improved performance for situations in which only non-biomedical training data is available and this performance improvement may positively affect NLP components such as parsers (Chapter 5) that rely on part-of-speech tags. The next chapter extends the investigation of Section 4.3.3 into how POS tagging performance affects parsers.

Chapter 5

Effect of POS Tagging Errors on Dependency Parsing

Processing speed and performance has been critical to the usability and acceptance of software by health professionals (Section 2.3.9). Dependency parsers have the potential to quickly and correctly map text input to linguistic structures for semantic understanding purposes. MST [81] and Malt [98] were the two best dependency parsers in the "*CoNLL-X shared task on Multilingual Dependency Parsing*" [14] which measured dependency parsing performance on 13 languages. This challenge demonstrates the generalizability of MST and Malt's parsing techniques.

McDonald et al. and Nivre et al. (Section 2.10) analyzed and compared MST and Malt's parsing errors. However, the effect of POS tagging errors on MST and Malt's dependency parsing remains unknown. It is thought that POS tagging errors cause a large portion of dependency parsing errors. If MST and Malt parsers are applied to biomedical text-processing tasks, then these parsers will likely be subject to imperfectly POS-tagged input.

In this chapter, I analyze how MST and Malt respond to imperfectly POS tagged input in the biomedical domain. I evaluate MST and Malt on perfectly tagged corpora. Since MST performs better than Malt, I modified Malt such that the improved Malt parser is equivalent to MST. I then subject MST, Malt and my improved Malt parser to imperfectly tagged input. Malt parsers demonstrated more resilience to imperfect POS-tag input over MST. For both parsers, most errors were due to miss-tagging of verbs, nouns and adjectives. Since Malt-based parsers performed best, a Malt-based parser was selected for my biomedical NLP system .

5.1 Parser comparisons

I trained and evaluated two state-of-the-art dependency parsers, MST and Malt, that placed first and second in a CoNLL-X challenge. I employed two biomedical corpora and one non-biomedical corpus to evaluate the dependency parsers. Having evaluated the parsers and my parser improvements using ideal input, I employed a frequently used POS tagger to examine the effect of tagging errors on the dependency parsers.

5.1.1 Corpora

Parser comparisons occurred using the NLTK-WSJ, BioIE and GENIA corpus. NLTK-WSJ contains newspaper text. BioIE and GENIA contain biomedical abstracts. Section 2.6.1 contains details such as corpus preparation. Several notable characteristics include

- GENIA contains large deep dependency trees whereas BioIE contains broad shallow trees. PTB is in between these two.
- Few words are shared between corpora. The greatest percentage of shared words occurs between BioIE and GENIA and is largely due to nouns and adjectives.

5.1.2 Training

I trained MST using the same setup reported to have successfully parsed Section 23 of the PTB. In particular, I trained MST with the arguments "training-k:5 loss-type:nopunc decode-type:proj". I trained Malt using the setup extracted from Malt's pre-trained English dependency parser.

5.1.3 Evaluation metrics

The CoNLL-X challenge evaluated parsers using labeled attachment score (LAS). LAS is the percentage of tokens that have a correctly predicted head, dependent and arc label. The CoNLL-X challenge excluded punctuation from scoring. McDonald, Crammer and Pereira [77] report unlabeled attachment scores (UAS). UAS is similar to LAS but excludes the dependency label. Their UAS scoring includes punctuation. Clegg and Shepherd use several evaluation metrics to evaluate dependency graphs. They do not restrict parsing to dependency trees. Their F_{dep} metric is the harmonic

mean of LAS precision and recall. For dependency trees (not graphs), LAS precision is equivalent to recall and thus equivalent to F_{dep}.

I adopted UAS and LAS to evaluate parsing accuracy. I also reported the percentage of perfect sentences (PS). A parser outputs a perfect sentence if the sentence has a perfect LAS. Punctuation is scored. Developing a mechanism to exclude punctuation adds a layer of complexity to an evaluation and may reduce research reproducibility unless the mechanism is well documented. For example, if heuristics are used to remove punctuation, and these are not documented, then reproducing such results could be difficult. Furthermore, some heuristics may not be applicable to specialized domains such as the biomedical domain.

5.1.4 Confidence intervals

Confidence intervals for 95% confidence were calculated using the normal approximation method of the binomial confidence interval [83]. All UAS and LAS confidence intervals are less than ± 0.0022, or 0.22%. All PS confidence intervals are less than ± 0.012, or 1.2%. Non-intersecting confidence intervals indicate statistical significance. Consequently, if the difference between UAS and LAS scores is greater than 0.44% then there is statistical significance at $p \leq 0.05$. Similarly, a PS difference of 2.4% is statistically significant at $p \leq 0.05$.

5.1.5 Plain comparison

Table 5.1 presents the results comparing MST and Malt. MST tends to perform about 2% better in each category. This performance difference is statistically significant.

5.1.6 Malt-all

Malt divides training data into chunks to decrease training time. Yamada et al. [134] report that such divisions decrease parsing performance. I trained Malt without this dividing behavior to understand the dividing behaviour's negative effect on performance. Malt trained without division is called Malt-all.

Table 5.1 presents the results comparing Malt and Malt-all. Malt-all generally performs better than Malt, often a statistically significant difference. Training without the dividing beviour does not completely eliminate the gap between Malt style parsing and MST. MST generally performs better than Malt-all.

Table 5.1: Parsing results (⋆best)

Parser	Training	Testing	UAS	LAS	PS
MST	NLTK-WSJ	BioIE	⋆63.5	⋆59.4	8.3
MST	NLTK-WSJ	GENIA	⋆81.4	71.4	0
MST	GENIA	BioIE	⋆69.3	⋆57.3	14.1
MST	BioIE	GENIA	84.0	72.3	17.1
Malt	NLTK-WSJ	BioIE	60.7	56.6	10.8
Malt	NLTK-WSJ	GENIA	78.1	70.0	15.5
Malt	GENIA	BioIE	66.4	55.3	14.4
Malt	BioIE	GENIA	83.4	71.7	18.1
Malt-all	NLTK-WSJ	BioIE	62.0	58.5	11.1
Malt-all	NLTK-WSJ	GENIA	80.1	71.2	17.2
Malt-all	GENIA	BioIE	67.4	56.3	⋆15.1
Malt-all	BioIE	GENIA	83.9	72.0	18.5
M-Baseline	NLTK-WSJ	BioIE	61.9	58.4	⋆11.3
M-Baseline	NLTK-WSJ	GENIA	80.1	71.3	17.4
M-Baseline	GENIA	BioIE	67.9	56.5	14.6
M-Baseline	BioIE	GENIA	83.9	72.1	18.6
Mediware	NLTK-WSJ	BioIE	62.3	58.9	11.2
Mediware	NLTK-WSJ	GENIA	80.4	⋆71.6	⋆17.5
Mediware	GENIA	BioIE	69.0	57.2	14.2
Mediware	BioIE	GENIA	⋆84.3	⋆72.6	⋆19.6

5.1.7 Malt improvements

Malt was targeted for performance improvements to eliminate the differences in parsing scores between MST and Malt. Furthermore, Malt is less resource intensive, can be faster when training and parsing, and scored lower.[1] Malt parsing was re-implemented within my current software framework. M-Baseline refers to the re-implementation of Malt and Mediware refers to my parsing improvements. I trained M-Baseline and Mediware without dividing behaviour.

M-Baseline differs from Malt by a static feature. This feature is the dependency label attached to the first input word. It was removed because no dependency label is ever attached to the first input word. In a Malt configuration, the feature is written as *OutputColumn(DEPREL, Input[0])*.

M-Baseline and Mediware's features (Table 5.2) come from three data structures.

[1] I also discovered that MST modified its input converting numeric values to a placeholder. This is undesirable in the biomedical domain given that numeric values are often important (e.g., oxygen levels in the blood, pulse, blood pressure).

The data structures are

- the sequence of past operations (transitions). The last operation is labelled -1 and the prior one -2.
- the processing stack. The top of the stack is labelled 0, with increasing integer labels from 0.
- the input words. The first input is labelled 0, with increasing integer labels from 0.

Table 5.2: Parse features (a=Malt/Mediware baseline, e=Mediware)

Data structure	Word	Lowercase stem	POS tag	Edge label	Operation label
operation[-1]					e
operation[-2]					e
stack[0].node	a	e	a,e		
stack[0].left	a	e	a,e	a,e	
stack[0].right			a,e		
stack[1].node	a	e	a,e		
stack[1].left			a,e		
stack[1].right	a	e	a,e	a,e	
stack[2].node	a	e	a,e		
stack[3].node			a,e		
input[0].node	a	e	a,e		
input[1].node	a	e	a,e		
input[2].node			a,e		

The input and processing stack hold dependency tree nodes. These nodes or their relations provide features to the classifier and are referenced through labels. The label *node* specifies the current node. The label *left* and *right* respectively specify the left most dependent prior to the current word and the right most dependent after the current word (prior and after relative to the input sequence). The features are a word, a word's stem, a word's POS tag, a dependency edge label for the edge linking a word to its head word and an operation label returned by the oracle.

Mediware differs from M-Baseline in its use of lowercase stems instead of words and by the addition of previous operation labels. Information is lost (e.g., suffix) in stemming; however, stemming transforms a word into a form that is more likely to be found outside of training. Lost information is retained by other features (e.g., retained by the word's POS tag).

Adding previous operation labels provides Mediware with its computational state along with its linguistic state. The computational state should help Mediware in cases where it encounters a unique or infrequently occurring linguistic state. Mediware's SVM parameters were degree 2, gamma 0.18, coefficient 0.18 and epsilon 1.

Table 5.1 presents the results comparing Malt-all, M-Baseline and Mediware. M-Baseline is roughly equivalent to Malt-all. Mediware performs better than M-Baseline and Malt-all in several categories. Mediware and MST generally perform best of all parsers and parsing configurations.

5.1.8 Imperfect POS tagging

I evaluated the effect of POS tagging on parsing. TnT POS tagged each corpus prior to parsing. I selected TnT because it is well known [59], was frequently employed to tag biomedical text (Chapter 4) and is available in NLTK (NLTK tagging demos). I mirrored the training, application and evaluation of the parsers for the tagger. For example, if the parser was trained on BioIE and applied to GENIA then the tagger was trained on BioIE and applied to GENIA. This is likely representative of real world biomedical software system training, that all software system components are trained using the same data sets.

The lower the tagging accuracy, the more a dependency parser must contend with imperfect input. The lowest tagging accuracy occurs when training on NLTK-WSJ and testing on BioIE. Table 5.3 reports all tagging accuracies. All tagging accuracy confidence intervals are less than ± 0.0018, or 0.18%.

Table 5.3: Tagging accuracy

Training	Testing	Accuracy
NLTK-WSJ	BioIE	77.2
NLTK-WSJ	GENIA	83.2
GENIA	BioIE	84.8
BioIE	GENIA	92.4

Mediware outperforms MST on imperfectly POS tagged corpora (Table 5.4). Mediware's performance improvement is often statistically significant.

Table 5.4: Imperfect POS tag parsing (⋆best)

Parser	Training	Testing	UAS	LAS	PS
MST	NLTK-WSJ	BioIE	56.4	48.1	4.3
MST	NLTK-WSJ	GENIA	70.3	55.3	0
MST	GENIA	BioIE	66.6	54	12.7
MST	BioIE	GENIA	80	65.7	13.7
Malt-all	NLTK-WSJ	BioIE	56.7	48.3	9
Malt-all	NLTK-WSJ	GENIA	71.1	56.1	⋆10.2
Malt-all	GENIA	BioIE	66.3	53.9	⋆13.7
Malt-all	BioIE	GENIA	79.8	65.4	14.9
Mediware	NLTK-WSJ	BioIE	⋆57.2	⋆48.7	⋆9.1
Mediware	NLTK-WSJ	GENIA	⋆71.3	⋆56.3	⋆10.2
Mediware	GENIA	BioIE	⋆66.9	⋆54.4	13.1
Mediware	BioIE	GENIA	⋆80.3	⋆65.9	⋆15.6

5.2 Discussion

POS tag errors negatively affect, without exception, all parsers across all metrics (Table 5.5). LAS and PS metrics generally follow UAS trends. For example, training with NLTK-WSJ and testing on GENIA produces the worst scores (the one exception being Malt-all PS). The remaining analysis mainly focuses on UAS scores.

Table 5.5: Parsing differences due to imperfect POS tagging

Parser	Training	Testing	UAS	LAS	PS
MST	NLTK-WSJ	BioIE	-7.1	-11.3	-4.0
MST	NLTK-WSJ	GENIA	-11.1	-16.1	0
MST	GENIA	BioIE	-2.7	-3.3	-1.4
MST	BioIE	GENIA	-4.0	-6.6	-3.4
Malt-all	NLTK-WSJ	BioIE	-5.3	-10.2	-2.1
Malt-all	NLTK-WSJ	GENIA	-9.0	-15.1	-3.5
Malt-all	GENIA	BioIE	-1.1	-2.4	-1.4
Malt-all	BioIE	GENIA	-4.1	-6.6	-3.6
Mediware	NLTK-WSJ	BioIE	-5.1	-10.2	-2.1
Mediware	NLTK-WSJ	GENIA	-9.1	-15.3	-7.3
Mediware	GENIA	BioIE	-2.1	-2.8	-1.1
Mediware	BioIE	GENIA	-4.0	-6.7	-4

Mediware's UAS decreases are similar to Malt-all's. This suggests that Mediware's improvements do not increase Mediware's resilience to POS tagging errors. Malt's transition-based parsing appears more resilient to POS tagging errors than MST's

graph-based parsing. MST's UAS generally decreases more than Mediware's.

I derived coarse POS categories in accordance with McDonald et al. [78]. The categorization was verb, adverb, pronoun, noun, adjective, determiner, symbols (including punctuation), prepositions and several other small categories. Table 5.6 reports tagging accuracy for these coarse categories. UAS decreases are best explained by coarse POS categories compared to detailed POS categories, in Table 5.3. Table 5.7 presents results from performing a linear regression (model $UAS = C_1 tag + C_0$) on UAS scores explained by coarse tagging accuracies. Table 5.7 suggests that POS tagging errors that change a token's POS category are more harmful than intra-category changes.

Table 5.6: Coarse tagging accuracy

Training	Testing	Coarse accuracy	3-Coarse accuracy
NLTK-WSJ	BioIE	90.1	87.8
NLTK-WSJ	GENIA	88.2	82.7
GENIA	BioIE	95.7	95.8
BioIE	GENIA	94.2	91.6

Table 5.7: UAS and LAS scores explained by tagging accuracy via linear regression

Parser	Output	Tag	Coarse tag	3-Coarse tag
MST	UAS	$p \leq 0.55$	$p \leq 0.026$	$p \leq 0.016$
Malt-all	UAS	$p \leq 0.74$	$p \leq 0.061$	$p \leq 0.009$
Mediware	UAS	$p \leq 0.73$	$p \leq 0.072$	$p \leq 0.021$
MST	LAS	$p \leq 0.55$	$p \leq 0.026$	$p \leq 0.016$
Malt-all	LAS	$p \leq 0.62$	$p \leq 0.025$	$p \leq 6.0^{-5}$
Mediware	LAS	$p \leq 0.63$	$p \leq 0.026$	$p \leq 0.00022$

On further inspection, UAS decreases related to three coarse POS categories. These are verbs, nouns and adjectives. Table 5.6 includes POS tagging accuracies restricted to these latter three tags (3-coarse accuracy). Table 5.7 also presents results from a linear regression using tagging accuracies restricted to verbs, nouns and adjectives. Tagging accuracy on verbs, nouns and adjectives better explains UAS scores compared to coarse tagging accuracies. Results (Table 5.7) are similar for LAS scores.

Table 5.8 contains the average number of dependents per coarse tag for the top five tags. Relative to other tags, verbs, nouns and adjectives have numerous dependents. Verbs and nouns have the most. Errors in tagging verbs, nouns and adjectives not only affect the current dependency but also affect dependencies between the current

token and its dependents. This error cascade is likely responsible for the correlation between the accuracy on these 3 coarse POS tags and the UAS and LAS.

Table 5.8: Average number of dependents per coarse tag (top 5 tags)

Tag	NLTK-WSJ	GENIA	BioIE
Verb	2.5	2.3	2.2
Noun	1.2	1.4	1.4
IN (e.g., preposition)	0.91	0.96	0.91
TO (the word to)	0.41	0.62	0.62
Adjective	0.50	0.49	0.60

The effect of tagging accuracy on verbs, nouns and adjectives is illustrated by Figure 5.1, Figure 5.2 and Figure 5.3. These figures relate UAS changes relative to F-measure changes (for F-measure see Section 2.6.2). Table 5.9 presents the F-measure and F-measure change for verbs, nouns and adjectives.

