Defining, Extracting, and Applying Events in NLP Tasks for Clinical Corpora

Prescott Klassen

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Fei Xia, Chair

Meliha Yetisgen, Chair

Emily M. Bender

Lucy Vanderwende

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Defining, Extracting, and Applying Events in NLP Tasks for Clinical Corpora

Prescott Klassen

Co-Chairs of the Supervisory Committee:
Professor Fei Xia
Department of Linguistics

Professor Meliha Yetisgen
Department of Bioinformatics and Medical Education

This dissertation explores defining, extracting, and applying clinical events in three studies of applied clinical natural language processing (NLP)—pneumonia report classification, acquired lung injury (ALI) report classification, and critical follow-up recommendation sentence identification. The goals of the research are to: (1) define a set of events for the clinical domain, (2) develop a clinical corpus of event annotations, (3) extract event representations from clinical records, and (4) apply event representations to multiple NLP tasks in the clinical domain. In order to repeat processes and adapt general research methodologies to the specific requirements of each study, a framework is created for event analysis, corpus development, event detection, and event-based feature extraction.

The pneumonia report classification study introduces a sub-corpus of rationale text snippets extracted from a corpus of X-ray reports, labeled for suspicion of pneumonia (PNA) and Cardio Pulmonary Infection Score (CPIS), and annotated for change-of-state and diagnosis events. Events are realized as dependency trees in the annotation and a three-stage event detection process is developed to extract events: (1) rationale snippet prediction by maximum entropy-based text classification, (2) conditional random fields (CRF) named entity recognition (NER), and (3) relation extraction (RE) by dependency parsing. Event-based

features are generated from the change-of-state and diagnosis event dependency trees and their performance, alone and in combination with baseline n-gram features, is evaluated in pneumonia report classification experiments. In final experimental results, incorporating χ^2-ranked feature selection and an optimal feature selection threshold, event-based features in combination with baseline n-gram features perform best for both PNA (F-score +.5) and CPIS (F-score +2.0) labels.

To explore the adaptability of the change-of-state and diagnosis events to other disease detection tasks, the second study applies the three-stage event detection process and modules from the pneumonia report classification study to the task of ALI report classification. In final experimental results, incorporating χ^2-ranked feature selection and an optimal feature selection threshold, change-of-state and diagnosis event-based features, alone and in combination with baseline n-gram features, do not improve the overall performance over the baseline (F-score -.6).

In the third and final study, an alternate, non-dependency tree-based model for event representation is explored for critical follow-up recommendation sentence identification. A corpus of 8,000 radiology reports from multiple institutions and across twelve modalities, is annotated for: (1) critical recommendation sentences, (2) entities that provide a reason for the recommendation, a suggested follow-up test, and a recommended timeframe for the follow-up test, as well as (3) a four-label category of criticality and importance. To improve the performance of recommendation sentence categorization, a template of report properties, metadata, named entities, and default computed values is aggregated into an event structure for feature extraction and compared against and in combination with baseline n-gram features in classification experiments. In final experimental results, select event-based features in combination with baseline n-gram features, incorporating χ^2-ranked feature selection and an optimal feature selection threshold, perform best (F-score +2.0) in critical recommendation classification experiments.

The research in this dissertation demonstrates that event-based features, when combined with other types of features, such as n-grams, can improve the performance of applied clinical NLP classification tasks. Simple models for events, such as the dependency tree structures for change-of-state and diagnosis events described in this study, make the annotation of events and event detection with off-the-shelf open-source tools easy to explain and straightforward to implement. The release to the Web of a general research framework, an annotated corpus for change-of-state and diagnosis events, an annotation schema and guidelines, and event detection modules based on open source software, provides opportunity for other researchers to extend and adapt the research presented in this study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ALI: acquired lung injury

BIME: Biomedical Informatics and Medical Education

BIONLP: biomedical or bioinformatic natural language processing

CUI: Concept Unique Identifier

CPIS: Clinical Pulmonary Infection Score

EHR: electronic health record

IAA: inter-annotator agreement

IE: information extraction

IRB: institutional review board

NER: named entity recognition *or* recognizer

NLP: natural language processing

RE: relation extraction

UMLS: unified medical language system

SVM: support vector machine

VAP: ventilator acquired pneumonia

ABBREVIATIONS AND TERMS USED IN RESULTS TABLES

Abbr	Term	Description
S	System	Total items generated by system
G	Gold	Total items in gold standard
Θ	Threshold	Threshold value used for feature selection
TP	True Positive	# of items classified as true by both the system and the gold standard
TN	True Negative	# of items classified as false by both the system and the gold standard
FP	False Positive	# of items classified as true by the system and false by the gold standard
FN	False Negative	# of items classified as false by the system and true by the gold standard
P	Precision	% of true system results that are true items in the gold standard: i.e., TP/(TP + FP)
R	Recall	% of true items in the gold standard that are true system results: i.e., TP/(TP + FN)
F1	F-score	F-score with β of 1 where F-$\beta = (1+\beta^2)*(P*R)/(\beta^2+P+R)$ or more simply 2 * ((P*R)/(P+R))
Acc	Accuracy	The percentage of correct system results (TP + TN)/(TP+TN+FP+FN) or TP/N when no TN are calculated (N is total items)
Macro	Macro-averaged	Depending on type of experiment, an average of multiple types and/or an average of averages across folds
Micro	Micro-averaged	Depending on type of experiment, results of multiple type and/or fold experiments calculated with total TP, FN, and FP values

Table 1: Common abbreviations and terms featured in performance and evaluation results tables

ACKNOWLEDGMENTS

I would like to thank my co-chairs, Dr. Fei Xia and Dr. Meliha Yetisgen, and the members of my reading committee, Dr. Emily M. Bender and Dr. Lucy Vanderwende, for their support and encouragement over the last four years. I would also like to thank Dr. Will Lewis for his contributions as a member of my reading committee for my General Exam. I would like to thank my fellow students in the Linguistics department and past and present members of the UW bioNLP Lab, especially Dr. Wen-wai Yim, Dr. Lucas McCarthy, Elena Pellicer, and Michael Tepper, for their help annotating the pneumonia report classification and critical recommendation corpora I used in my research. I would like to thank Dr. Martin Gunn and Dr. Thomas Payne for their help in developing the importance and criticality category labels for classifying critical follow-up recommendations and their effort vetting and annotating sentences in the critical recommendation multi-institutional corpus. I would also like to thank Joyce Parvi and Mike Furr for their administrative support throughout my PhD program, and David Brodbeck and Zachary Eagle for their help installing, configuring, and maintaining the often complicated technical environments I requested for my research and experiments. I would also like to thank my fellow PhD students, Ryan Georgi and Michael Goodman, for their friendship and advice over the years.

I would like to thank my wife for her support and all of the sacrifices she has made while I have been pursuing my PhD. I would also like to thank my parents for always being there for me with love, support, and encouragement.

DEDICATION

To Monica Lisafeld, my partner and best friend, and to my parents, Peter and Luella Klassen.

Chapter 1
INTRODUCTION

The electronic health record (EHR) is the primary repository of patient information in modern healthcare systems and contains both structured and semi-or-unstructured data. While structured data, such as instrument readings and test results, come to mind as the most common type of data in a patient's record, it is actually semi-or-unstructured data, in the form of narrative free-text reports, that makes up the largest portion of patient data in the EHR (Chapman and Cohen, 2009). The shift to digital records, over the last ten years, has transformed clinical informatics and created a new branch of biomedical natural language processing (bioNLP), *clinical NLP*, focused on the EHR and applied tasks in the clinical domain.