Table 5.9: F-measure and its decrease for verbs, nouns and adjectives

Training	Testing	Verb	ΔVerb	Noun	ΔNoun	Adj	ΔAdj
NLTK-WSJ	BioIE	0.83	-0.17	0.90	-0.10	0.59	-0.41
NLTK-WSJ	GENIA	0.81	-0.19	0.87	-0.13	0.54	-0.46
GENIA	BioIE	0.95	-0.05	0.97	-0.03	0.86	-0.14
BioIE	GENIA	0.94	-0.06	0.94	-0.06	0.79	-0.21

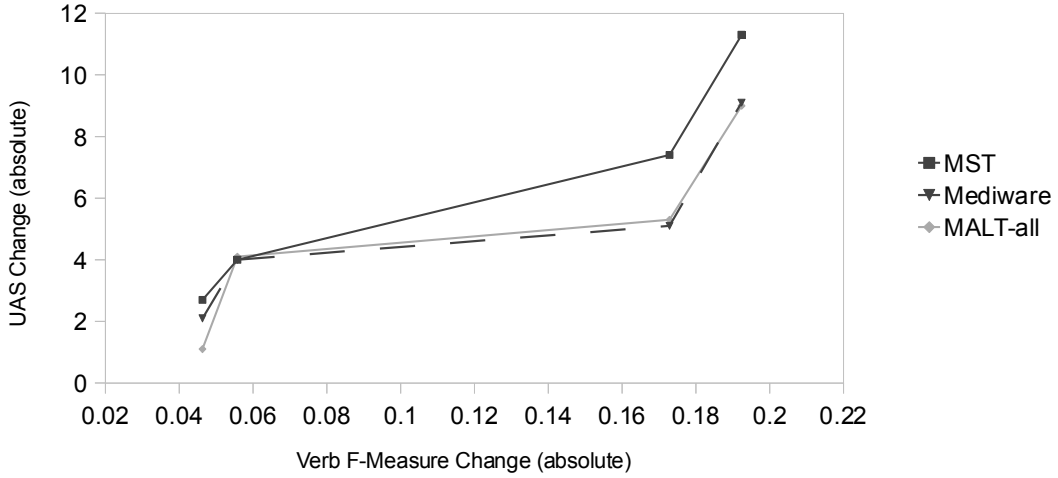

Figure 5.1: UAS change vs verb F-measure change

MST performs better than Malt on verbs and adjectives. Since a large accuracy decrease occurs in these two categories MST is likely most affected. Malt uses a rich

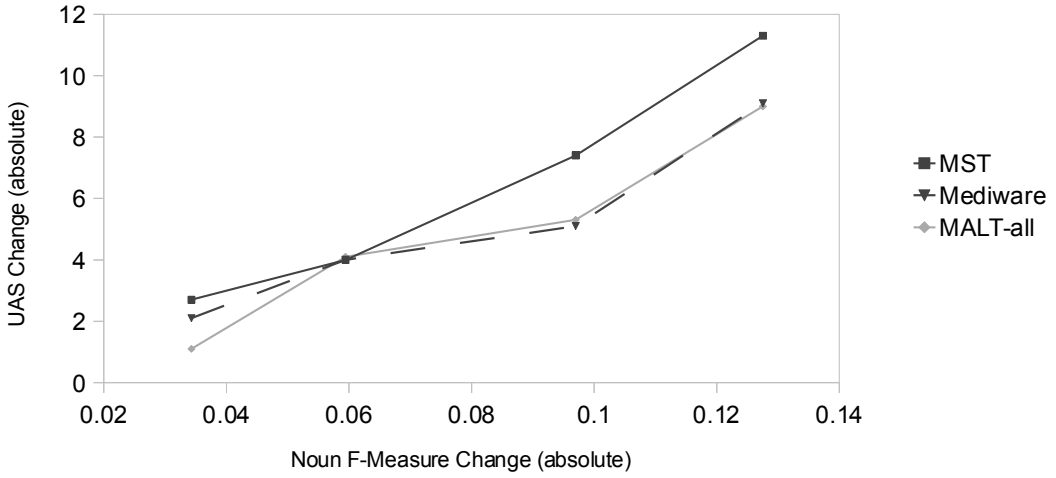

Figure 5.2: UAS change vs noun F-measure change

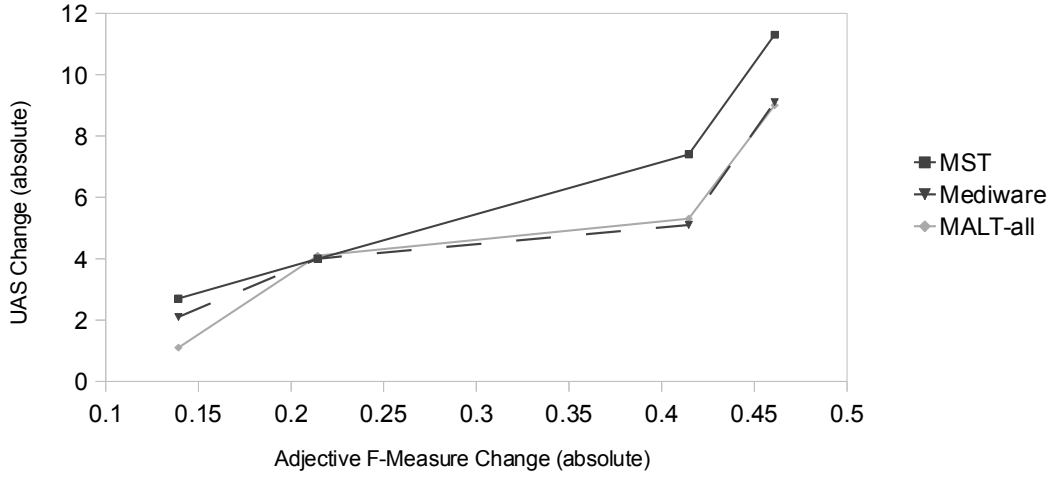

Figure 5.3: UAS change vs adjective F-measure change

feature set for parsing decisions and may be able to offset tagging errors with other features. In other words, MST depends more heavily on POS tags for decisions and may consequently be most affected by POS tagging errors. Malt's local focus may also be advantageous. POS tagging errors outside of Malt's local scope do not affect any local decisions, whereas errors have global consequences for MST.

Malt divides learning and training data into groups to speed processing. Malt trains a classifier for each group. Although data may be divided in numerous ways, the Malt English setup and configuration (December 2010) divides data on POS tags. This choice may have significant consequences for poorly tagged data since the repercussion of an erroneous tag is to apply the wrong classifier. It may be better to split data on word endings (e.g., suffix or a word's last two letters).

If a POS tagger is paired with either MST or Malt then a good POS tagger is likely one that reduces inter-category POS tagging errors and performs well on verbs, nouns and adjectives. In other words, my results suggest POS tagger evaluation metrics. My results also suggest that POS taggers should focus on features that differentiate verbs, nouns and adjectives (e.g., capital letters, suffixes).

5.3 Summary

MST and Malt are state-of-the-art dependency parsers. This chapter presented a comparison between MST and Malt on imperfectly POS tagged biomedical text. This chapter informs my parser selection (Malt) for automated clinical coding (Chapter 6) and my biomedical NLP system as a whole.

Since Malt initially performed worse than MST, Malt was adapted to the biomedical domain. I called this adaption Mediware, and its performance was roughly equivalent to MST's prior to comparing parsers on imperfectly POS tagged biomedical text. Since Malt's adaptions do not rely on text characteristics unique to biomedical text, the adaptions may improve dependency parsing on other domains.

MST was most affected by imperfectly POS tagged biomedical text. I attributed MST's drop in performance to two POS categories (verbs and adjectives) where MST had more potential for performance loss than Malt. I also attributed Malt's resilience to POS tagging errors to its use of a rich feature set and a local scope in decision making. These latter two properties either compensate for or ignore POS tagging errors. Given the above, Malt parsers are more appropriate for my biomedical NLP system since these parsers perform best.

POS taggers developed for use with MST and Malt should give special consideration to correctly tagging verbs, nouns and adjectives. Errors in these latter three categories significantly effect parsing accuracy.

Chapter 6

Automated Clinical Coding with Semantic Atoms and Topology

ACC is expected to supplement, enhance or replace manual coding practices. Beyond its effect on manual coding practices, ACC may be incorporated into biomedical NLP. For example, ACC could be used to normalize diverse natural language phrases (Section 2.3, diversity of language). Furthermore, ACC may inform biomedical NLP by providing machine processable health context to biomedical NLP algorithms (Section 2.3, domain knowledge).

Previous ACC research focuses on mapping narrative phrases to terminological descriptions (e.g., concept descriptions) (Section 2.11). These methods make little or no use of the additional semantic information available through topology. I investigated a token based ACC approach that begins to exploit additional semantic information available in SNOMED CT. Exploiting additional semantic information may increase ACC effectiveness through better

- mapping of linguistic structures into SNOMED CT
- result filtering that identifies a few highly probable concepts rather than many possible concepts
- use of SNOMED CT's post-coordinated expressions allowing for greater and more accurate coding.

This chapter demonstrates the viability of non-phrase based ACC methods and suggests that further study of such methods is warranted.

6.1 Coding considerations

Prior to developing an ACC method, sentinel events were examined in context of knowledge representation (Section 2.5). The lessons learned from this process and the additional consideration of missing terminology is discussed with respect to ACC.

Despite the large number of SNOMED CT concepts, many concepts found in clinical narrative (narrative concepts) must be post-coordinated. My ACC method does not output post-coordinated expressions due to their additional implementation complexity. In particular, a system that performs both pre and post coordinated encodings must recognize when to apply one encoding over the other. With respect to my ACC method, the goal is to develop an understanding of pre-coordinated encoding before attempting to recognize when post-coordination should be applied.

Certainty (e.g., may represent cancer), quantitative (e.g., 2 pills) and temporal (e.g., past fracture) information are poorly represented by SNOMED CT. Consequently, these types of information should be encoded external to SNOMED CT.

A consideration not discussed in Section 2.5 is missing terminology. For example, SNOMED CT is missing terminology such as the abbreviation *mets* (metastasis). Without the binding between concepts and narrative (e.g. through concept descriptions), there is no direct mechanism for coding missing terminology. Such codings may be feasible through other means and may require substantial additional effort.

Attempting to represent sentinel events in SNOMED CT uncovered three design considerations. The first is to account for post-coordinated encoding in method design. The second is to represent certainty, quantitative and temporal information outside of SNOMED CT. The third is to allow for additional terminology to be bound to concepts. The latter accounts for any terminology absent from SNOMED CT such as physician specific (local) terminology that is unlikely to be included in a SNOMED CT release.

6.2 ACC method

My ACC method encodes tokens as SNOMED CT concepts and manipulates (e.g. combines) concepts according to linguistic structures present in narrative text. The fundamental principle underlying my ACC method is to reduce semantic gain and loss. Semantic gain occurs when additional semantics are captured by the coding that do not exist in narrative. Semantic loss occurs when semantics are omitted in

the coding that occur in narrative. Although this is an implicit objective of many coding methods, I present the objective explicitly because this principal dominated my conceptualization and development.

6.2.1 High-level description

Step 1

- Use language processing techniques to identify each description's tokens and normalize these tokens to semantic atoms (Figure 6.1 drawing B). The final result is that each concept is semantically defined by a set, or multiple sets, of semantic atoms.

The first step reduces concept descriptions to normalized token sets. Example tokens are words such as *the* and *infection*, acronyms or abbreviations such as *HIV*, numeric values, dates such as *10/2011*, etc. Normalization addresses variation in token form. For example, the forms *fracture* and *fractures* are nearly semantically equivalent. It may be desirable to represent both forms using *fracture*. As another example consider the word *two*, the number *2* and a personal notation such as ‖ meaning *2*. These could be represented by the number *2*. These normalized tokens are called semantic atoms. Semantic atoms are an internal (computational) semantic representation.

Semantic atoms are collected in sets. Each concept is described by a set of semantic atoms or by multiple semantic atom sets. Multiple semantic atom sets may be used to distinguish between preferred semantics (e.g., preferred descriptions) and synonymous semantics (e.g., non-preferred descriptions).

Additional descriptions may be associated to concepts prior to applying Step 1. This accounts for local medical terminology absent in SNOMED CT. The exact process that converts concept descriptions to semantic atoms is dependent on the language processing techniques and algorithms selected. Applying Step 1 to the SNOMED CT concept 91302008 with description "Systemic infection (disorder)" could result in the semantic atom set {systemic infection disorder}, where the semantic atoms are separated by space.

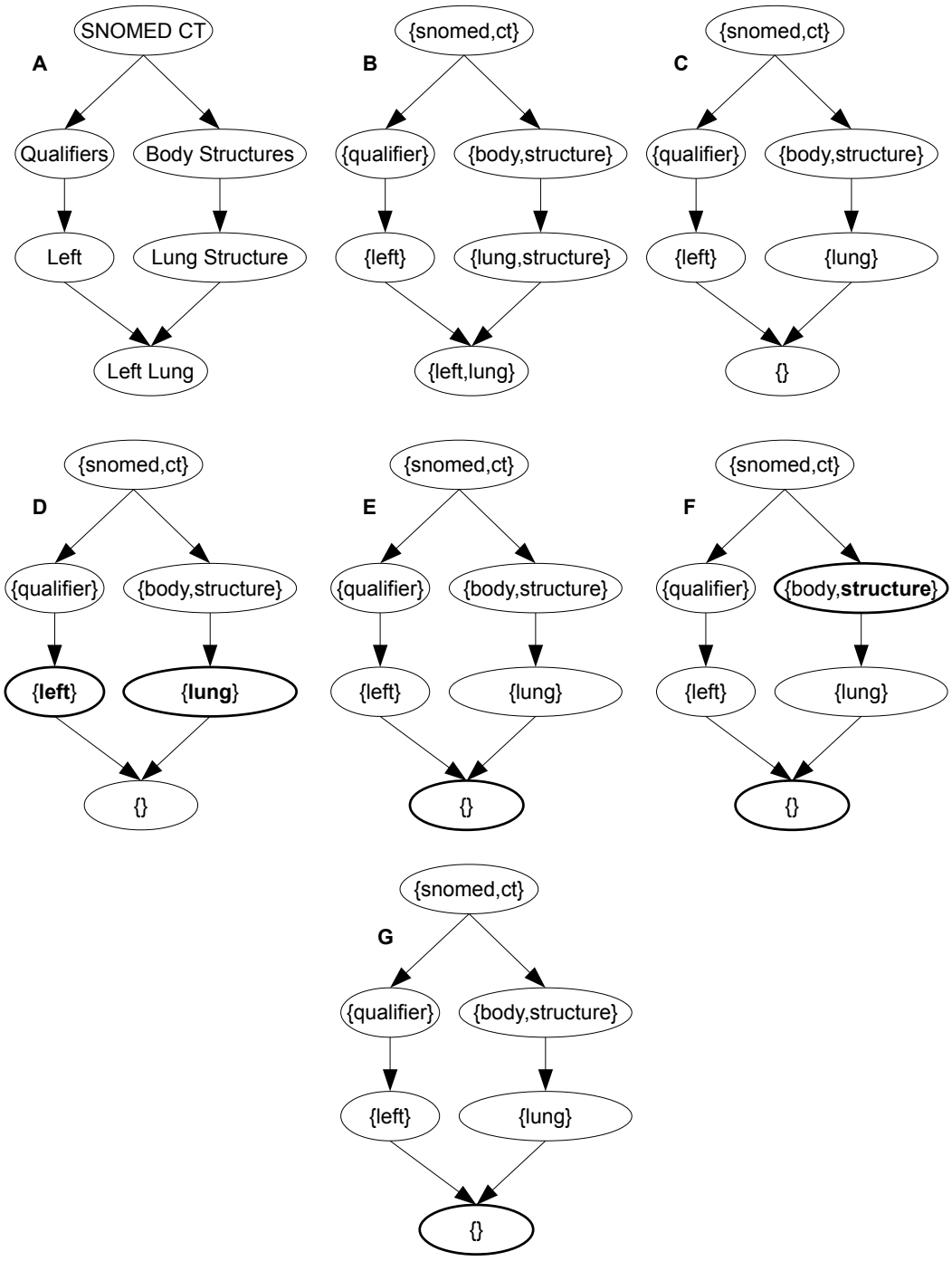

Figure 6.1: A simplified visual representation of ACC steps 1-5 for the phrase *structure of left lung*

Step 2

- Identify where each semantic atom is first introduced in the terminology by traversing the terminology's topology and discarding previously introduced semantic atoms (Figure 6.1 drawing C). This process may be repeated or run simultaneously to handle multiple semantic atom sets associated with each concept.

Step 2 is accomplished by traversing the SNOMED CT topology starting from the root (most fundamental concept) and discarding a concept's semantic atoms that have been previously introduced by its semantic ancestors (Algorithm 4 and Algorithm 5). For example, if *fracture (clinical finding)* is a descendent of *Clinical finding* then *clinical* and *finding* would be discarded from *fracture (clinical finding)* resulting in *fracture*.

Algorithm 4 *createUniqueSemantics* isolates semantics in SNOMED CT

```
Input: root, concepts
  excludedConcepts ← exclude( concepts )
  reduceGraphSemantics( root, ∅, excludedConcepts )
  for c in concepts do {adjust semantic reduction}
    if semantics( c ) = ∅ then
      if hasOneParent( c ) then
        bindUniqueSemantics( semantics( c ), c )
      else
        for p in parents( c ) do
          if semantics( c ) = semantics( c ) ∩ semantics( p ) then
            bindUniqueSemantics( semantics( c ), c )
          end if
        end for
      end if
    end if
    if isSingleWordSemantic( c ) then
      bindUniqueSemantics( semantics( c ), c )
    end if
  end for
```

Semantic ancestors depend on a semantic hierarchy definition. A semantic hierarchy may differ from the conventional SNOMED CT hierarchy. For example, if a concept, C, is related to a concept, R, then semantic ancestors may include R, a subset of R's ancestors or descendants (Figure 6.2). An inclusive case (Figure 6.2, right side) treats R, its ancestors and its descendants as semantic ancestors.

Algorithm 5 *reduceGraphSemantics* removes redundant semantics

Input: concept, parentSemantics, excludedConcepts
  collect(parentSemantics, concept)
  if not processedAllParents(concept) **then**
    return
  end if
  if concept ∈ excludedConcepts **then**
    newParentSemantics ← parentSemantics
  else
    uniqueSemantics ← semantics(concept) \ collectedSemantics(concept)
    bindUniqueSemantics(uniqueSemantics, concept)
    newParentSemantics ← collectedSemantics(concept) ∪ semantics(concept)
  end if
  for child of concept **do**
    reduceGraphSemantics(child, newParentSemantics)
  end for

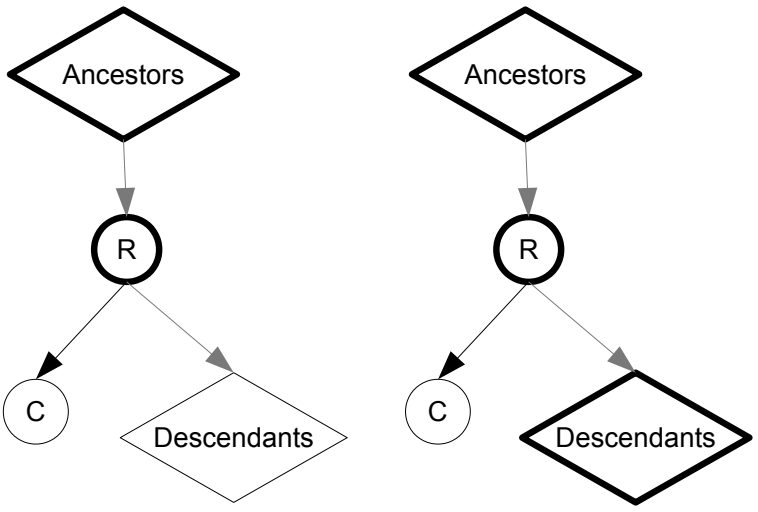

Figure 6.2: Two example semantic hierarchy cases (highlighted) of concept C

Step 3

- Perform token-level coding by retrieving, for each token, all SNOMED CT concepts that introduce this token (via its semantic atoms) to the SNOMED CT topology (Figure 6.1 drawing D and F).

A token-level coding locates all concepts that introduce a token's semantic atom (or atoms) to the SNOMED CT topology. This step is incorporated into Algorithm 6. A token-level coding does not imply clinically coding each token. Particular situations or requirements may see coding restricted to certain word types, etc. For example, coding could be restricted to verbs or noun phrases.

Step 4

- Manipulate (e.g., combine) concepts, possibly lists of concepts, according to the linguistic structures present in the partially coded narrative text and according to SNOMED CT's accepted pre-coordinated and post-coordinated operations (Figure 6.1 drawing E and G). Step 4 and 5 refer to concepts; expressions are implied as well.

Step 4 maps linguistic structures to SNOMED CT pre-coordinated and post-coordinated operations. This step is incorporated into Algorithm 6. Post-coordination involves SNOMED CT expressions. IHTSDO define concepts and expressions, and specify how these may interact during pre and post coordination.

ACC may use token-level and linguistic structure codings to code progressively more complex linguistic structures. Coding linguistic structures does not imply coding every linguistic structure. Particular situations or requirements may see coding restricted to certain structures such as noun phrases.

Step 5

- Simplify results by selecting those concepts that have no semantic ancestors within the result set (Figure 6.1 drawing G). In other words, select the semantically general concepts.