In the general domain,[1] statistical NLP systems have had a long history of harnessing the linguistic features of text to accomplish applied tasks. Tools such as tokenizers, part-of-speech taggers, and phrase structure parsers have contributed basic linguistic information and features to solving text processing challenges for decades. One of the most simple features, the *word unigram*, can be extracted from text with an off-the-shelf tokenizer or word segmenter, and is useful, because of its simplicity, for creating baseline results across tasks, domains, and genres. As EHR management systems integrate NLP components, it is now possible to encode into clinical narrative corpora deeper layers of both syntactic and semantic information and create systems to extract comprehensive clinical information from

[1] In this study, the term general domain refers to a genre- or task-neutral domain of language, for example, English or Chinese. It it used to differentiate NLP tools and approaches designed to apply broadly to any text in English as opposed to those designed for specific domains and tasks such as biomedical literature or clinical text.

them (Albright et al., 2013). Deeper linguistic structures, native to the domain or genre, can add to a better understanding of the entities and relations contained within the body of clinical text and empower a layer of semantic annotation beyond shallow features such as the word unigram.

My research is motivated by clinical events, which are important actions or observations of patient state, documented in the narrative medical record. Admission notes, discharge summaries, and progress reports are examples of clinical records that document important events such as admission, medication, discharge, drug interaction, diagnosis, and change of state. Marking events in clinical records as entities connected by relations—directed, labeled arcs—communicates more meaningful and more connected information about a patient's state over time than traditional bag-of-words or bag-of-concepts approaches. In this study, I explore research questions about events in clinical text and measure the impact of features extracted from clinical events on the performance of systems applied to three NLP tasks in the clinical domain: (1) pneumonia report classification, (2) acute lung injury (ALI) report classification, and (3) critical follow-up recommendation identification.

I begin this chapter by exploring three studies (see Table 1.1) that motivate the research questions and goals of my dissertation in Section 1.1. In Section 1.2, I pose two fundamental research questions concerning clinical events and discuss how I will address those research questions by setting four overall research goals in Section 1.3. In Section 1.4, I briefly outline my overall research methodology and in Section 1.5, I discuss the potential contribution and significance of my research and conclude with a description of the overall structure and organization of my dissertation.

1.1 Background

The focus of my research—defining, extracting, and applying events to clinical NLP tasks—is motivated by the potential to improve the performance of feature-based classification systems described in three studies, each of which introduced a corpus of annotated radiology reports and a clinical NLP system that used supervised machine learning to classify and label clinical

text in the context of an applied task. In all three studies, after extracting different types of features, such as syntactic features from part-of-speech tagging or medical vocabularies, the authors found n-gram features the highest ranked by feature selection and best performing in their experiments. See Table 1.1 for a listing of the three clinical NLP applied tasks and related studies.

Clinical NLP Task	Related Research
Acute lung injury (ALI) report classification	Yetisgen-Yildiz et al. (2011), Xia and Yetisgen-Yildiz (2012), and Yetisgen-Yildiz et al. (2013b)
Pneumonia report classification	Xia and Yetisgen-Yildiz (2012), Bejan et al. (2012), Bejan et al. (2013b), Tepper et al. (2013), Bejan et al. (2013a), and Vanderwende et al. (2013)
Critical follow-up recommendation identification	Yetisgen-Yildiz et al. (2013a)

Table 1.1: A listing of applied clinical NLP tasks and related studies that motivate clinical event research

1.1.1 Pneumonia report classification

Xia and Yetisgen-Yildiz (2012) introduced a corpus and an NLP system to detect patients with ventilator-associated pneumonia (VAP) from radiology reports drawn from the EHR system of the University of Washington Harborview Medical Center. The system was developed as a component of a future VAP phenotype detection project. VAP results when a patient acquires pneumonia while supported with mechanical ventilation. The early stage detection of VAP is difficult due to non-specific symptoms, subjective assessment criteria, and the lack of a single test to conclusively diagnose the disease (Zilberberg and Shorr, 2010). Clinicians must continually monitor and weigh multiple pieces of patient data at multiple

points of time to make their diagnosis. An automated NLP-based system, as described in Xia and Yetisgen-Yildiz (2012), can integrate text-based information sources, such as radiology reports, into an overall disease surveillance system and result in early detection.