Results are simplified by discarding concepts whose semantic ancestors exist within the results. This step is incorporated into Algorithm 6. For example, if *clinical finding* and *fracture (clinical finding)* exist in the results then *fracture (clinical finding)* would be discarded leaving a more general concept *clinical finding*.

Algorithm 6 *locate* locates and ranks concepts

```
Input: semantics, root
  conceptLists ← newList()
  for s in semantics do {convert semantic into ranked concepts}
    rankedConcepts ← findAndRankConcepts( s )
    append( rankedConcepts, conceptLists )
  end for
  ranking ← newScoredList( root )
  for rankedConcepts in conceptLists do {merge ranked concepts}
    tmp ← newList()
    for b in ranking do
      for c in rankedConcepts do
        rankedChildren ← sharedChildren( b, c )
        append( rankedChildren, tmp )
      end for
    end for
    ranking ← tmp
    simplify( ranking ) {Removes specializations}
  end for
  return ranking
```

Post-coding

Depending on the application, one or more of the coded results may be used. For example, an ACC application would likely use the highest ranked coding whereas semi-automated clinical coding could see all results presented to a human coder.

6.2.2 Implementation specifics

This section refines the high-level algorithm presented above (Section 6.2.1). This section presents implementation specifics required to reproduce the evaluation results in Section 6.3. For example, semantic scoring of concepts is discussed.

Step 1

Each concept's descriptions are reduced to two semantic atom sets. One set includes a concept's preferred descriptions and the other set the concept's non-preferred descriptions. This conceptually separates the semantic atoms into preferred and synonymous semantics. This separation is important when scoring token-level codings.

Tokens, and consequently semantic atoms, are filtered by POS tag. Permissible semantic atoms are derived from adjectives, nouns, verbs, adverbs, foreign words, symbols and numbers. Each permissible token is normalized according to its POS tag and several rules. For example, plural nouns become singular and verb forms are normalized to their base form.

Step 2

Semantic ancestors encompass *defining* and *qualifying* relationships. These relationships include a refinability attribute. Refinability specifies details of a concept's defining or qualifying characteristics. Refinability may be restricted to a related concept only (not refinable, left side of Figure 6.2), a related concept and its descendents (optionally refinable, right side of Figure 6.2) or solely a related concept's descendents (mandatory refinement). Currently, all relationships that are refinable are treated as not refinable. This definition is applied recursively to obtain all ancestor semantics for a particular concept.

Caveat for Step 2

Identifying where semantic atoms are first introduced is complicated by semantic removal. Consequently, semantic removal is identified and semantic atoms surrounding removal are adjusted.

A concept may be distinguished from its semantic ancestors by adding or removing semantic atoms. In the example above, *fracture* was added to *clinical finding* creating a new concept *fracture (clinical finding)*. Semantic addition is handled by collecting a concept's ancestor semantic atoms and removing these from the current concept's semantic atoms. The remaining semantic atoms are first introduced in the semantic hierarchy by the current concept.

Semantic removal is more problematic. *Ankle* may be removed from *ankle and foot* to create a new concept *ankle*. The concept *ankle and foot* does not introduce the concept *ankle* but a broader concept inclusive of *ankle*. Since this broader concept lacks a specialized name, it is named with a conjunction of its children. Semantic removal cases, such as the latter, need to be identified and handled to properly recognize the introduction of semantic atoms. The following schemes were applied to identify and address semantic removal:

- Concepts with a description containing the text *(combined site)* are excluded while handling semantic addition. These concepts are handled individually after handling semantic addition.
- Concepts that share all of their semantic atoms with a single parent have their semantic atoms reinstated after handling semantic addition.
- All concepts described by one token (e.g., brain) have their semantic atoms reinstated after handling semantic addition.

Step 3

A token-level coding locates all concepts that introduce a token's semantic atom (or atoms). The retrieved concepts are scored to identify a token's most representative concept. Different scoring algorithms may be implemented to optimize scoring for particular situations or requirements.

A token-level score is the sum of the token match, semantic difference and popularity sub-scores. The token match score quantifies the similarity between a token and a concept's descriptions. The semantic difference sub-score models how far a

concept is from introducing a single semantic to the semantic hierarchy. The popularity sub-score models a concept's importance via its affect on other concepts. The following definitions are used when describing token-level scoring. To simplify the description below, it is assumed that each token results in a single semantic atom.

- t is a token
- a is t's semantic atom
- c is a concept
- pt is c's preferred term
- syn is a set containing c's descriptions excluding pt
- s_{pt} is a set containing semantic atoms derived from pt
- s_{syn} is a set containing semantic atoms derived from descriptions in syn
- $s = s_{pt} \cup s_{syn}$
- p is a set containing c's parents (semantic hierarchy)
- h is a set containing c's children (semantic hierarchy)

The sub-scores:

- Token match: $m = \begin{cases} 4, & \text{if } pt = t \\ 3, & \text{if } t \in syn \wedge a \in s_{pt} \\ 2, & \text{if } t \in syn \\ 1, & \text{if } a \in s_{pt} \\ 0, & \text{otherwise} \end{cases}$
- Semantic difference: $sd = 2/(|s| + |p|)$
- Popularity: $pop = max((|h| - |s| - |p|)/(|h| + 1), 0)$

Step 4

Token codes are manipulated by mapping linguistic structures present in the partially coded narrative to topological operations. Currently, linguistic structures are mapped to a single *merge* operation. This operation computes the shared semantic descendents between two concepts and selects the most general of these semantic descendents.

A merge operation should obey the established pre and post coordination rules and expression format established by the IHTSDO. To simplify my prototype, I relaxed refinability attributes and focused on pre-coordinated concepts (for details see *Step 2*).

Coding linguistic structures is further explained by example (Figure 6.1 drawings D to G). Consider the phrase *structure of left lung*. In this phrase, *left* modifies *lung* and *left lung* modifies *structure*. My ACC implementation first addresses *left* and *lung* by locating the concepts (if any) that share these semantics. This is accomplished by locating all concepts that introduce the semantic *left* as well as all concepts that introduce the semantic *lung* (as per the token-level coding). The merge operation is performed on all *left-lung* concept pairs. The result is a concept list where all concepts share *left* and *lung* semantics. The results are ranked to locate the best (highest ranked) concept for the current linguistic structure (i.e., left lung) and to adjust concept scores for further merging. The *left-lung* codings are combined with the *structure* codings to produce a final coding for the entire *structure of left lung* linguistic structure.

My ACC method implementation acts on pairs of head and dependent tokens contained in dependency parsing structures. Using token features (e.g., POS tags) and the linguistic relationship between tokens, token-level codings are merged to produce a list of codings that represent both tokens and their linguistic relationship. This process may be applied repeatedly; progressively merging coded tokens and linguistic structures. The latter process starts with the dependency trees' leaves and progresses to the root.

Several linguistic structures are coded. They are adjective phrases, noun phrases, prepositional phrases and verbs. The object of the preposition must be a noun phrase or adjective phrase (accounts for processing errors). Verb structures are restricted to the verb itself, adverb modifiers, simple subjects and objects. Linguistic definitions for the coded structures are relaxed. A linguistically strict approach excludes mal-

formed linguistic structures that may occur due to language processing errors. For example, an adjective incorrectly labelled as a general *dependent* rather than the correct *adjective modifier* is handled as an *adjective modifier*. This pragmatic approach likely results in improper handling of several linguistic structures and blurs the separation between the linguistic structures handled by my ACC method implementation and those that are not. However, this pragmatic approach improved coding results in pre-evaluation testing on sentinel event text.

Merge results are scored for subsequent merging and for determining the best (highest scoring) coding for the current linguistic structure. Different scoring algorithms could be implemented to maximize scoring for particular situations or requirements. Merge scores are calculated as the sum of the two pre-merge concept scores multiplied (adjusted) by the closeness of the merged concept semantics to narrative concept semantics. Semantic closeness is calculated as the average of four sub-scores. The sub-scores are the closeness of preferred term semantics, closeness of synonym semantics, semantic sufficiency and concept popularity sub-scores. The following definitions are used when describing the merge adjustment sub-scores.

- α is a set of semantic atoms representing the current linguistic structure
- c is a concept
- s_{pt} is a set containing semantic atoms derived from c's preferred term
- s_{syn} is a set containing semantic atoms derived from c's synonym descriptions
- h is a set containing c's children (semantic hierarchy)

The sub-scores:

- Closeness of preferred term semantics: $(2|s_{pt} \cap \alpha|)\ /\ (|s_{pt}| + |\alpha|)$
- Closeness of synonym semantics: $(2|s_{syn} \cap \alpha|)\ /\ (|s_{syn}| + |\alpha|)$
- Semantic sufficiency: $|(s_{pt} \cup s_{syn}) \cap \alpha|/|\alpha|$
- Concept popularity: $1 - 1/(1 + h)$

Step 5

The last step simplifies the results by discarding specialization. Simplification is based on generalization and specialization defined in SNOMED CT by the IHTSDO.

6.3 Evaluation

I performed two evaluations. In the first evaluation (named *recall*), my ACC method processed current SNOMED CT concept descriptions (version July2011). The objective was to recall the correct SNOMED CT concept associated with each concept description. For example, my ACC method was expected to output concept code 187622006 given the input description "Malignant tumor of labial mucosa". A total of 769,427 descriptions were automatically coded (each concept has multiple descriptions).

My ACC method correctly coded 85.6% of the concept descriptions. The highest possible score is 98.2% because identical descriptions associate with multiple concepts. A score of 98.2% may be obtained by ACC systems performing an exact string match between the input string and the concept descriptions. My ACC method could be modified to score 98.2% by doing an exact match prior to invoking my ACC method as described.

The second evaluation (named *precision*) compared my ACC method to MetaMap on encoding palliative care diagnosis and referral problems in SNOMED CT. MetaMap was selected because it is well known, is publicly available and has performed well on similar tasks (e.g., extracting medical problems from narrative [86]) and is thus a good benchmark. Small texts were encoded using my ACC method and MetaMap. To asses ACC correctness, I compared the automated codings to a human verified gold standard encoding. I then compared my ACC method's correctness to MetaMap's.

Lee et al. [66] created an anonymized data set of palliative care diagnosis and referral problems. The data set contains small texts such as "breast or cervical origin metastatic ca wi lung mets". Lee et al. encoded the texts as pre-coordinated SNOMED CT concepts, post-coordinated expressions and non-coded texts. In other words, the data set contains texts of varying ACC difficulty. It is important to note that Lee et al. manually encoded texts and verified automated encodings. Thus, all encodings included expert oversight.

The data were filtered by removing duplicate texts, texts that contained punctuation and texts that matched existing SNOMED CT descriptions. Texts contained punctuation if they contained any of the following symbols *[—!:;,.?]*. Texts matched existing SNOMED CT descriptions if they were an exact lower case string match to an existing description. Filtering guaranteed that the second evaluation did not duplicate the first (recall) evaluation. It also guaranteed that ACC would not need

to cross punctuation boundaries - a small simplification.

The data set originally contained 51656 texts. There was 38046 duplicated texts, 578 texts matching existing SNOMED CT descriptions and 8,437 texts containing punctuation. This left 4595 for use. A computer script randomly selected a sample of 300 texts from the 4595. Table 6.1 and 6.2 characterize the population (4595) and random sample (300). These tables suggest that the sample is fairly representative of the population.

I manually verified the previously assigned codes, simplified post-coordinated expressions to single concepts and encoded non-coded texts. Of the 300 texts, 82 were non-coded and 7 were post-coordinated. To accomplish verification, simplification and encoding, I manually encoded all texts. I verified previously assigned codes by comparing them to my coding. In 5 cases, I simplified the coding to account for missing context (context present during the original research). For example, *on neck* encoded as "malignant melanoma of neck" was simplified to "entire neck". When simplifying post-coordinated expressions, I relied on my manual coding and the codes within post-coordinated expressions. In most cases, the post-coordinated expressions contained the simplifying concept. For example, the post-coordinated expression "64572001: 116676008=35917007" was simplified to "35917007". Appendix C contains the codes for the 5 pre-coordinated cases, the post-coordinated expressions and non-coded texts.

Since MetaMap does not output SNOMED CT codes, I converted SNOMED CT concepts to MetaMap identifiers to assess MetaMap's correctness. I used the Metathesaurus web application[1] to convert SNOMED CT concepts to MetaMap identifiers. When a SNOMED CT concept mapped to several MetaMap identifiers, a MetaMap identifier was selected based on provided descriptions. For example, the text "osteoarthritis" is coded in SNOMED CT as 396275006. The Metathesaurus provides two possible MetaMap identifiers C0022408 and C0029408 for the SNOMED CT code 396275006. The MetaMap identifier descriptions (below) guided my MetaMap identifier selection.

- C0022408: CSP/PT — condition in which there is a deviation from or interruption of the normal structure or function of the joints. A joint is where two or more bones come together, like the knee, hip, elbow or shoulder. Joints can be damaged by many types of injuries or diseases. Arthritis or simply years of

[1] `uts.nlm.nih.gov`

Table 6.1: Sample characteristics

Characteristic	Population	Sample
Words	19232	1190
Unique words	2416	385
Unique words as %	12.6	32.4

Table 6.2: Top twenty most frequent tokens in the population and sample

Population	Sample
ca	ca
mets	mets
with	with
lung	lung
liver	liver
to	to
wi	wi
&	cell
and	and
cancer	cancer
of	bone
cell	&
bone	of
brain	breast
colon	metastatic
metastatic	pancreatic
-	brain
prostate	colon
bladder	squamous
renal	renal

use may cause a joint to wear away. This can cause pain, stiffness and swelling. Over time, a swollen joint can become severely damaged. ...

- C0029408: CSP/PT — noninflammatory degenerative joint disease occurring chiefly in older persons, characterized by degeneration of the articular cartilage, hypertrophy of bone at the margins, and changes in the synovial membrane, accompanied by pain and stiffness. Osteoarthritis is the most common form of arthritis. It causes pain, swelling and reduced motion in your joints. It can occur in any joint, but usually it affects your hands, knees, hips or spine. ...

I converted SNOMED CT code 396275006 to MetaMap identifier C0029408. This one-to-many encoding occurred 28 times on 300 concepts. Appendix C includes these 28 cases.

I coded the texts using MetaMap version 2011 (data 2011AA) and the MetaMap options "-R SNOMED -I". I manually entered each description and selected the highest scoring MetaMap coding. When multiple codings scored equally, the first coding output was selected.

MetaMap correctly coded 48 texts (16%) and my ACC method correctly coded 90 texts (30%), see Table 6.3. Confidence intervals for 95% confidence were calculated using the normal approximation method of the binomial confidence interval [83]. All confidence intervals are less than +/- 0.052 or 5.2% demonstrating a statistically significant difference. McNemar's test [92] was applied to further assess statistical significance. Statistical significance is $p \leq 4.6^{-6}$.

Table 6.3: Evaluation results summary

Test	ACC Method	Accuracy (%)
Precision	MetaMap	16.0
Recall	Developed	85.6
Precision	Developed	30.0

6.4 Discussion

My ACC method outperformed MetaMap (30% vs. 16%). There are likely two main reasons for this performance difference. First, my ACC method uses more

complex linguistic structures that inform the ACC algorithm when to merge words. For example, adjective and noun modifiers in a noun phrase are handled prior to a prepositional phrase that modifies said noun phrase. Second, my ACC method's merge operation discards concepts not descending from all pre-merge concepts. This quickly excludes weak semantically-matching concepts. Consequently, scoring functions need only rank (or differentiate) between several concepts (e.g., 10) rather than a large number (e.g., 100). With regard to real use, such as use within a clinical workflow, my ACC method's performance is likely too weak. However, my ACC method may be applicable under certain conditions such as those of this thesis.

6.4.1 Error analysis

I analyzed the first 100 texts from the second evaluation (precision) to quantify my method's ACC errors. ACC errors were categorized by source[2] such as a spelling mistake. The sources and their definitions are:

- Multi-source: an ACC error caused by multiple sources excluding unknown (e.g., spelling and parsing error)
- Unrecognized terms: an ACC error due to unrecognized terms such as abbreviations (e.g., mets for metastasis)
- Spelling: an ACC error caused by a spelling mistake
- Unknown: an ACC error without a known source (e.g., a difficult text to code)
- Concept scoring: an ACC error in the scoring heuristic. The correct code was found but was not ranked first.
- NLP error (machine learning): an ACC error due to machine learning. A correct and complete input resulted in an incorrect NLP output.
- Linguistic: an error caused by incomplete linguistic knowledge, particularly morphological knowledge (e.g., cecum/cecal)
- Implied or missing information: an ACC error due to implied or missing information (e.g., "breast with bone mets" implying "breast cancer...")

[2]Error sources may not be conclusive/concrete since modifying software components and adding local terms to the terminology could leave an error unresolved due to the interdependence between software components, algorithms and terminology data.

Table 6.4 summarizes the error source frequencies. Spelling mistakes and unrecognized terms caused about half of the errors.

Table 6.4: Sources of ACC error

Source	Frequency (%)
Multi-source	17
Unrecognized terms	17
Spelling	12
Unknown	6
Concept scoring	5
NLP error (machine learning)	5
Linguistic	4
Implied or missing information	2

Stanfill et al. [117] do not report specific error sources but make general mention of NLP difficulties. Recent works (Secion 2.11.2) report parsing errors, missing terms, linguistic errors, spelling mistakes and errors specific to each ACC method. The error sources observed in the second evaluation match with errors reported by previous research efforts.

6.4.2 Precision coding examples

Below are three examples where my ACC method assigned the correct code to a text and MetaMap did not. Each example contains the text that was encoded followed by the correct SNOMED CT/MetaMap codes, my ACC method's coding and MetaMap's coding.

Text: breast ca with lung mets

- Correct codes: 254837009 / C0006142 (Malignant neoplasm of breast)
- Assigned: 254837009 - Malignant tumor of breast
- MetaMap: C0006141 - Breast

Text: non-amall cell lung ca with brain mets

- Correct codes: 254637007 / C0007131 (Carcinoma, non-small-cell lung)
- Assigned: 254637007 - Non-small cell lung cancer

- MetaMap: C0006104 - Brain

Text: posterior pharynx cancer

- Correct codes: 187709007 / C0555271 (Malignant neoplasm of posterior pharynx)
- Assigned: 187709007 - Malignant neoplasm of posterior pharynx
- MetaMap: C0153405 - Malignant neoplasm of pharynx

6.5 Summary

My ACC method encodes linguistic structures by encoding tokens and manipulating token-level encodings by mapping linguistic structures to topological operations in SNOMED CT. This differs from previous ACC methods by focusing on token-level encodings rather than phrase-level encodings.

My ACC method performed well in a recall test and performed significantly better than MetaMap in a precision test. Most ACC errors were attributed to spelling mistakes and unrecognized terms. Improved ACC could impact many biomedical domain services such as search, data communication or patient record classification. As a concrete example, I incorporated my ACC method into my solution for a medical IE task described in Chapter 7. For this task, my ACC addresses the biomedical NLP difficulties of language diversity and the need to incorporate medical domain knowledge during processing.

Chapter 7

Sentinel Event Extraction from Palliative Care Consult Letters

Sentinel events may provide usable information about disease trajectory and consequently affect clinical recommendations for patients and families. These events are likely important in palliative patient care and to patient quality of life. This chapter describes an extension and evaluation of my biomedical NLP system that processes palliative care consult letters and extracts sentinel event information. The evaluation validates my biomedical NLP and IE system and also reveals areas for improvement. Given the IE task's uniqueness, this chapter is also a novel research contribution.