The corpus introduced by Xia and Yetisgen-Yildiz (2012) is made up of narrative chest X-ray reports annotated by medical experts for suspicion of pneumonia (PNA) and Clinical Pulmonary Infection Score (CPIS)[2] (Zilberberg and Shorr, 2010). See Table 4.1 for the CPIS and PNA labels used to annotate the pneumonia report classification corpus. Tepper et al. (2013) extended the corpus by asking medical expert annotators to try to identify, within each report, one or more text snippets that provided a concise rationale for the report's CPIS and PNA category labels. They called the highlighted spans of text *rationale snippets*. In the majority of reports, a single snippet provides the rationale for both a CPIS and a PNA label. Using the snippet annotations and a supervised machine learning approach, they trained and tested a snippet prediction module, for both CPIS and PNA label categories, that automatically identified and extracted snippets from X-ray reports. They called these automatically extracted text spans *predicted* rationale snippets.

To evaluate the information value of rationale text snippets in the pneumonia report classification task they conducted a series of experiments, for both CPIS and PNA, where they extracted a common set of feature types from the text of: (1) an entire X-ray report, (2) the gold standard rationale snippet annotations in an X-ray report, (3) the predicted rationale snippets in an X-ray report, and (4) the combination of a snippet type and an entire X-ray report (Tepper et al., 2013). The results of the pneumonia report classification experiments for CPIS and PNA demonstrated that features extracted from the text of rationale snippets alone, both *oracle* (2) and *predicted* (3), out-performed the *baseline* (1) and all other configurations (4). See Section 4.2.1 for detailed results.

Error analysis in Tepper et al. (2013) revealed that there are limitations when experiments are based exclusively on shallow text features, such as unigrams. For the snippet, *The*

[2]http://www.surgicalcriticalcare.net/Resources/CPIS.php

previously noted right upper lobe opacity consistent with right upper lobe collapse has resolved, the system made an error and mislabeled the snippet because it supported one category entirely up to but not including the crucial words *has resolved* (Tepper et al., 2013). The system's basic word unigram feature set only provided words and their frequencies to the classifier and did not convey the semantic importance of the words *has resolved* to the meaning of the sentence. An annotation structure capable of explicitly linking and describing the relationship between the words *has resolved* and the observations that precede it in the example sentence could provide the system with the correct information that previously observed symptoms of disease, lung opacity and collapse, have resolved. In Section 1.1.2, I discuss the work of Vanderwende et al. (2013) who proposed a structure, the change-of-state event tuple, to address the limitations of n-gram models in describing complex events.

Bejan et al. (2013a) explored using a novel unit of classification based on time to detect pneumonia in clinical reports. They introduced an annotated corpus of 1040 intensive care unit (ICU) reports, including admit notes, ICU input records, and discharge summaries, for a patient cohort of 100 over a timeline of eight days. Each labeled instance for pneumonia prediction, positive or negative, was defined as a a collection of reports from a specific day on the eight-day timeline and a *look-back* period of n days. Features were extracted from the set of reports defined by timeline parameters.

Bejan et al. (2013a)'s novel approach to feature extraction, expanded on in Bejan et al. (2013b), was based on assertion classification. They related Unified Medical Language System (UMLS)[3] concepts embedded in expressions to assertion categories assigned to the expression by an assertion classifier. These semantically-motivated assertion features were then used in the pneumonia detection experiments and improved performance overall by F-score +5.75 when compared to baseline features in a configuration where the *look-back* period was set to 1 day (Bejan et al., 2013a). The results demonstrated that semantically-motivated feature types, such as assertion classification features, can improve the performance of pneu-

[3]https://www.nlm.nih.gov/research/umls/

monia detection tasks when integrated with other feature types, such as n-grams.