7.1 Software system description

Figure 7.1 depicts information flow through my software system. The general idea is to pre-process input text (consult letters) into categories, apply NLP and ACC to each category's text and extract sentinel event information from NLP/ACC output. Most sentinel event information is extracted by modeling the input (consult letters) as features and training a classifier on these feature models. Some sentinel event information is extracted directly from NLP/ACC output. The subsections below explain each processing step identified as boxes in Figure 7.1.

7.1.1 Pre-processing

Pre-processing categorizes the input text into two main categories, *present* and *history*. The history category contains data relevant to the current case but occurring

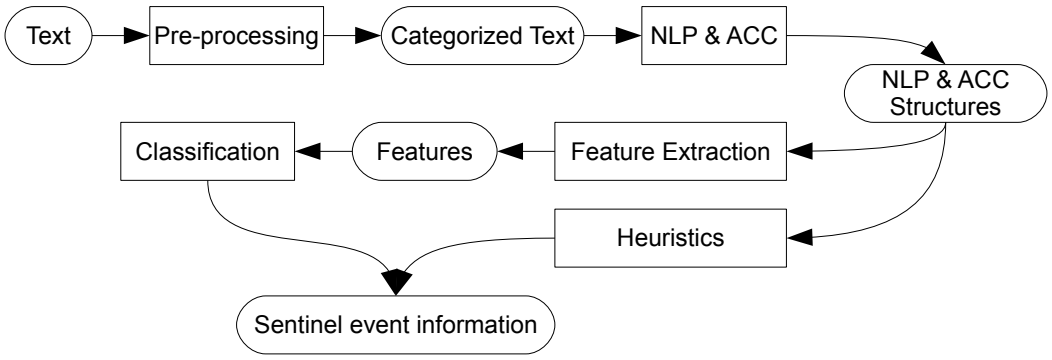

Figure 7.1: Information flow through the sentinel event IE software system

prior to it. The present category contains temporally immediate data. This separation corresponds to a high-level separation implicitly conceptualized by several palliative care physicians (personal communication, 2011). It also partially corresponds to the temporal dichotomy present in the sentinel events (e.g., history of/recent venous thromboembolism).

Input text is also separated into *header* and *allergies* categories. The header category contains information occurring between the text's beginning and its first section title. Header content is generally phrase-based and is potentially difficult to parse (e.g., due to a lack of punctuation). By creating a separate header category, header content parsing errors do not affect other categories' content.

Medications provide important clues for assessing a patient's condition and inferring sentinel event information (personal communication, 2011). Patients may be allergic to certain medications. Without context, software systems could misinterpret an allergy as a medication course. For example, a patient noted as allergic to an antibiotic could be interpreted as having an infection without the allergy context. Separating allergy information from general content reduces medication ambiguity.

Palliative care consult letters include sections such as *previous medical history* and *allergies*. These sections occur inconsistently across consult letters. For example, one consult letter may include *past medical history* and another *previous medical history*. In some cases, section titles change dramatically or disappear. A pattern-based approach mapped consult letter section titles to the four previously described categories (e.g., header).

A regular expression identified section titles. Section titles were assumed to be 40 characters or less of uppercase text. If a section title contained *history* then its content

was categorized as historical. If a section title contained *allerg* then its content was categorized as allergy related. Header content was identified by its location in the input text. All other content was categorized as *present*.

Pre-processing associated lines of text (content) to each category. Manual inspection of several consult letters revealed that lines separated linguistically independent items such as phrases or sentences. For example, lines often separated list items. Unfortunately, sentences sporadically crossed new line boundaries. A pattern-based approach corrected for text crossing new line boundaries. If a line with more than 40 characters ended without terminating punctuation (e.g., a period) and the next line had more than 40 characters and began without a capital then the two lines were stitched into one.

7.1.2 NLP and ACC

The software developed in the previous chapters, including the additions and modifications discussed below, process section content output from the pre-processing step. NLP is augmented with negation detection and ACC is augmented with medication brand name terms.

Medication brand names

Many consult letters include medication brand names such as Tylenol rather than generic names such as Acetaminophen. As discussed in Section 7.1.1, medication information is important to sentinel event IE. Since medication brand names are absent from SNOMED CT, these were added to the ACC terminology via my local terminology extension described in Step 1 of Section 6.2.1. This implies binding additional terminology (e.g., Tylenol) to the appropriate SNOMED CT concept. Concept and term bindings may be extracted from RxNorm[1]. RxNorm provides normalized clinical drug names and links these names to "vocabularies" such as SNOMED CT. Drug brand names were extracted following Merrill et al.'s method [85].

Negation

Negation identification allows a NLP system to differentiate between non-negated and negated statements. For example, negation identification differentiates between

[1] `www.nlm.nih.gov/research/umls/rxnorm/`

cancer and *no cancer*.

NegEx [18] is an algorithm for identifying negated statements in English biomedical text. NegEx locates negation phrases (e.g., not) in biomedical text based on a manually constructed list. Negation identification in palliative care consult letters follows from NegEx. Chapman et al. [18] found that several common negation phrases capture a large portion of all negation phrases. Table 7.1 contains the top five negation phrases and their frequencies. A script counted NegEx negation phrases in palliative care consult letters. The top three non-verb phrases (no, not, without) are employed to locate negation in palliative care consult letters. The phrases were selected to provide 95% coverage by frequency of the non-verb negation phrases. The top three non-verb phrases provide 95.2% coverage.

Table 7.1: Common negation phrases in palliative care consult letters

Negation Phrase	Frequency (%)
no	63.9
not	30.2
denies	1.9
without	1.1
to exclude	0.6

My negation identification implementation acts on dependency trees whereas NegEx acts on token sequences. Given the additional complexity of dependency trees, I implemented limited negation identification. NegEx applied at the token level approximates which tokens are negated. That is, NegEx identifies a negation phrase and a potential target, allowing a maximum of five tokens between the negation and the target. My method is similar in that it identifies a negation phrase (word) and target (noun phrase). When the trigger words no, not and without are located in a dependency parse, negation is applied to the noun phrase associated with the parse. The noun phrase is found by following the trigger word's dependency edge to its head. If the head is not a noun phrase then negation is not applied. Negations affected features during feature-based IE.

7.1.3 Feature-based IE

Most sentinel event IE is treated as a classification task and is subject to this subsection's sentinel event IE method. Each palliative care consult letter is modeled by

features and is associated with sentinel event information (e.g., has-sepsis/no-sepsis). A classifier learns the association between features (palliative care consult letter models) and labels (sentinel event information). A trained classifier infers omitted labels from a consult letter's features. A classifier is trained for each label set (Section 7.2).

Features are extracted from consult letters as follows. NLP converts text to NLP structures (e.g., dependency graphs). ACC augments these NLP structures with SNOMED CT codes (for NLP/ACC structures see Chapter 6). Rather than transforming all SNOMED CT encoded linguistic structures into features, linguistic structures are filtered. The post-filter linguistic structures are noun, adjective, preposition and verb phrases. From these linguistic structures, the highest ranking associated SNOMED CT code (if any) becomes a consult letter feature. Each feature is also contextualized by a category assigned during pre-processing (header, allergy, history, present). For example, the noun phrase *brain metastases* with an associated highest ranked SNOMED CT code of 4225005, and categorized in *history* during pre-processing, would be encoded as the feature *history.4225005 = True*. In negation cases the feature value is *false* and omitting the feature implies its absence in the consult letter.

Related work (Secion 2.4) demonstrated consistent use and success of SVMs. Consequently, SVMs were trained to extract sentinel event information. A grid search of SVM parameter space establishes the best parameters for each SVM. A grid search is less likely to be misled by local minima and maxima compared to a directed search. In some cases, SVMs overfit the training data which may result in all testing instances classified as the same category, or fewer categories than expected. For example, *brain metastases* occurs 19 times in 200 consult letters even though an SVM classifies all consult letters as the dominant non-metastases category. When an SVM overfits training data, a decision tree classifier (C4.5, J48) [2] is used instead. The C4.5/J48 decision tree performs pruning in an attempt to not overfit data. SVMs may overfit data due to noise. This is likely the case for sentinel event IE given that consult letters are modeled with approximately 16,000 features. Decision trees are sensitive to individual features and process features sequentially avoiding some pitfalls experienced by SVMs. For example, in this thesis work SVMs failed to identify true positive examples if few of these examples existed in the data set.

7.1.4 Pattern-based IE

Sentinel event information such as creatinine levels (integer values) may not be inferred by trained classifiers. I consequently used pattern-based IE to extract date and integer values. The pattern-based approach is similar regardless of the information being extracted. The approach first locates a key token or SNOMED CT code such as creatinine or 15373003. Dates to the left and right of the key are identified and marked, from tokens closest to the key to those farthest away. Dates are identified by month. Year and day information is located thereafter. Locating dates excludes numeric date values such as the day (e.g., 21) or the year (e.g., 2009) from being falsely identified as, for example, creatinine levels. Once dates are identified and marked, integer values are identified. The closest date and integer value are bound to the key. In the example sentence "On March 16, 2008, and on March 17, 2008, electrolytes were normal, creatinine was 202.", creatinine would be bound to the level 202 and the date March 17, 2008.

7.2 Evaluation

I evaluated my software's ability to extract sentinel event information against physician collected and physician extracted sentinel event information. Physicians[2] collected sentinel event information prior to dictating palliative care consult letters. A physican (Dr. Thai) extracted sentinel event information years after initial data collection and dictation had commenced. Figure 7.2 depicts the data creation timeline and data use during evaluation.

Physicians collected sentinel event information during a previous study by Thai et al. [121, 122]. In this study, palliative care consult physicians collected sentinel event information during their first consultation with a patient by completing the form in Appendix D. Data collection was performed specifically for the study and is not routine. A data set of palliative care consult letters paired with their collected sentinel event information was constructed by randomly selecting 200 consult letters from Thai et al.'s data set. A support staff member manually anonymized each consult letter by replacing identifying information such as names with anonymous placeholders (e.g., Abcname).

A *physician (Dr. Thai) manually extracted* sentinel event information for 15 ad-

[2]Physicians providing palliative care as part of the palliative care team.

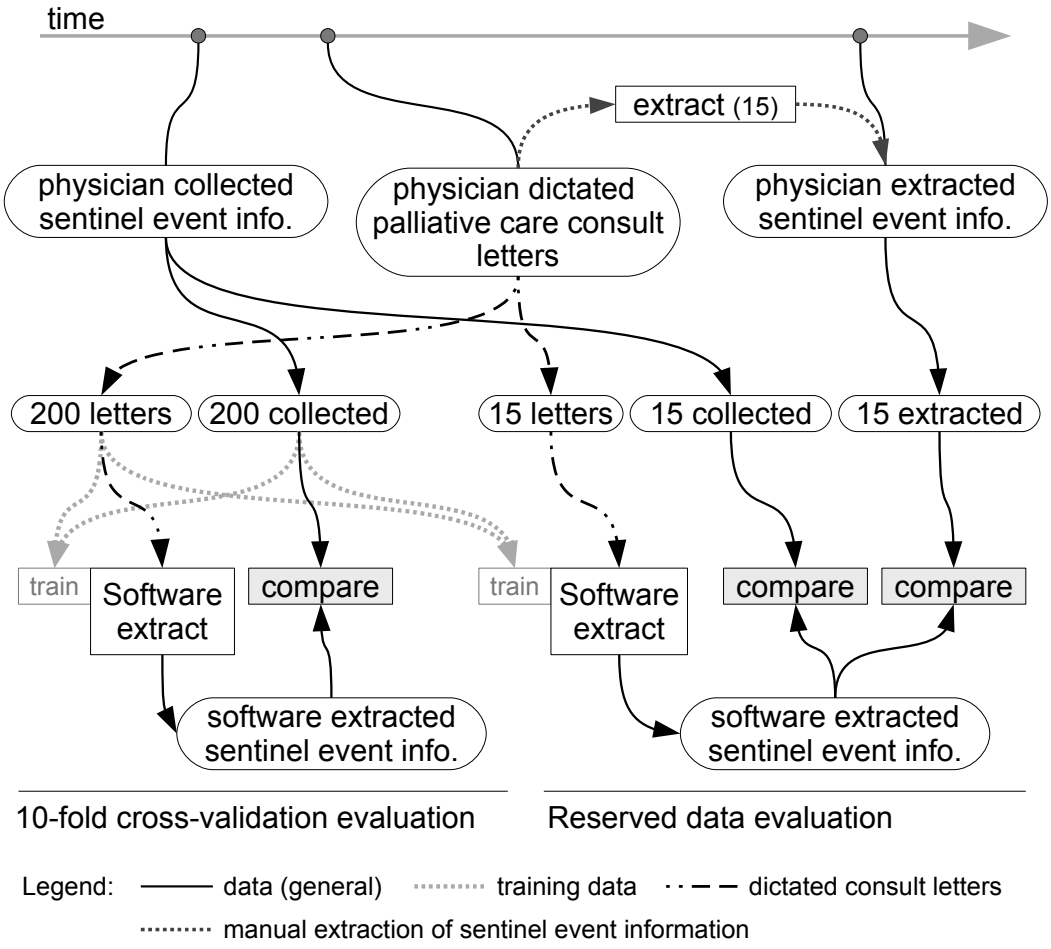

Figure 7.2: Data creation timeline and use during evaluation

ditional consult letters. Time constraints limited manual extraction to 15 consult letters. These 15 consult letters formed the test set for the evaluation described in Section 7.2.2 and was key to a knowledge transfer between an expert and myself. The 15 consult letters were manually selected for knowledge transfer purposes.

The final data set consisted of 200 consult letters paired with their collected sentinel event information and an additional 15 consult letters paired with physician extracted and physician collected sentinel event information. Section 7.2.1 describes the use of the 200 palliative care consult letters paired with their collected sentinel event information for training and testing (Figure 7.2, 10-fold cross-validation). Section 7.2.2 describes the use of the 200 palliative care consult letters paired with their collected sentinel event information for training and the 15 reserved consult letters for testing (Figure 7.2, reserved data).

I restricted IE to sentinel event information generally recorded in a consult letter rather than all data collected using the Appendix D form. These restrictions were established during a personal communication with Dr. Thai. For example, certain dates recorded during data collection are typically omitted from consult letters since these dates are clinically irrelevant for the purposes of the consult letter. For example, a VTE onset date may not be relevant in the case where a patient died in hospital of causes unrelated to VTE. The data used in this evaluation are (extraction output in square brackets):

1. Dyspnea [yes/no]
2. Dyspnea at rest [yes/no]
3. Delirium [yes/no]
4. Brain metastases (leptomeningeal) [yes/no]
5. Sepsis [yes/no]
6. Infection [yes/no]
7. Infection site [sites include urinary tract/intra-abdominal/skin]
8. Chest infection, aspiration related [yes/no]
9. IV antibiotic use [yes/no]
10. IV antibiotic use response [no/partial/complete]

11. Oral antibiotic use [yes/no]
12. Oral antibiotic use response [no/partial/complete]
13. Serum creatinine [integer; pattern-based IE]
14. Serum creatinine date [date; pattern-based IE]
15. Dysphagia [yes/no]
16. Previous VTE [yes/no]
17. VTE [yes/no]
18. ICU Stay [yes/no]
19. ICU length of stay in days [integer; pattern-based IE]

7.2.1 10-fold cross-validation

The first evaluation compared my software's ability to extract sentinel event information to sentinel event information collected during the patient's first consultation with a palliative care physician. This comparison used a 10-fold cross-validation method. In 10-fold cross-validation, data is randomly split into 10 evenly sized groups. Each group acts as test data while the remaining data (9 groups) act as software training data. For pattern-based IE, the approach was evaluated directly on the 200 consult letters (no training/testing splits). This evaluation used 200 palliative care consult letters paired with their collected sentinel event information for training and testing.

Results appear in Table 7.2. Confidence intervals for 95% confidence were calculated using the normal approximation method of the binomial confidence interval [83]. My software's average accuracy was 73.6% with a range of 37.5% to 95.5%. The confidence interval average was +/- 5.5%.

7.2.2 Reserved data

The second evaluation compared my software's ability to extract sentinel event information to physician extracted and physician collected information on 15 reserved consult letters. Training data consisted of the 200 data pairs used in Section 7.2.1. For pattern-based IE, the approach was evaluated directly on the 15 consult letters (no training).

Table 7.2: Sentinel event extraction 10-fold cross-validation results

Sentinel Event Info.	Accuracy (%)	C.I. +/−	Avg. f-msr.
Dyspnea	79.5	5.6	70.0
Dyspnea at rest	78.5	5.7	66.3
Delirium	81.0	5.4	79.5
Brain metastases	86.0	4.8	58.2
Sepsis	78.5	5.7	54.2
Infection	50.5	6.9	49.9
Infection site	53.0	6.9	26.5
Chest infection, aspiration related	95.5	2.9	75.1
IV antibiotic use	70.5	6.3	60.3
IV antibiotic use response	57.5	6.9	29.1
Oral antibiotic use	69.5	6.4	45.0
Oral antibiotic use response	68.5	6.4	28.9
Serum creatinine	55.5	6.9	-
Serum creatinine date	37.5	6.7	-
Dysphagia	72.5	6.2	58.3
Previous VTE	93.0	3.5	48.8
VTE	94.0	3.3	49.3
ICU Stay	87.0	4.7	56.1
ICU length of stay in days	90.5	4.1	-
Average	73.6	5.5	53.5

Results appear in Table 7.3. Confidence intervals for 95% confidence were calculated using the normal approximation method of the binomial confidence interval [83]. My software's average accuracy was 78.9% with a range of 53.3% to 100% when evaluated against physician extracted sentinel event information. When evaluated against physician collected sentinel event information, my software's accuracy was 71.9% with a range of 46.7% to 100%. Confidence intervals fell between +/- 0% and +/- 25%.

Table 7.3: Automated sentinel event IE accuracies on physician extracted and collected data

Sentinel Event Info.	On Extracted	On Collected
Dyspnea	86.7	86.7
Dyspnea at rest	80.0	73.3
Delirium	73.3	66.7
Brain metastases	100.0	100.0
Sepsis	73.3	73.3
Infection	40.0	53.3
Infection site	66.7	53.3
Chest infection, aspiration related	80.0	86.7
IV antibiotic use	66.7	60.0
IV antibiotic use response	53.3	46.7
Oral antibiotic use	73.3	53.3
Oral antibiotic use response	73.3	46.7
Serum creatinine	100.0	66.7
Serum creatinine date	100.0	60.0
Dysphagia	66.7	66.7
Previous VTE	93.3	100.0
VTE	100.0	100.0
ICU Stay	86.7	86.7
ICU length of stay in days	86.7	86.7
Average	78.9	71.9

7.3 Discussion

My software system extracts selected sentinel event information with average accuracies of 73.6% (range 37.5% - 95.5%), 78.9% (range 53.3% - 100%) and 71.9% (range 46.7% - 100%), see Table 7.2 and Table 7.3 respectively). These results are similar to

state-of-the-art results reported by Stanfill et al. [117] (accuracy range, about 60% - 100%; precision range, about 10% - 100%; recall range, about 25% - 100%).

7.3.1 Frequency

Table 7.4 contains the sentinel event information frequencies on physician extracted and collected data. Low frequency items are difficult for classifiers to learn. For example, skin infection with a frequency of 1.5% in the 10-fold cross-validation implies that the classifier has, in the best case, two training examples and one testing example. Low frequency sentinel event information often leads an SVM to overfit training data. When analysis suggested that SVMs were overfitting the training data, decision trees (with pruning) were applied. The better performing classifiers was selected. Decision tree use generally corresponds to low frequency information, frequencies below 20%. Table 7.4 also indicates decision tree use. Much of the sentinel event information occurs infrequently and is likely difficult for classifiers.

7.3.2 Information gap

Inconsistencies exist between physician collected and physician extracted sentinel event information. Table 7.5 quantifies these inconsistencies. For example, collected and extracted delirium information matched 86.7%, or differed by (gap of) 13.3%.

Several factors may contribute to the observed gap, such as (personal communication with Dr. Thai 2012, except for first item):

- *Data collection or extraction error*: Physicians may have erred when collecting or extracting sentinel event information. This could be as simple as checking a wrong box or as complex as making a medical error.

- *Quality*: Palliative care consult letters may vary in quality. For example, residents may not produce a consult letter of the same quality as an experienced physician.

- *Research artifact*: Specific sentinel event information was collected for research purposes. This information may rarely, or never, be included in a palliative care consult letter. For example, many dates are excluded.

- *Relevance*: Sentinel event information may be excluded from palliative care consult letters because the information may be of secondary importance or could

Table 7.4: Sentinel event information frequencies, as percentages, on physician extracted and collected data (* indicates use of decision tree)

Sentinel Event Information	200 collected	15 collected	15 extracted
Dyspnea	29.5	33.3	40
Dyspnea at rest	27	40	33.3
Delirium	36	66.7	60
Brain metastases *	8.5	0	0
Sepsis	21.5	46.7	20
Infection *	43	73.3	73.3
Infection site, chest infection *	18.5	26.7	33
Infection site, intra-abdominal *	10	20	13.3
Infection site, urinary tract *	8.5	13	20
Infection site, skin *	1.5	0	0
Infection site, other *	4	13	0
Chest infection, aspiration related *	6	0	6.7
IV antibiotic use	32.5	40	33
IV antibiotic response, none *	9	20	0
IV antibiotic response, partial *	14	13.3	0
IV antibiotic response, complete *	8.5	6.7	0
Oral antibiotic use	24.5	46.7	26.7
Oral antibiotic response, none *	4.5	20	0
Oral antibiotic response, partial *	12	20	6.7
Oral antibiotic response, complete *	7	6.7	0
Serum creatinine	98	100	73.3
Serum creatinine date	98.5	100	66.7
Dysphagia	28	20	20
Previous VTE *	4.5	0	6.7
VTE *	2.5	0	0
ICU Stay	6.5	6.7	6.7
ICU length of stay in days	6.5	6.7	6.7

Table 7.5: Match between physician collected and physician extracted sentinel event information, over 15 consult letters

Sentinel Event Info.	Match (%)	Gap (%)	Small Gap
Dyspnea	93.3	6.7	√
Dyspnea at rest	86.7	13.3	√
Delirium	86.7	13.3	√
Brain metastases	100	0	√
Sepsis	73.3	26.7	
Infection	73.3	26.7	
Infection site	66.7	33.3	
Chest infection, aspiration related	93.3	6.7	√
IV antibiotic use	93.3	6.7	√
IV antibiotic use response	60.0	40.0	
Oral antibiotic use	66.7	33.3	
Oral antibiotic use response	53.3	46.7	
Serum creatinine	66.7	33.3	
Serum creatinine date	60.0	40.0	
Dysphagia	100	0	√
Previous VTE	93.3	6.7	√
VTE	100	0	√
ICU Stay	100	0	√
ICU length of stay in days	93.3	6.7	√

impede readability (e.g., unnecessary increase in cognitive load). In particular, infection, antibiotic use and antibiotic response (i.e., most large gaps) were collected after seeing the patient, getting a measure of treatment response and examining antibiotic use printouts for the preceding two to four weeks. These details would generally interfere with a consult letter's readability.

Assuming data collection and extraction error is minimal then the gap represents an information gap between physician collected and consult letter sentinel event information. Figure 7.3 illustrates this information gap. In other words, the information gap implies a disagreement between physician collected and extracted sentinel event information or a difficulty in extracting sentinel event information (e.g., information is missing). In context of software extracted sentinel event information, the information gap may suggest which extraction tasks are more difficult than others and where the software is likely to perform poorly relative to physician collected or extracted information. The latter assumes that software performance will at best match that of a physician.

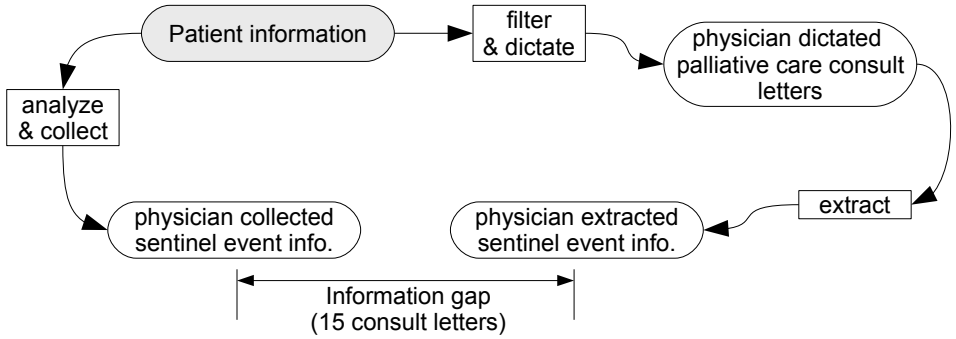

Figure 7.3: Information gap over 15 palliative care consult letters

7.3.3 Comparison to physician collected information

The information gap potentially explains some performance decreases when comparing software extracted sentinel event information to physician collected information on reserve data. A linear regression (Equation 7.1) produces a $p \leq 3.8^{-5}$. Thus, there is a correlation between the information gap and software performance for physician collected information (reserve data). This implies that the information gap limits performance on reserve results. Given the correlation's p-value, it is likely the dominant

factor affecting this performance.

$$collected\ reserve\ result = M \times gap + C \tag{7.1}$$

If the reserve data is a representative sample of the 10-fold data then the information gap is likely present over all data. If the information gap is present over all data then it may also explain 10-fold results. A sample size of 15 is not sufficient (confidence intervals of up to 25%) to simply extend the information gap to the 10-fold data. However, a linear regression (Equation 7.2) between software performance on physician collected reserve results and 10-fold results produces a $p \leq 4.9^{-5}$. Thus, the physician collected reserve data is similar to the 10-fold data when using my software as a similarity metric. It is likely that the information gap seen in the reserve data is present in the same form in the 10-fold data.

$$collected\ reserve\ result = M \times ten\_fold + C \tag{7.2}$$

A manual inspection of the linear regression (Equation 7.2) identifies three potential outliers. The potential outliers are the numerically identified points in Figure 7.4. The numbers correspond to those listed in Section 7.2. Of the three potential outliers, serum creatinine date appears to be an outlier. A pattern-based approach extracts serum creatinine date information. Better performance of serum creatinine on reserve consult letters implies that these consult letters better match pattern expectations.

Despite some differences, the reserve data is likely representative of the 10-fold data. As in the case of physician collected information (reserve data), the information gap should explain my software's performance decrease over sentinel event information on 10-fold data. A linear regression (Equation 7.3) produces a $p \leq 0.00013$. Thus, there is a correlation between the information gap and software performance on 10-fold data. The information gap is likely the dominant factor affecting performance on 10-fold data.

$$ten\_fold = M \times gap + C \tag{7.3}$$

7.3.4 Comparison to physician extracted information

My software performs better when compared to physician extracted sentinel event information rather than physician collected information (reserve data). This difference is nearly significant with $p \leq 0.076$ given a paired Wilcox signed rank test [113].

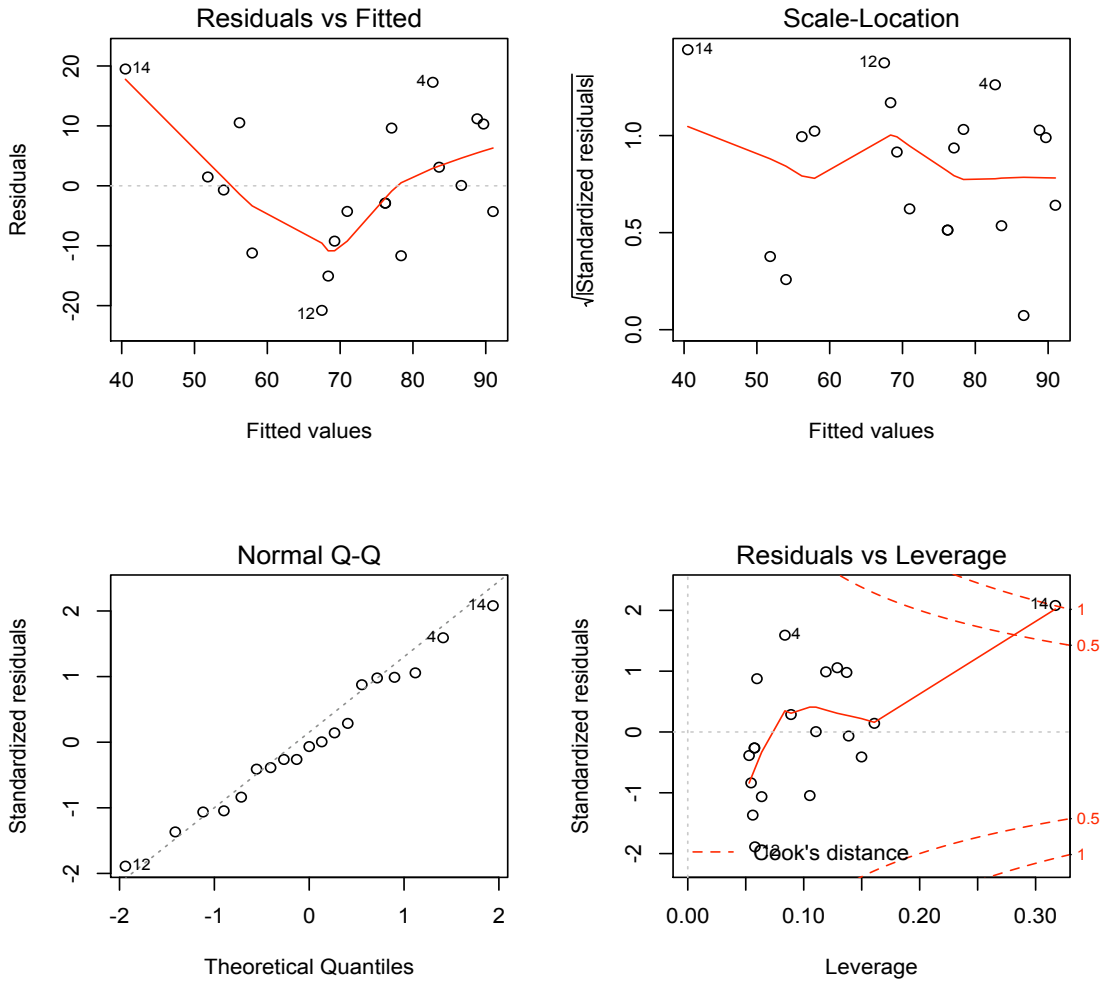

Figure 7.4: Outlier visualization for a linear regression between 10-fold results and physician collected results

Despite training and patterns targeting collected data, my software performs more closely to a physician extracting sentinel event information from a consult letter.

The information gap should not explain software results on physician extracted information since the physician (Dr. Thai) and software operate from the same source: palliative care consult letters. In other words, an information gap no longer exists. A linear regression (Equation 7.4) does not produce a correlation ($p \leq 0.2$). This suggests that poor software performance is due to software issues rather than an information gap.

$$extracted\ reserve\ result = M \times gap + C \tag{7.4}$$

Table 7.6 presents software extraction performance on physician extracted sentinel event information, ordered worst to best. Poor performance generally occurs on infection, antibiotic and antibiotic response related sentinel event information. Section 7.3.2 discusses how this information is filtered from the palliative care consult letter to promote consult letter readability. In other words, a physician infers these sentinel event information during extraction from consult letters. Since my software lacks strong deductive capabilities, performance drops across infection, antibiotic and antibiotic response are expected.

Although performance on dysphagia and delirium is within expectations (e.g., 10-fold results), the information gap suggests that software performance should be better than recorded. Palliative care consult letters contained sufficient dysphagia and delirium information that collected and extracted sentinel event information matched. Because of my software's complexity, the exact nature of these errors is unknown.

7.3.5 Clinical value of results in context of quality assurance

It is difficult to gauge my software's value as a quality assurance tool in palliative care. My software's value depends on the software's performance as well as administrative and clinical factors. For example, how would a quality assurance tool integrate into palliative healthcare workflow? Assuming it could be integrated then would the tool have a positive affect on palliative healthcare professionals and patients? Excluding these considerations, my software's performance (e.g., $\geq 80\%$ vs. physician) on the sentinel event information below is likely sufficient for use. Even this claim depends on context and may be inaccurate given final assessments performed in clinical context.

Sentinel event information for which extraction performance is greater than or equal to 80% versus a physician and positive training examples occur with a frequency

Table 7.6: Software extraction performance on physician extracted sentinel event information, ordered worst to best

Sentinel Event Info.	On Extracted (%)
Infection	40
Recent IV antibiotic use response	53.3
Infection site	66.7
Recent IV antibiotic use	66.7
Dysphagia	66.7
Delirium	73.3
Sepsis	73.3
Oral antibiotic use	73.3
Oral antibiotic use response	73.3
Dyspnea at rest	80
Chest infection aspiration related	80
Dyspnea	86.7
ICU Stay	86.7
ICU length of stay in days	86.7
Previous VTE	93.3
Brain metastases	100
Serum creatinine	100
Serum creatinine date	100
VTE	100

of 20%:

- Dyspnea
- Dyspnea at rest
- Chest infection aspiration related
- Serum creatinine
- Serum creatinine date
- Dysphagia

7.4 Summary

I built on my previous research contributions and created software that extracts sentinel event information from palliative care consult letters. I extended my contributions by pre-processing (e.g., categorizing text as present or past) consult letters and post-processing NLP and ACC structures (e.g., feature-based classification). I also incorporated negation and medication brand names during NLP and ACC.

I evaluated my IE system against physician collected and physician extracted sentinel event information. Software performance on physician collected information is limited by the observed information gap. The information gap represents the disagreement between physician collected and extracted sentinel event information (e.g., values exist but disagree) and the difficulty in extracting sentinel event information from consult letters (e.g., information is missing or implied). Sentinel event IE performance decreases as the information gap grows. Software performance on physician extracted information could not be explained by the information gap. Instead, software performance is generally weaker for inferred sentinel event information. Overall, software extraction accuracy averaged 74.8% and ranged from 37.5% to 100%. My software's performance is similar to state-of-the-art results reported by Stanfill et al. [117].

Chapter 8

Conclusions and future work

I proposed several novel directions for medical language processing and applied my biomedical NLP software, implementing these new directions, to palliative care consult letters. I developed a tokenization framework, a method for cross-domain POS tagging and an automated clinical coding method. I also compared two dependency parsers on imperfect POS tagging on biomedical corpora. A summary of my research contributions follows.

Tokenization

Biomedical tokenization is often problematic due to atypical use of symbols and other irregularities in biomedical text. A lack of guidance how to modify and extend existing tokenizers to new domains, inconsistencies between tokenizers and tokenizer idiosyncrasies limit tokenizer use. I developed the token lattice design pattern and the associated token transducer identification guidelines. The token lattice design pattern and guidelines provide consistency in tokenizer creation, result in an extensible tokenizer, and are expected to produce consistent token output across differing tokenizer implementations. The token lattice design pattern is also expected to apply to domains other than the biomedical domain.

The token lattice design pattern addresses a tokenizer's input, output and internal components. The components are a token lattice and lattice constructor, a best lattice-path chooser and token transducers. The token lattice design pattern provides a framework where biomedical text processing components such as named entity recognizers can interact cooperatively (as token transducers). A tokenizer implementing the token lattice design pattern and guidelines was evaluated as equivalent

to a leading biomedical tokenizer. Given the results, the token lattice design pattern is a viable approach to tokenization. The results also demonstrate that ambiguous tokenizations can be disambiguated through POS tagging. In doing so, POS tag sequences and training data have a significant impact on proper text tokenization.

POS tagging

There is limited training data for training biomedical NLP systems. Consequently, biomedical NLP systems may be trained on non-biomedical data and applied to biomedical text, negatively impacting performance. This is the case for POS tagging. I developed a cross-domain POS tagging method (TcT) that outperforms several existing methods in cross-domain POS tagging. TcT may improve performance for situations in which only non-biomedical training data is available. TcT also provides insight on how to adapt existing algorithms to restricted training conditions.

TcT ignores previously assigned POS tags to avoid feedback errors and assigns POS tags given the current token, one previous token and one following token. TcT performed better than three frequently used biomedical text taggers (mxpost, TnT and Brill) in cross-domain POS tagging. TcT differs from the work to date that has focused on retraining existing POS taggers or training taggers on large data sets such as web-based data sets.

Parsing

Parsers rely on correct POS tags during processing. MST and Malt are state-of-the-art dependency parsers that have been evaluated using perfect POS tag input. Unlike previous evaluations, I tested these two parsers on imperfect POS tag input and compared their results. I found that Malt was more resilient to imperfect POS tags. My comparison may help developers choose a parser and help researchers address performance weaknesses manifested during evaluation.

Malt was adapted to the biomedical domain. Since Malt's adaptions do not rely on text characteristics unique to biomedical text, the adaptions may improve dependency parsing in other domains. I called this adaption Mediware and its performance was roughly equivalent to MST's prior to comparing parsers on imperfectly POS tagged biomedical text. MST was most affected by imperfect POS tags. I attributed MST's drop in performance to two POS categories (verbs and adjectives) where MST had more potential for performance loss than Malt. I also attributed Malt's resilience

to POS tagging errors to its use of a rich feature set and a local scope in decision making. These latter two properties compensate for and ignore POS tagging errors.

Automated clinical coding

Standardized biomedical terminologies such as SNOMED CT are promising tools for biomedical computing. A difficulty is correctly mapping biomedical text to standardized terminology entities (i.e., ACC). I developed an ACC method that maps biomedical text to SNOMED CT. My method performed well in a recall test and performed significantly better than MetaMap in a precision test. Given standardized terminology's use in biomedical domains (e.g., search, data communication, record classification), my ACC improvements could impact many biomedical services.

My ACC method encodes linguistic structures by encoding tokens as SNOMED CT concepts and by manipulating token-level encodings by mapping linguistic structures to topological operations in SNOMED CT. My ACC differed from previous ACC methods by focusing on token-level encodings rather than phrase-level encodings.