1.1.2 The change-of-state event

Vanderwende et al. (2013) reviewed the results of experiments and error analysis in Tepper et al. (2013) and concluded that the majority of text in rationale snippets described observations of change in a patient's state (for example, *minimal patchy atelectasis in the right lung is seen, mildly improved since the prior study*). Also, rationale snippets frequently included diagnostic statements based on an interpretation of observations of change (for example, *Bilateral patchy pulmonary opacities may be secondary to pneumonia or edema*). Observations of change-of-state appeared more often in snippet text than in non-snippet text.[4] This insight motivated Vanderwende et al. (2013) to propose a change-of-state event. Previous change-of-state analyses, for example Sun et al. (2013) and Saurı et al. (2006), focused on an analysis where events are expressed as verbs. In the pneumonia report classification chest X-ray corpus, report sentences can drop subject or object arguments, and even the verb itself, partially due to a terse style. More often than not, change-of-state events are expressed as nouns and adjectives that imply a missing verb or other syntactic constituent that would make the sentence fragment more complete. Examples from the X-ray corpus include: *interval increase*, *stable*, *decreasing*, *no change*, and *persistent*.

To implement their model of a change-of-state event, Vanderwende et al. (2013) introduced a change-of-state event tuple. See Figure 1.1 for an example of a change-of-state n-tuple and Section 4.2.2 for a detailed explanation of each n-tuple slot and its relation to the original text of the rationale snippet.

---

[4]Taking a random sample of 100 snippets, the researchers found that 83/100 included some signal for change of state, while a random sample of 100 non-snippet sentences included only 61/100 mentions of change-of-state.

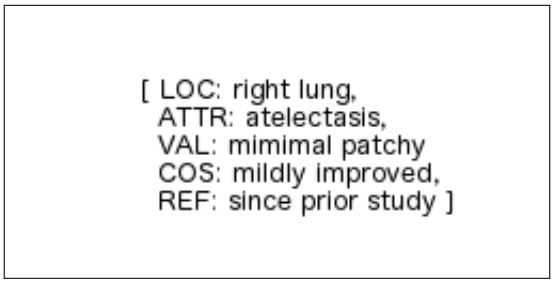

Figure 1.1: An example of an event tuple

In summary, Vanderwende et al. (2013) proposed a new type of annotation for the pneumonia report classification corpus, a change-of-state event, which is modeled as an n-tuple of five fields: `Cos`, `Attr`, `Val`, `Loc`, and `Ref`. The motivation for this new annotation was to address the gap of meaning between the language of the radiology reports and the pneumonia report classification system described in Tepper et al. (2013). For the first study in my dissertation, I introduce an alternate model to the change-of-state n-tuple based on a dependency tree structure.

1.1.3 Adapting change-of-state events to a new applied NLP task

In the first study of my dissertation, pneumonia report classification, I define change-of-state and diagnosis events based on an analysis of rationale snippets extracted from a corpus annotated with PNA and CPIS labels for VAP phenotype detection. To explore the adaptability of the change-of-state and diagnosis event extraction modules and feature types to other similar disease report classification tasks, I select a corpus and clinical NLP system for ALI detection as described in Yetisgen-Yildiz et al. (2013a). *ALI is a critical illness consisting of acute hypoxemic respiratory failure with bilateral pulmonary infiltrates that is associated with pulmonary and non-pulmonary risk factors* (Rubenfeld et al., 2005).

Yetisgen-Yildiz et al. (2013a) introduced an annotated corpus and developed a statistical NLP system for ALI report classification. Three Critical Care Medicine specialists annotated 1748 chest X-ray reports, generated for 629 patients from the Harborview Medical Center,

with labels *consistent* (positive) or *non-consistent* (negative) with ALI. The corpus was used to test and train a maximum entropy (MaxEnt) classification system with the MALLET[5] (McCallum, 2002) machine learning toolkit and an n-gram-based feature set. See Section 5.2 for performance results and a more detailed description of the study.

I compared the description of VAP and ALI in biomedical literature (Ware and Matthay, 2000; Zilberberg and Shorr, 2010) and determined, based on the similarity of the language used in the clinical narrative and a shared domain-specific vocabulary, the ALI and pneumonia report classification corpus have common linguistic features. For the second study of my dissertation, I apply change-of-state and diagnosis events, defined and developed on the pneumonia report classification corpus, to the ALI report classification corpus.