Sentinel event IE

Having software tools aid palliative care teams in identifying sentinel event information in palliative care could aid palliative care teams and improve patient treatment. For example, software could identify sentinel event information, using the information to find relevant clinical guidelines for palliative care teams or to suggest treatment plans.

I built on my previous research contributions and created software that extracts sentinel event information from palliative care consult letters. My information extraction software incorporates negation and medication brand names during NLP and ACC. My software performance's on physician collected information correlated with the information gap. The information gap represents the disagreement between physician collected and extracted sentinel event information (e.g., values exist but disagree) and the difficulty in extracting sentinel event information from consult letters (e.g., information is missing or implied). Sentinel event information extraction performance decreases as the information gap grows. My software's performance on physician extracted information is weaker for inferred sentinel event information such as a patient's response to antibiotics. Overall, my software's performance is similar to state-of-the-art results.

Summary

The components above and the Punkt sentence segmentation system (Section 2.7) form a biomedical NLP system for sentinel event IE. These components and their composition are a blueprint for building a biomedical NLP and IE system for sentinel event information. Each component was engineered to address biomedical NLP difficulties and knowledge representation issues, while accounting for successful biomedical NLP techniques. The resulting blueprint proposed in this thesis and its components have been empirically evaluated and are evidenced to perform successfully, albeit under specific conditions. The evaluations and conclusions throughout this thesis are expected to generalize beyond the experimental constraints existent during system prototyping.

8.1 Biomedical NLP difficulties

In this thesis, I selected and addressed specific biomedical NLP challenges. These challenges stemmed from biomedical NLP difficulties, consequently my contributions speak to several difficulties. The nature of my contributions regarding biomedical NLP difficulties are as follows.

Character: Biomedical text often differs in character from general text (e.g., newspaper). My tokenizer, POS tagger and parser implement techniques for handling biomedical text's characteristics. For example, my POS tagger is token-focused rather than POS-sequence focused. My tagger was more resilient to syntactic changes across corpora (e.g., ungrammatical nature of some biomedical texts).

Data: A hurdle to system development is accessing appropriate training data. My biomedical NLP system was developed using free publicly available training data. This demonstrates that such training data is effective when developing biomedical NLP systems and, in some cases, may be sufficient.

Design and implementation: To address biomedical NLP design and implementation difficulties, system components were designed to be extensible (e.g., tokenizer) and robust (e.g., selection and design of components to perform well under diverse conditions). Since components from the tokenizer to the ACC method are data-driven, my biomedical NLP system is likely reusable across diverse domains and languages given appropriate training data.

Diversity of language: Biomedical text contains diverse statements having similar or identical semantics. My system demonstrates that ACC using SNOMED CT is a successful means for normalizing textual diversity.

Domain knowledge: There is value in biomedical NLP systems that include and understand domain knowledge. Although primitive, merge operations during ACC use SNOMED CT biomedical domain knowledge to eliminate biomedically unfeasible coding candidates.

Modifiers: Biomedical concept modifiers (certainty, quantitative, degree and temporal) provide valuable information when processing text. My thesis outlines which information may be captured by SNOMED CT and which information should be encoded external to SNOMED CT.

Usability: A concern of biomedical NLP system users is application responsiveness (i.e., speed). My software's NLP components operate in linear time with respect to tokens and support the viability of linear time NLP for biomedical applications.

8.2 Future work

This section discusses several future research directions. These directions are organized by research contribution and are expected to improve sentinel event extraction and software system robustness.

Tokenization

Tokenization would benefit from improved handling of unexpected symbols. Unexpected symbols are unknown to all token transducers and consequently a portion of the tokenizer's input is not segmented. One possibility is generating several potential tokenizations given the unexpected symbol and including these token sequences in the token lattice. Consider the example text "name@company.com" with the unexpected symbol @. Upon encountering this text, several token sequences (space delimited) including "name @ company . com", "name@company . com" and "name@company.com" could be generated and included in the token lattice. Token sequence selection could occur through POS tagging, as described.

Sentence segmentation is a natural extension to token-lattice tokenization. Rather than restricting the token-lattice to sentences, the lattice could be extended across sentences. A sentence ending tag such as "." would identify sentences or sentence like segments. To accomplish the latter, it may be necessary to include whitespace in POS tagging in order to identify sentence segments such as document sections.

Selecting a correct tokenization need not be limited to POS tagging. POS tagging could act as a preliminary filter reducing the number of token sequence possibilities. Syntactic parsing could be used to further score the remaining tokenizations. In this way, tokens are selected using some further measure of grammatical fitness.

POS tagging

Token centric POS tagging would likely improve with the selection of appropriate machine learning techniques. This would relieve the POS tagger of the naive Bayes' feature independence assumption. Furthermore, features such as suffix and capitalization could be extracted from all tokens rather than the current token to be tagged. This would likely help in cross-corpus tagging when unknown tokens share features with known tokens.

Including additional tokens to the left and right of the current token may improve POS tagging. If such inclusions adversely affect performance then a *use-on* strategy could be employed. In a *use on* strategy, the tagger could focus on the current token and consider additional left and right input tokens in low confidence situations. That is, context is used (conditionally) to disambiguate several similarly scored tags but is otherwise ignored. This could be beneficial when tagging clinical documents because these are known to be ungrammatical.

Parsing

Imperfect POS tagging information was generated using a single POS tagger. Employing additional taggers would vary the imperfect POS tagging input available for testing parsers. Additional testing would likely confirm and potentially narrow the causes of parsing error due to imperfect POS tagging.

Malt might be improved through additional parsing restrictions since restrictions often help machine learning systems. A three-pass Malt parser that processes noun phrases, verb phrases (narrowest definition: auxiliary, verb and adverb) and finally full sentences may improve parsing. These restrictions would relieve the parser from

differentiating between all possible dependency edges to differentiating between several restricted sets. Processing noun and verb phrases reduces edge length. Since Malt performs worse on longer dependency edges, this three-pass processing may further improve performance.

Automated clinical coding

Several mechanisms for correcting specific ACC errors are suggested below. Correcting specific errors may also fix multi-source errors. For example, correcting a spelling mistake and an unrecognized term could fix a multi-source error dependent on these latter two errors.

A medically aware spell checker could be run prior to ACC to correct spelling mistakes. Improving the ACC's NLP components would likely reduce NLP (machine learning) and linguistic errors. For example, a larger quantity of training data and training data matching the application domain could improve NLP.

Additional information and context could resolve errors due to missing information. For example, additional information may be required to correctly interpret *breast* in the text "*breast with bone mets*". Missing information may be accessible in previous sentences or phrases in other text data.

Non token-level scoring should account for terminology available at token-level concepts. For the purpose of this paragraph, a merged concept is a concept resulting from a merge operation. An issue occurs when scoring merged concepts. A merged concept's semantics may differ from the original linguistic semantics and pre-merge concept semantics (e.g., by use of synonyms). A merged concept score is adjusted using pre-merge semantics. A fairer scoring system would account for linguistic semantics and semantics from token-level concepts.

ACC should be extended to post-coordinated SNOMED CT expressions. The challenge in doing so will likely be recognizing when to use post-coordination versus when to use pre-coordination, and how to score all expressions fairly.

Sentinel event IE

IE from palliative care consult letters should move towards generalized IE rather than specific IE. This suggests that a better understanding of biomedical knowledge representation is needed and that IE should likely be based on extensible inference and deductive systems rather than specialized IE systems. A knowledge representation

structure that accounts for certainty, quantitative, degree and temporal information in a simple, clear and consistent manner is desirable. A clear internal representation of biomedical information could permit loose coupling between a biomedical text's representation and the deductive systems processing the representation.

8.3 Final remarks

All biomedical NLP systems are dependent on training data. Free public biomedical training data were sufficient for developing my biomedical NLP system (English). This suggests that researchers should switch from creating large training corpora to creating small testing corpora. In other words, there is likely sufficient training data available for biomedical NLP systems, but these systems lack evaluation corpora required to assess their performance over diverse data sets.

There is a general tendency to improve biomedical NLP performance through additional training data. Although successful, such avenues have diminishing returns. How much data is enough? Since "necessity is the mother of invention", developing biomedical NLP under limited conditions should spur innovation. Furthermore, limited conditions force some degree of generalizability on the developed methods because the application's scope is almost guaranteed to differ from the development's limited conditions.

The result of a correct tokenization through POS tagging suggests that biomedical NLP components should influence each other's processing to a greater degree than convention. This does not imply that software component's must be tightly coupled. Instead, it implies that the traditional separation between components and the data that is shared between them should be investigated further.

Greater effort should be placed on constructing biomedical NLP systems that are generalizable rather than specialized. In the future, biomedical NLP systems will likely process data containing rare, perhaps unique, medical situations. By definition, insufficient training data will exist for these cases. Biomedical NLP systems will need to generalize in the greatest sense to effectively process these cases. Advancements in generalizable systems should focus on late-stage processing components (e.g., ACC or IE), where improvements will likely have an appreciable impact.

Overall, this thesis is a blueprint for building a biomedical NLP system for sentinel event IE. This thesis likely applies to NLP and NLP systems in general even though conceptualization and evaluation focused on biomedical domain difficulties

and specific biomedical challenges.

Appendix A

Related publications

1. Neil Barrett and Jens Weber-Jahnke. Applying natural language processing toolkits to electronic health records – an experience report. In James G. McDaniel, editor, *Studies in Health Technology and Informatics*, volume 143 of *Advances in Information Technology and Communication in Health*, pages 441–446. IOS Press, 2009.

2. Neil Barrett and Jens Weber-Jahnke. ehealth interoperability with web-based medical terminology services - a study of service requirements and maturity. *Journal of Emerging Technologies in Web Intelligence*, 1(2):153–160, November 2009.

3. Neil Barrett and Jens Weber-Jahnke. Building a biomedical tokenizer using the token lattice design pattern and the adapted viterbi algorithm. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 473–478. IEEE, 2010.

4. Neil Barrett and Jens Weber-Jahnke. Building a biomedical tokenizer using the token lattice design pattern and the adapted viterbi algorithm. *BMC Bioinformatics*, 12(Suppl 3):S1, 2011.

5. Neil Barrett and Jens Weber-Jahnke. A token centric part-of-speech tagger for biomedical text. In *Proceedings of the 13th conference on Artificial intelligence in medicine*, AIME'11, pages 317–326, Berlin, Heidelberg, 2011. Springer-Verlag.

6. Neil Barrett, Jens Weber-Jahnke, and Vincent Thai. N. Barrett, J. Weber-Jahnke, and V. Thai. Automated clinical coding using semantic atoms and topology. In *Computer-based Medical Systems (CBMS), 2012 25th International Symposium on*, pages 1–6, June 2012.

7. Neil Barrett and Jens Weber-Jahnke. The effect of part-of-speech tagging errors on dependency parsing. In *ICML 2012 Workshop on Clinical Data Analysis*, 2012.

Appendix B

Secondary Segmentor Instructions

You are asked to segment a sentence into its tokens (pieces). Here's an example (sentence followed by tokens, one per line):

A car, faster than lighting, was painted red.

A
car
,
faster
than
lighting
,
was
painted
red
.

When segmenting a sentence you are permitted to 1) separate and 2) delete pieces of the sentence. In the example above, spaces were deleted and punctuation was separated from its adjoining word.

Tokens may have spaces (whitespace)[1]. Some people *may* choose to do the following:

[1]This example was meant to demonstrate that tokenization may not follow an obvious pattern, such as tokens separated by whitespace

New York is a big city.

New York
is
a
big
city
.

Below are segmenting rules that you must follow, These rules apply to very few situations. For most cases, you will decide how to segment a sentence.

- Consider the following as separate tokens (upper or lower case): 'll 're 've n't 's '

- Abbreviations of closed-class words must be expanded. Example: The sentence "Jon/Roger are running." would become "Jon and Roger are running." Table B.1 lists closed-class words.

Apply what you've just learned to these examples:

Entire upper dental arch (body structure)

Entire
upper
dental
arch
(
body
structure
)

Royal Navy - non-commissioned personnel (occupation)

Royal
Navy
-
non-commissioned
personnel
(

occupation
)

Posterior cervical spinal cord injury, without spinal injury, C1-4
Posterior
cervical
spinal
cord
injury
,
without
spinal
injury
,
c1
to
4

Primidone 50mg tablet
Primidone
50
mg
tablet

Precorrin-3B C17-methyltransferase
Precorrin-3B
C17-methyltransferase

Table B.1: A list of closed-class words

a	either	nobody	though
about	enough	none	through
above	ever	nor	throughout
across	every	of	till
after	everybody	off	to
against	everyone	on	toward
all	everything	once	under
along	except	one	underneath
although	few	onto	until
among	for	or	up
an	from	ours	upon
and	he	ourselves	us
another	her	out	we
any	hers	outside	what
anybody	herself	over	whatever
anyone	him	past	when
anything	himself	per	where
around	his	several	whether
as	how	she	which
at	I	since	whichever
because	if	so	while
before	in	some	who
behind	inside	somebody	whoever
below	into	someone	whom
beneath	it	sufficient	whomever
beside	its	than	with
between	itself	that	within
beyond	like	the	without
both	many	theirs	yet
but	me	them	you
by	mine	themselves	yours
despite	myself	these	yourself
down	neither	they	yourselves
during	near	this	
each	no	those	

Appendix C

Manual Coding Details

This appendix details encoding practices of Section 6.3. It contains details on the simplified encoding of 5 pre-coordinated cases and the simplified encoding of post-coordinated expressions, as well as the encoding of non-coded texts. Furthermore, the one-to-many mapping of a SNOMED CT concept to MetaMap identifiers is described below.

Table C.1 contains simplified codings that account for missing context (context present during the original research). Table C.2 contains the simplified post-coordinated expression codes. Tables C.3, C.4 and C.5 contain the codes for previously non-coded texts.

Table C.6 contains the SNOMED CT codes that mapped to multiple MetaMap identifiers and the selected MetaMap identifier. The table omits 7 duplicate entries.

Table C.1: Simplified pre-coordinated codings

Text	Previous	Simplified
ca lung wi mets	315006004, Metastasis from malignant tumor of lung	363358000, Malignant tumor of lung
ca	254837009, Malignant tumor of breast	86049000, Malignant neoplasm, primary
stage iv gastric ca with liver mets	94381002, Secondary malignant neoplasm of liver	363349007, Malignant tumor of stomach
on neck	188046002, Malignant melanoma of neck	302550007, Entire neck
lung&bone	363358000, Malignant tumor of lung	39607008, Lung structure

Table C.2: Simplified post-coordinated codings

Text	Expression	Simplified
mesothelioma rt lung wi mets	64572001: 116676008=62064005, 363698007=3341006	62064005, Mesothelioma, malignant
adenocarcinoma with liver mets	64572001: 116676008=35917007	35917007, Adenocarcinoma, no subtype
likely pancreatic with bone mets	41769001: 246090004=363418001	363418001, Malignant tumor of pancreas
carcinoid tumor with extensive intraab mets	64572001: 116676008=189607006	189607006, Carcinoid tumor - morphology
cholangioca with liver mets	64572001: 116676008=70179006	70179006, Cholangiocarcinoma
aneurysms & glaucoma	64572001: 116676008=85659009	432119003, Aneurysm
rt lung ca with mets	363358000: 363698007=3341006	93880001, Primary malignant neoplasm of lung

Table C.3: Non-coded text encodings (1/3)

Text	Codes
problem adjusting to medications	405177001
future options	128927009
ca lung stage 4 nsc wi mets	423121009
grade 4 glioblastoma	393563007
suggestions for t reatment	71388002
long standing anemic thrombocytopenia purpura	402653004
colon/rectal primary wi mets	363351006
menigioma (head)	19453003
cancer of pleural lining	62064005
posterior pharynx cancer	187709007
m myeloma	109989006
pancreatic adenocarcinoma with mets	94459006
metastatic angiosarcoma	403977003
chronic renal failure secondary to sclerosing	235917005
placement for future	71861002
invasive ductal ca	82711006
ca breast wi nets	254837009
metastatic adenocarcinoma wi liver & kidney mets	4590003
intramedullary nail right femur	89344001
pancreatic adeno ca	363418001
low grade non-hodgkinsllymphoma	118601006
brain malignancy	428061005
cardiac disease history	275544003
t4 n2 scc of larynx wi mets	363429002
large oropharyngeal squamous cell ca	423464009
lymphoma right leg	93196000
high grade glioma-brain stem with mets to chest & abdomen	444545003
family exhausted	405205002
esophagus cancer pain control	225782006
metastatic breast ca with mets	94297009
melanoma left calf	93639006
small cell lung with intrathoracic disease	254632001

Table C.4: Non-coded text encodings (2/3)

Text	Codes
metastatic fibrous sarcoma	53654007
patient cannot go home	103312009
and hospice placement	305336008
sarcoma connective tissue/ sarcoma left pelvis	449293008
cortical atrophy	43100002
pancreatic carcinoma	372142002
ltc/sm	53597013
multiple medical problems	64572001
paracentesis for ascites	89305009
pacemaker/nervous system	14106009
bronchial esophageal fistula	235641005
sub acute	19939008
no support a home	276038000
lymphona/lunch	21964009
redwater health care centre	264361005
left breast ca wi mets	372137005
malignant melanoma with brain mets	372244006
benign mucinous cystoadenoma causing extensive intra abdominal pathology	71307009
metastatic cell ca	128462008
small cell with liver mets	74364000
non-small cell ca wi lung mass	254637007
left buccal cancer	363386008
meningitis carcinomatose	230156002
irreversible cognitive failure	2470005
inflammation lt breast	266579006
ca aortic	93677000
locally advanced & metastatic pancreatic ca	363418001
unknown gastrointestinal with lung & liver mets	94313005
olivopontine cerebellar atrophy	67761004
breast or cervical origin metastatic ca wi lung mets	128462008
bonal and nodal mets in dediasternum	94605004
melanoma (eye) with mets	274087000

Table C.5: Non-coded text encodings (3/3)

Text	Codes
squamous cell ca - mandible	422572002
ca lung rul wi bone mets	363358000
recurrent bronchogenic ca stage 3b	254622008
intermediate grade lymphoma of the lung	21964009
non-cancer cardiopulmonary	87837008
pain control cancer of lung	225782006
glioma intramedullary spine	363438000
fibrosarcoma of leg (rt) lung mets	443250000
non-cancer other	-
mets ca	128462008
glioblastoma with mets	276826005
renal cell with local & lung mets	93849006
diffuse large cell ca	22687000
small cell anaplastic ca	58248003
non-cancer	-
squamous cell ca h&n piriform tumor	402815007
left lateral foot	182307001
upper airway destruction	301252002

Table C.6: Selected MetaMap identifiers

SNOMED CT code	Selected MetaMap identifier
2776000	C0011206
13645005	C0024117
19453003	C1384407
53741008	C1956346
62064005	C0345967
69896004	C0003873
70179006	C0206698
75478009	C0032343
86049000	C1306459
93677000	C0153657
93781006	C1879358
93849006	C1306058
225782006	C1304888
230156002	C0220654
254622008	C0007121
254915003	C0279702
269466003	C0153509
363351006	C0949022
363354003	C0007847
396275006	C0029408
443250000	C0016057

Appendix D

Example Palliative Care Consult Letter

CONSULTANT: Dr. xxx

DATE OF CONSULTATION: 22 December 2007

Thank you for referring this pleasant unfortunate 54-year-old patient to Community Consult Team for regional palliative care program.

Diagnosis metastatic ovarian cancer of granulosis cell type. Reason for referral placement and plans.

HISTORY OF PRESENT ILLNESS:
Patient was diagnosed with ovarian cancer in 1992 which was treated with total hysterectomy and oophorectomy. No chemotherapy was received but she had radiation treatment after the surgery four days a week for three months. She was symptom free for nine years but there was recurrence of the disease in 2000. This patient is well known to our team and has been seen on multiple occasions by Community Consult Team.

The history of her present concerns starts in March 6 2007 when she had a CT scan of the abdomen followed by another CT scan in May 2007 following admission to hospital for recurrent UTI. In May 2007, CT of the abdomen and pelvis showed multiple metastatic lesions in mesenteric area which have decreased in sizes compared to the CT scan in March 6 2007. Mild splenomegaly was present and two hepatic

masses were noted. A mass of 2.7 cm was noted in the left anterior abdominal wall. There was also a hypodense mass in the lower pole of the right kidney. On that admission, the patient was receiving antibiotics for urinary tract infection. Apparently, the patient started to do self catheterization since October 2006 and had an episode of rupture of the bladder due to urinary retention.

In June 2007, the patient was again admitted for urosepsis and a repeat CT scan at that time showed pelvicaliceal as well as ureteric dilatation bilaterally to the level of the UVJ. There was a tiny calcification over the distal left ureter. Omental metastasis was noted. Pancreas and adrenals were unremarkable and there was mild enlargement of the spleen. Small and nonobstructing left kidney stone was present. Retrograde urography in June 2007 for assessing both renal collecting systems showed bilateral hydroureter and hydronephrosis. This was followed by cystoscopy and that showed an inflamed bladder wall but differential diagnosis of neoplastic process, chronic inflammation, interstitial cystitis or radiation fibrosis was discussed. Biopsy showed only urothelial inflammation with no evidence of malignancy. A nuclear medicine exam in July 2007 revealed a smaller right kidney and a defect at the lower pole of the right kidney, probably due to cortical scarring. A mechanical obstruction at the UVJ was noted. An abdominal ultrasound in July 2007 showed mesenteric lymph nodes and bilateral hydronephrosis and calculi on the left side. Subsequently, the patient received bilateral ureteric stents.

The patient has been seen by our team in October 2007 and the main issue at that time was fatigue and assessment for disposition plans. She was seen as appropriate for a long term care facility placement.

PRESENT CONCERNS:
On today's interview, the patient denied any discomfort.

SYSTEMS REVIEW:
On further questioning, she stated that she has had pain in her bladder area since last night and she rated it at 3-4/10 and is due to her catheter not functioning very well. She states that every time this happens, irrigation of the catheter would help. The pain does not aggravate with moment and is a dull pain with no radiation. She has not used any break through analgesia. She does not have any opiate toxicity symptoms. She rates her tiredness at 10/10, nausea 0/10. She says that once in a week, she gets nausea for a couple of hours and uses a few breakthroughs of Maxeran

and she will be ok until the next week. Her bowel movements are regular and daily and her last bowel movement was on 20 December 2007. She denied depression and anxiety and rates her drowsiness as 5-6/10. She states that she sleeps well at night time. She rates her appetite as 0/10 and she eats 100% of her food and drinks water well. She rates her feeling of well being at 5/10. She denied shortness of breath. She states that she has had some severe cold in the past few days and she is getting over it and she has a slight moist cough but no phlegm. She states that her catheter needs to be changed every month. She has no problem with swallowing. She walks with her four-wheeled walker and goes outside to smoke and comes back independently. She is independent with most of her activities of daily living. She denied any weakness or numbness in her extremities.

PAST MEDICAL HISTORY:
Diabetes type II, metastatic ovarian ca, DVT, pulmonary emboli, Cushing syndrome, hiatal hernia, GERD, depression, asthma, obstructive sleep apnea, rheumatoid arthritis/osteoarthritis, adrenal insufficiency due to adrenalectomy in 1999 for adrenal mass, bladder rupture in October 2006, obesity, UTI/urosepsis, nonalcoholic steatohepatitis, tonsillectomy, left leg tumor removed in 1968, cholecystectomy 1976, hysterectomy for ovarian ca 1992, adrenalectomy 1999 and polyp removed from her large bowel.

ALLERGIES:
Sulfa trimethoprim, ASA, Keflex, Flagyl, Ceclor, gabapentin, Actos and Lipitor.

PSYCHOSOCIAL HISTORY:
She has been separated for 22 years and she has three children who are supportive. She used to live alone in Edmonton and is on AISH but she has given up her apartment and has no place of residence any more. Her son helped her with her groceries and her daughter also was helping and they are very supportive. She is a smoker of half a pack per day for 28 years and there is no history of alcohol consumption. She worked in factories and retail stores. There is no personal directive in place and her code status is DNR. Her cage is 0/4.

CURRENT MEDICATIONS:
Please see medication profile on the chart.

PHYSICAL EXAMINATION:

On today's examination, she is active, alert and oriented. Her mini-mental status examination is 30/30 and her expected is 27/30. Her PPS is 70%. There are no lymph nodes in the neck, no thrush was noted. There is a slight amount of secretion in her airways but there is no wheezing or crackles. Air entry is normal. Normal S1 and S2 were noted. No S3 or S4 were audible. Grade II systolic ejection murmur at LLSV was audible. Bowel sounds are present in all four quadrants. Abdomen is obese, tympanic and very tender to palpation in any part of the abdomen, especially right upper quadrant and lower abdomen. Hepatomegaly was noted. Multiple surgical scars are seen in the abdominal wall. Lower extremities showed discoloration of the right lower extremity associated with engorged superficial varicose veins. There is no swelling and Holman sign was negative. Neurological examination of the cranial nerves and upper and lower extremities was normal. Deep tendon reflexes were +1 in upper and lower extremities and plantar reflexes are downgoing. Upon palpation of the spine, there is some tenderness in the lower C-spine, T-L spine and in sacral area was noted. She has a Foley catheter in situ which, at times, drains a slight amount of blood as well as the urine.

IMPRESSION:

Symptoms at the moment: The malfunctioning of the catheter as per patient's observation is the main issue; hence I suggest that the Foley catheter to be irrigated. I also suggest that Dilaudid 1 mg po q2h prn for break through analgesia to be added to her current regimen of pain killers. Also in regard to her current cold and secretions in the airway, I suggest that Combivent 1 puff b.i.d. prn may be used. I also suggest Gravol to be discontinued.

DISPOSITION:

Considering the patient's function and her comments that she does not feel that she is ready for hospice admission, I have to agree with her observation and at the moment, she seems an appropriate candidate for long term care facility such as Assisted Living placement.

I again thank you for referring this pleasant patient to Community Consult Team for Regional Palliative Care Program and we will be more than happy to come back in the future and assess for hospice in a case of deterioration of function and progression of the disease.

Appendix E

Clinical Assessment Tools in Predicting Survival of Terminally Ill Patients in Different Palliative Care Settings

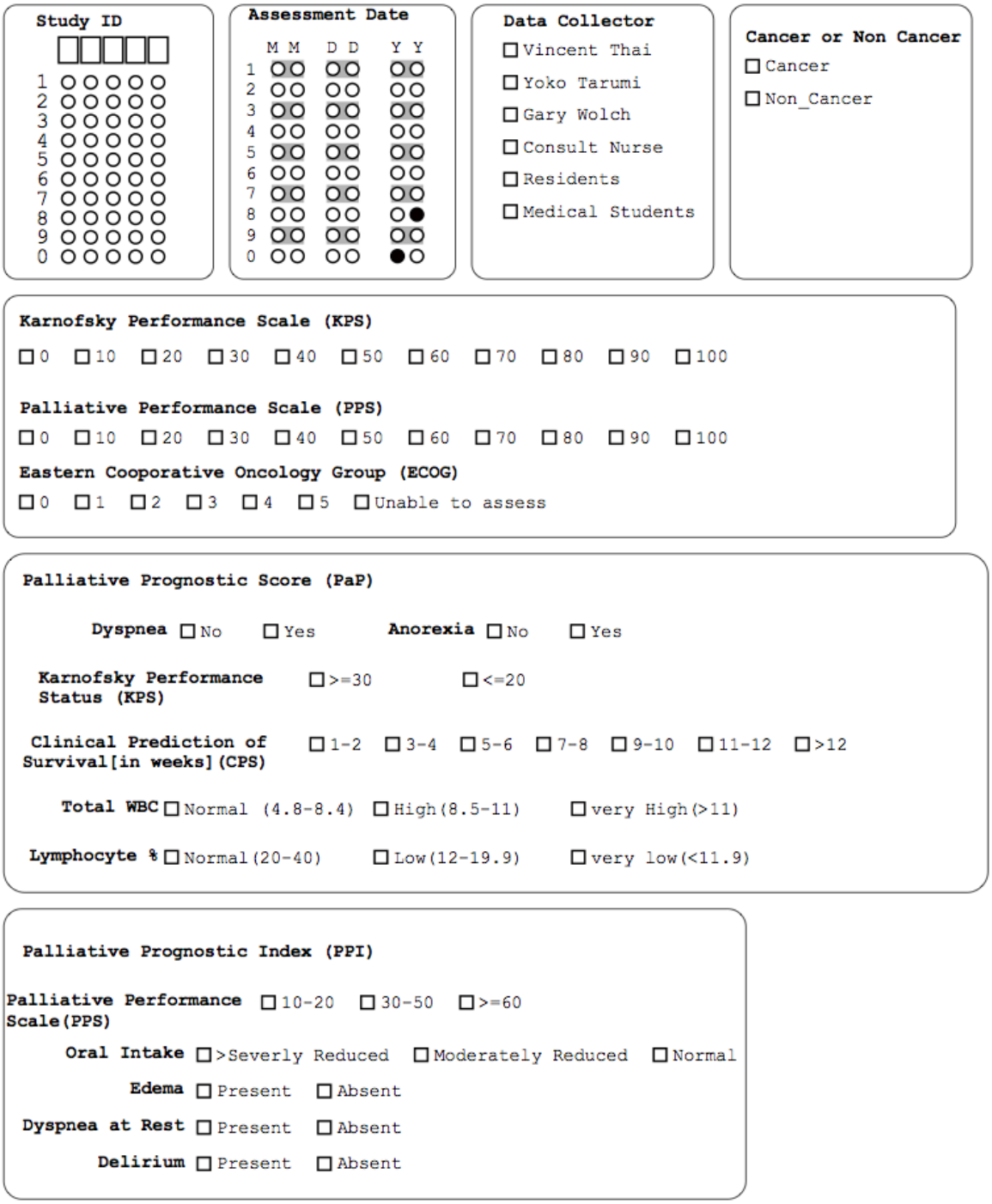

Figure E.1: Clinical assessment tools page 1

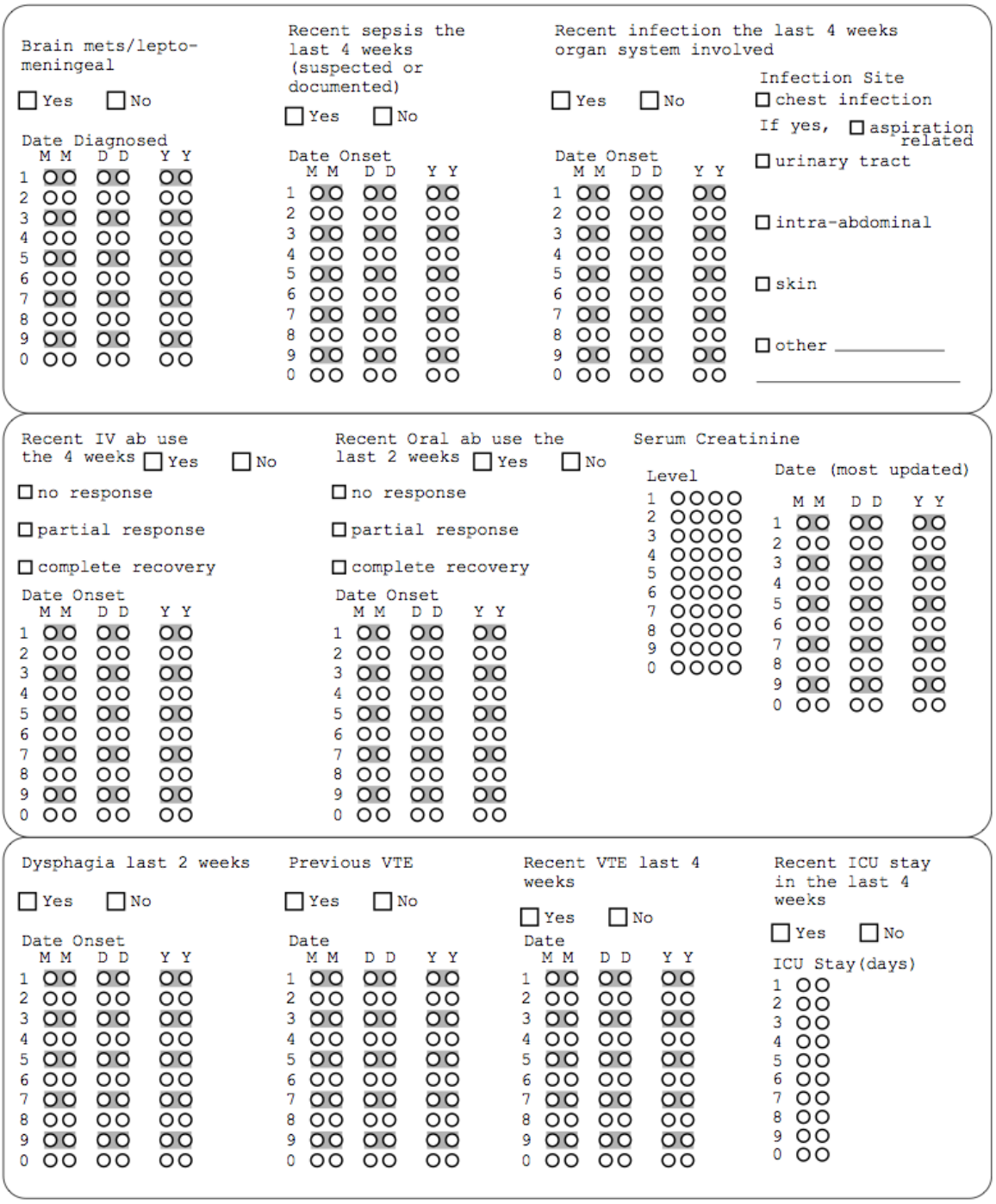

Figure E.2: Clinical assessment tools page 2

Bibliography

[1] Adrian Akmajian, Richard A. Demers, Ann K. Farmer, and Robert M. Harnish. *Linguistics: An Introduction to Language and Communication*. Massachusetts Institute of Technology, Cambridge, MA, USA, 5th edition, 2001.

[2] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, Cambridge, MA, USA, 2 edition, 2010.

[3] Robert Arens. A preliminary look into the use of named entity information for bioscience text tokenization. In *Proceedings of the Student Research Workshop at HLT-NAACL 2004 on XX*, HLT-NAACL '04, pages 37–42, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[4] Alan R. Aronson. Metamap: Mapping text to the umls metathesaurus. http://skr.nlm.nih.gov/papers, 2006.

[5] Alan R Aronson and François-Michel Lang. An overview of metamap: historical perspective and recent advances. *J Am Med Inform Assoc*, 17(3):229–36, 2010.

[6] Mohammed Attia. Arabic tokenization system. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 65–72, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[7] Vijayaraghavan Bashyam and Ricky K Taira. Incorporating syntactic dependency information towards improved coding of lengthy medical concepts in clinical reports. In *Proceedings of the Workshopon BioNLP*, pages 125–132. Association for Computational Linguistics, 2009.

[8] R H Baud, C Lovis, A M Rassinoux, and J R Scherrer. Morpho-semantic parsing of medical expressions. *Proc AMIA Symp*, pages 760–764, 1998.

[9] R H Baud, C Lovis, P Ruch, and A M Rassinoux. A toolset for medical text processing. *Stud Health Technol Inform*, 77:456–461, 2000.

[10] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 120–128. Association for Computational Linguistics, 2006.

[11] Thorsten Brants. Cascaded markov models. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 118–125, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[12] Thorsten Brants. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[13] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21:543–565, 1995.

[14] Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164. Association for Computational Linguistics, 2006.

[15] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern Oriented Software Architecture: On Patterns and Pattern Languages*. John Wiley & Sons, 2007.

[16] D A Campbell and S B Johnson. Comparing syntactic complexity in medical and non-medical corpora. *Proc AMIA Symp*, pages 90–94, 2001.

[17] W W Chapman, W Bridewell, P Hanbury, G F Cooper, and B G Buchanan. Evaluation of negation phrases in narrative clinical reports. *Proc AMIA Symp*, pages 105–109, 2001.

[18] W W Chapman, W Bridewell, P Hanbury, G F Cooper, and B G Buchanan. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform*, 34(5):301–310, 2001.

[19] Lee M. Christensen, Henk Harkema, Peter J. Haug, Jeannie Y. Irwin, and Wendy W. Chapman. Onyx: a system for the semantic analysis of clinical text. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, BioNLP '09, pages 19–27, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[20] Lee M. Christensen, Peter J. Haug, and Marcelo Fiszman. Mplus: a probabilistic medical language understanding system. In *Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain*, pages 29–36, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[21] Cheryl Clark, John Aberdeen, Matt Coarr, David Tresner-Kirsch, Ben Wellner, Alexander Yeh, and Lynette Hirschman. Determining assertion status for medical problems in clinical records. In *2010 i2b2/VA Workshop on Challenges in Natural Language Processing for Clinical Data*, 2010.

[22] Andrew B. Clegg and Adrian J. Shepherd. Evaluating and integrating treebank parsers on a biomedical corpus. In *Proceedings of the Workshop on Software*, Software '05, pages 14–33. Association for Computational Linguistics, 2005.

[23] Anni R Coden, Serguei V Pakhomov, Rie K Ando, Patrick H Duffy, and Christopher G Chute. Domain-specific language models and lexicons for tagging. *J Biomed Inform*, 38(6):422–30, Dec 2005.

[24] Aaron M. Cohen. Five-way smoking status classification using text hot-spot identification and error-correcting output codes. *Journal of the American Medical Informatics Association*, 15(1):32–35, 2008.

[25] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[26] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2 edition, 2002.

[27] Marie-Catherine de Marneffe and Christopher D. Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics, 2008.

[28] Guy Divita, Allen C Browne, and Russell Loane. dtagger: a pos tagger. *AMIA Annu Symp Proc*, pages 200–3, 2006.

[29] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29:103–130, November 1997.

[30] AHIMA e-HIMTM Work Group on Computer-Assisted Coding. Delving into computer-assisted coding. *Journal of AHIMA 75*, 75(10):48A–H, Nov-Dec 2004.

[31] J S Elkins, C Friedman, B Boden-Albala, R L Sacco, and G Hripcsak. Coding neuroradiology reports for the northern manhattan stroke study: a comparison of natural language processing and manual review. *Comput Biomed Res*, 33(1):1–10, 2000.

[32] D. Franklin and I. Muchnik. An introduction to support vector machines. In James Abello and Graham Cormode, editors, *Discrete Methods in Epidemiology*, volume 70, pages 13–20. Rutgers State University of New Jersey, 2006.

[33] C Friedman. Towards a comprehensive medical language processing system: methods and issues. *Proc AMIA Annu Fall Symp*, pages 595–599, 1997.

[34] C Friedman. A broad-coverage natural language processing system. *Proc AMIA Symp*, pages 270–274, 2000.

[35] C Friedman, G Hripcsak, and I Shablinsky. An evaluation of natural language processing methodologies. *Proc AMIA Symp*, pages 855–859, 1998.

[36] C Friedman, C Knirsch, L Shagina, and G Hripcsak. Automating a severity score guideline for community-acquired pneumonia employing medical language processing of discharge summaries. *Proc AMIA Symp*, pages 256–260, 1999.

[37] Carol Friedman and George Hripcsak. Natural language processing and its future in medicine. *Academic medicine: journal of the AAMC*, 74(8):890–895, 1999.

[38] Carol Friedman and Stephen B. Johnson. Natural language and text processing in biomedicine. In Edward H. Shortliffe and James J. Cimino, editors, *Biomedical Informatics - Computer Applications in Health Care and Biomedicine*, Health Informatics, chapter 8, pages 312–343. Springer New York, 3rd edition, 2006.

[39] Carol Friedman, Pauline Kra, and Andrey Rzhetsky. Two biomedical sublanguages: a description based on the theories of zellig harris. *J Biomed Inform*, 35(4):222–235, 2002.

[40] Carol Friedman, Lyudmila Shagina, Yves Lussier, and George Hripcsak. Automated encoding of clinical documents based on natural language processing. *J Am Med Inform Assoc*, 11(5):392–402, 2004.

[41] J. Gerrein, A. Kherlopian, M. Berman, V. Wu, and A. Wald. Medfolink: bridging the gap between it and the medical community. *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 656–657, Aug. 2003.

[42] Jorge Grana, Miguel A. Alonso, and Manuel Vilares. A common solution for tokenization and part-of-speech tagging: One-pass viterbi algorithm vs. iterative approaches. In *Text, Speech and Dialogue, volume 2448 of Lecture Notes in Computer Science*, pages 3–10. Springer-Verlag, 2002.

[43] Jorge Grana, Francisco-Mario Barcala, and Jesus Vilares Ferro. Formal methods of tokenization for part-of-speech tagging. In *CICLing '02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 240–249, London, UK, 2002. Springer-Verlag.

[44] Dawn G. Gregg, Uday R. Kulkarni, and Ajay S. Vinzé. Understanding the philosophical underpinnings of software engineering research in information systems. *Information Systems Frontiers*, 3(2):169–183, 2001.

[45] Jin Guo. Critical tokenization and its properties. *Comput. Linguist.*, 23(4):569–596, 1997.

[46] U Hahn, M Romacker, and S Schulz. Why discourse structures in medical reports matter for the validity of automatically generated text knowledge bases. *Stud Health Technol Inform*, 52 Pt 1:633–638, 1998.

[47] U Hahn, M Romacker, and S Schulz. Discourse structures in medical reports–watch out! the generation of referentially coherent and valid text knowledge bases in the medsyndikate system. *Int J Med Inform*, 53(1):1–28, 1999.

[48] Udo Hahn and Joachim Wermter. High-performance tagging on medical texts. In *COLING '04: Proceedings of the 20th international conference on Compu-*

tational Linguistics, pages 973–980. Association for Computational Linguistics, 2004.

[49] M. Hassler and G. Fliedl. Text preparation through extended tokenization. In Zanasi, A and Brebbia, CA and Ebecken, NFF, editor, *Data Mining VII: Data, Text and Web Mining and Their Business Applications*, volume 37, pages 13–21. WIT Press/Computational Mechanics Publications, 2006.

[50] Ying He and Mehmet Kayaalp. A comparison of 13 tokenizers on medline. Technical Report LHNCBC-TR-2006-003, The Lister Hill National Center for Biomedical Communications, 2006.

[51] Jaqui Hewitt-Taylor. *Clinical Guidelines & Care Protocols*. Whurr Publishers Limited, West Sussex, England, 2006.

[52] G Hripcsak, G J Kuperman, and C Friedman. Extracting findings from narrative reports: software transferability and sources of physician disagreement. *Methods Inf Med*, 37(1):1–7, 1998.

[53] George Hripcsak, Li Zhou, Simon Parsons, Amar K Das, and Stephen B Johnson. Modeling electronic discharge summaries as a simple temporal constraint satisfaction problem. *J Am Med Inform Assoc*, 12(1):55–63, 2005 Jan-Feb.

[54] Yang Huang. *ChartIndex - A Contextual Approach to Automated Standards-Based Encoding of Clinicla Documents*. PhD thesis, Stanford University, 2006.

[55] O. Huseth and T. Brox Rost. Developing an annotated corpus of patient histories from the primary care health record. *Bioinformatics and Biomedicine Workshops, 2007. BIBMW 2007. IEEE International Conference on*, pages 165–173, November 2007.

[56] IHTSDO. *SNOMED Clinical Terms - User Guide*. The International Health Terminology Standards Development Organisation, July 2011.

[57] Jing Jiang and Chengxiang Zhai. An empirical study of tokenization strategies for biomedical information retrieval. *Inf. Retr.*, 10(4-5):341–363, 2007.

[58] Stephen B Johnson. Natural language processing in biomedicine. In Bronzino, editor, *The Handbook of Biomedical Engineering*, chapter Natural Language Processing in Biomedecine, pages 188.1–188.6. CRC Press, 2000.

[59] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, Upper Saddle River, New Jersey, 2009.

[60] J-D Kim, T Ohta, Y Tateisi, and J Tsujii. Genia corpus–semantically annotated corpus for bio-textmining. *Bioinformatics*, 19 Suppl 1:i180–2, 2003.

[61] T. Kiss and J. Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

[62] András Kuba, András Hócza, and János Csirik. Pos tagging of hungarian with combined statistical and rule-based methods. In Petr Sojka, Ivan Kopecek, and Karel Pala, editors, *Text, Speech and Dialogue*, volume 3206 of *Lecture Notes in Computer Science*, pages 113–120. Springer Berlin / Heidelberg, 2004.

[63] Sandra Kübler, Ryan McDonald, and Joakim Nivre. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, 2009.

[64] S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, L. Ungar, S. Winters, and P. White. Integrated annotation for biomedical information extraction. In *Proc. of HLT/NAACL*, pages 61–68, 2004.

[65] Kathryn Lanuke, Robin L Fainsinger, Donna DeMoissac, and James Archibald. Two remarkable dyspneic men: when should terminal sedation be administered? *J Palliat Med*, 6(2):277–281, Apr 2003.

[66] Dennis H Lee, Francis Y Lau, and Hue Quan. A method for encoding clinical datasets with snomed ct. *BMC Med Inform Decis Mak*, 10:53, 2010.

[67] Kaihong Liu, Wendy Chapman, Rebecca Hwa, and Rebecca S Crowley. Heuristic sample selection to minimize reference standard training set for a part-of-speech tagger. *J Am Med Inform Assoc*, 14(5):641–50, 2007.

[68] Jacqueline L. Longe, editor. *The gale encyclopedia of medicine*, volume 1-5. Thomson and Gale, Farmington Hills, Michigan, 2006.

[69] Edward Loper and Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, pages 63–70, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[70] C Lovis and R H Baud. Fast exact string pattern-matching algorithms adapted to the characteristics of the medical language. *J Am Med Inform Assoc*, 7(4):378–391, 2000 Jul-Aug.

[71] Henry J Lowe, Yang Huang, and Donald P Regula. Using a statistical natural language parser augmented with the umls specialist lexicon to assign snomed ct codes to anatomic sites and pathologic diagnoses in full text pathology reports. *AMIA Annu Symp Proc*, 2009:386–90, 2009.

[72] Yanjun Ma and Andy Way. Bilingually motivated domain-adapted word segmentation for statistical machine translation. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 549–557, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[73] Brian Mac Namee, John Kelleher, , and Sarah Jane Delany. Medical language processing for patient diagnosis using text classification and negation labelling. In *i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data*, 2008.

[74] Christopher D. Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part I*, CICLing'11, pages 171–189. Springer-Verlag, 2011.

[75] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19(2):313–330, 1993.

[76] Ryan McDonald, Koby Crammer, and Fernando Pereira. Flexible text segmentation with structured multilabel classification. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 987–994, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[77] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 91–98. Association for Computational Linguistics, 2005.

[78] Ryan McDonald and Joakim Nivre. Characterizing the errors of data-driven dependency parsing models. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 122–131, 2007.

[79] Ryan McDonald and Joakim Nivre. Analyzing and integrating dependency parsers. *Comput. Linguist.*, 37(1):197–230, March 2011.

[80] Ryan Mcdonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *In Proc. of EACL*, pages 81–88, 2006.

[81] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005.

[82] Genevieve B Melton and George Hripcsak. Automated detection of adverse events using natural language processing of discharge summaries. *J Am Med Inform Assoc*, 12(4):448–457, 2005.

[83] William Mendenhall and Terry Sincich. *Statistics for the Engineering and Computer Sciences*. Dellen Publishing Company, San Francisco, 1984.

[84] Eneida A Mendonca, Janet Haas, Lyudmila Shagina, Elaine Larson, and Carol Friedman. Extracting information on pneumonia in infants using natural language processing of radiology reports. *J Biomed Inform*, 38(4):314–321, 2005.

[85] Gary H. Merrill, Patrick B. Ryan, and Jeffery L. Painter. Construction and annotation of a umls/snomed-based drug ontology for observational pharmacovigilance. In *IDAMAP*, 2008.

[86] Stephane Meystre and Peter J Haug. Natural language processing to extract medical problems from electronic clinical documents: performance evaluation. *J Biomed Inform*, 39(6):589–599, 2006 Dec.

[87] Stéphane M Meystre, Julien Thibault, Shuying Shen, John F Hurdle, and Brett R South. Textractor: a hybrid system for medications and reason for their prescription extraction from clinical text documents. *journal of the American Medical Informatics Association*, 17(5):559–562, 2010.

[88] John E. Miller, Manabu Torii, and K. Vijay-Shanker. Adaptation of pos tagging for multiple biomedical domains. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, BioNLP '07, pages 179–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[89] Y. Miyao, R. Saetre, K. Sagae, T. Matsuzaki, and J. Tsujii. Task-oriented evaluation of syntactic parsers and their representations. *Proceedings of ACL-08: HLT*, pages 46–54, 2008.

[90] Yusuke Miyao, Kenji Sagae, Rune Saetre, Takuya Matsuzaki, and Jun'ichi Tsujii. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25(3):394–400, Feb 2009.

[91] Michela Montesi and Patricia Lago. Software engineering article types: An analysis of the literature. *J. Syst. Softw.*, 81(10):1694–1714, October 2008.

[92] M. Ashley Morrison. *Encyclopedia of Research Design*, volume 2, pages 779 – 782. SAGE Publications, Inc., 2010.

[93] P G Mutalik, A Deshpande, and P M Nadkarni. Use of general-purpose negation detection to augment concept indexing of medical documents: a quantitative study using the umls. *J Am Med Inform Assoc*, 8(6):598–609, 2001.

[94] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. Natural language processing: an introduction. *J Am Med Inform Assoc*, 18(5):544–51, 2011.

[95] Fiammetta Namer and Robert Baud. Predicting lexical relations between biomedical terms: towards a multilingual morphosemantics-based system. *Connecting Medical Informatics and Bio-Informatics*, pages 793–798, 2005.

[96] Anthony Nguyen, Julie Moore, Michael Lawley, David Hansen, and Shoni Colquist. Automatic extraction of cancer characteristics from free-text pathology reports for cancer notifications. *Stud Health Technol Inform*, 168:117–24, 2011.

[97] Anthony N Nguyen, Michael J Lawley, David P Hansen, and Shoni. A Colquist. Simple pipeline application for identifying and negating snomed clinical terminology in free text. In Vitali Sintchenko and Peter Croll, editors, *HIC 2009:*

Frontiers of Health Informatics - Redefining Healthcare, pages 188–196, August 2009.

[98] Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 221–225. Association for Computational Linguistics, 2006.

[99] Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez-Rodríguez. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 833–841. Association for Computational Linguistics, 2010.

[100] NLM. Specialist text tools, 2011.

[101] William O'Grady and John Archibald. *Contemporary Linguistic Analysis : an introduction*. Pearson Education Canada, Toronto, Ontario, 5th edition, 2004.

[102] Michel Oleynik, Percy Nohama, Pindaro Secco Cancian, and Stefan Schulz. Performance analysis of a pos tagger applied to discharge summaries in portuguese. *Stud Health Technol Inform*, 160(Pt 2):959–63, 2010.

[103] Serguei Pakhomov, James Buntrock, and Patrick Duffy. High throughput modularized nlp system for clinical text. In *Association for Computational Linguistics Interactive Poster and Demonstration Sessions*, pages 25–28, 2005.

[104] Serguei Pakhomov, Anni Coden, and Christopher Chute. Creating a test corpus of clinical notes manually tagged for part-of-speech information. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, JNLPBA '04, pages 62–65, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[105] Serguei V Pakhomov, Anni Coden, and Christopher G Chute. Developing a corpus of clinical notes manually annotated for part-of-speech. *International journal of medical informatics*, 75(6):418–429, June 2006.

[106] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12):1053–1058, 1972.

[107] T. Pedersen. Determining smoker status using supervised and unsupervised learning with lexical features. In *i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data*, 2006.

[108] Sampo Pyysalo, Tapio Salakoski, Sophie Aubin, and Adeline Nazarenko. Lexical adaptation of link grammar to the biomedical sublanguage: a comparative evaluation of three approaches. *BMC Bioinformatics*, 7 Suppl 3:S2, 2006.

[109] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, pages 133–142, 1996.

[110] P Ruch, R Baud, and A Geissbuhler. Evaluating and reducing the effect of data corruption when applying bag of words approaches to medical records. *Int J Med Inform*, 67(1-3):75–83, 2002.

[111] P Ruch, P Bouillon, G Robert, R Baud, and A M Rassinoux. Tagging medical texts: a rule-based experiment. *Stud Health Technol Inform*, 77:448–455, 2000.

[112] P Ruch, J Wagner, P Bouillon, R H Baud, A M Rassinoux, and J R Scherrer. Medtag: tag-like semantics for medical document indexing. *Proc AMIA Symp*, pages 137–141, 1999.

[113] Neil J. Salkind, editor. *Encyclopedia of Measurement and Statistics*, volume 3, pages 1051–1053. Sage Reference, Thousand Oaks, CA, 2007.

[114] Mary Shaw. What makes good research in software engineering? *International Journal of Software Tools for Technology Transfer*, 4(1):1–7, 2002.

[115] L Smith, T Rindflesch, and W J Wilbur. Medpost: a part-of-speech tagger for biomedical text. *Bioinformatics*, 20(14):2320–1, Sep 2004.

[116] Illés Solt, Domonkos Tikk, Viktor Gál, and Zsolt T Kardkovács. Semantic classification of diseases in discharge summaries using a context-aware rule-based classifier. *Journal of the American Medical Informatics Association*, 16(4):580–584, 2009.

[117] Mary H Stanfill, Margaret Williams, Susan H Fenton, Robert A Jenders, and William R Hersh. A systematic literature review of automated clinical coding and classification systems. *J Am Med Inform Assoc*, 17(6):646–51, 2010.

[118] Holger Stenzhorn, Edson José Pacheco, Percy Nohama, and Stefan Schulz. Automatic mapping of clinical documentation to snomed ct. *Stud Health Technol Inform*, 150:228–32, 2009.

[119] Amarnag Subramanya, Slav Petrov, and Fernando Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 167–176, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[120] Yuka Tateisi, Yoshimasa Tsuruoka, and Jun-ichi Tsujii. Subdomain adaptation of a pos tagger with a small corpus. In *Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology*, LNLBioNLP '06, pages 136–137, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[121] Vince Thai, Francis Lau, Gary Wolch, Ju Yang, and Hue Quan. Sentinel events as a fresh look at an old problem: Prospective study in prognosticating hospitalized palliative patients. Draft summary document, 2010.

[122] Vincent Thai, Francis Lau, Gary Wolch, Ju Yang, Hue Quan, and Konrad Fassbender. Impact of infections on the survival of hospitalized advanced cancer patients. *Journal of Pain and Symptom Management*, 43(3):549 – 557, 2012.

[123] Katrin Tomanek, Joachim Wermter, and Udo Hahn. Sentence and token splitting based on conditional random fields. In *PACLING 2007 – Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 49–57. Melbourne, Australia, September 19-21, 2007. Melbourne: Pacific Association for Computational Linguistics, 2007.

[124] Shulamit Umansky-Pesin, Roi Reichart, and Ari Rappoport. A multi-domain web-based algorithm for pos tagging of unknown words. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1274–1282, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[125] Ozlem Uzuner. Recognizing obesity and comorbidities in sparse data. *J Am Med Inform Assoc*, 16(4):561–70, 2009.

[126] Ozlem Uzuner, Ira Goldstein, Yuan Luo, and Isaac Kohane. Identifying patient smoking status from medical discharge records. *J Am Med Inform Assoc*, 15(1):14–24, 2008.

[127] Ozlem Uzuner, Imre Solti, and Eithon Cadag. Extracting medication information from clinical text. *J Am Med Inform Assoc*, 17(5):514–8, 2010.

[128] Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *J Am Med Inform Assoc*, 18(5):552–6, 2011.

[129] Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th conference on Computational linguistics*, pages 1106–1110, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[130] Joachim Wermter and Udo Hahn. Really, is medical sublanguage that different? experimental counter-evidence from tagging medical and newspaper corpora. *Stud Health Technol Inform*, 107(Pt 1):560–564, 2004.

[131] Richard Wicentowsk and Matthew Sydes. Using implicit information to identify smoking status in smoke-blind medical discharge summaries. *Journal of the American Medical Informatics Association*, 15(1):29–31, 2008.

[132] Jesse O Wrenn, Peter D Stetson, and Stephen B Johnson. An unsupervised machine learning approach to segmentation of clinician-entered free text. *AMIA Annu Symp Proc*, pages 811–5, 2007.

[133] Hua Xu, Kristin Anderson, Victor R Grann, and Carol Friedman. Facilitating cancer research using natural language processing of pathology reports. *Stud Health Technol Inform*, 107(Pt 1):565–572, 2004.

[134] H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3. Citeseer, 2003.

[135] Li Zhou and George Hripcsak. Temporal reasoning with medical data–a review with emphasis on medical natural language processing. *J Biomed Inform*, 40(2):183–202, 2007.

[136] Li Zhou, Genevieve B Melton, Simon Parsons, and George Hripcsak. A temporal constraint structure for extracting temporal information from clinical narrative. *J Biomed Inform*, 39(4):424–439, 2006.

[137] David Zingmond and Leslie A. Lenert. Monitoring free-text data using medical language processing. *Comput. Biomed. Res.*, 26(5):467–481, 1993.