In order to evaluate the adaptability of my pneumonia report classification corpus-trained change-of-state and diagnosis event definitions and detection modules to the ALI task, I extract rationale snippets and change-of-state and diagnosis events from the ALI reports, generate event-based features, and integrate them into new ALI report classification experiments. I compare the results with my replicated baseline experiments and evaluate the adaptability of my snippet, named entity recognition (NER), and relation extraction (RE) modules trained on pneumonia report classification data to the ALI task.

1.1.4 Critical follow-up recommendation events

In the last decade, the use of imaging in patient care has dramatically increased, providing more diagnostic and screening opportunities for radiologists (Hendee et al., 2010). Given the rise in volume of imaging, the radiologist is challenged with reporting clinical information in an optimal way, avoiding communicating unimportant observations and interpretations of what they see in imaging that would be obvious to the ordering provider. At risk is the timely reporting of incidental findings and *critical follow-up recommendations* that may significantly impact the health of the patient in the short or medium term (Berland et al.,

[5]http://mallet.cs.umass.edu/

2010).

Yetisgen-Yildiz et al. (2011) introduced an annotated radiology imaging report corpus and NLP system to identify critical follow-up recommendations in radiology reports. Their goal was to integrate an NLP module into an overall EHR system to assist in the communication of important incidental findings between radiologist and clinician. In a subsequent study, Yetisgen-Yildiz et al. (2013b) annotated a small corpus of radiology reports and developed a gold standard to train and test a critical follow-up recommendation sentence identification application. See Section 6.2 for performance results and a more detailed description of the previous study.

For the third study in my disssertation, I explore representing a critical follow-up recommendation as an event. Unlike the change-of-state and diagnosis events I define for the pneumonia report classification corpus, I propose the structure of a critical follow-up recommendation event as a template of properties, rather than entities connected to one another with labeled arcs in a dependency tree structure.

1.2 Research questions

The change-of-state, diagnosis, and critical recommendation events that I propose in this dissertation are examples of task-specific events in the clinical domain that convey speaker meaning specific to the dialogue between radiologist and clinician. As opposed to event analysis in general linguistics, my exploration of events is constrained to the clinical narrative and a task-oriented focus—exploring a semantics of speaker meaning rather than the semantics of sentence meaning (Bender et al., 2015).

Two questions frame the research in my dissertation:

1. What are the components and constraints of a semantic representation that can describe speaker-meaning in a task-specific analysis of events in the clinical domain?

2. How does the analysis contribute to applied NLP tasks in the clinical domain such as classifying radiology reports to detect disease or critical follow-up recommendation identification?

In this study, entities, relations, and schema rules are examples of the *components and constraints of a semantic representation* I use to describe clinical event structures. By *speaker meaning*, I am describing the communication between a radiologist and a clinician or ordering provider concerning the radiologist's observations and interpretations of imaging data. By *task specific*, I am referring to an analysis of events constrained to a specific clinical task or domain, for example, pneumonia report classification.

1.3 Research goals

The goals of my research are to:

1. **Define a set of events for the clinical domain**—I analyze corpora and NLP systems for three clinical NLP tasks to define and mark events in clinical text that capture semantic information unique and valuable to the task.
2. **Develop a clinical corpus of event annotations**—I develop, in collaboration with NLP research annotators, a corpus of event annotations to test and train event extraction modules and apply them to NLP classification tasks.
3. **Extract event representations from clinical records**—I develop, train, test, and evaluate statistical NLP system modules that automatically extract event representations, based on task-specific definitions, from clinical records.

4. **Apply event representations to multiple NLP tasks in the clinical domain**—I use the extracted representations of events, the results of goals 1 through 3, as a source of features for three applied tasks: (1) pneumonia report classification, (2) ALI report classification, and (3) critical follow-up recommendation identification.

1.4 Overview of research methods

In this section, I briefly summarize the research methodology I used in my dissertation. A more thorough exposition of these research methods can be found in Chapter 3.

Each of the three applied NLP task studies in my dissertation has an unique goal, but all share common aspects: a corpus of radiology narrative reports, event representation and detection processes, and event-based feature extraction for classification. I applied a subset of the overall research methods I describe in this section to each task, adapting a more general framework to the specific needs of each study. See Table 1.2 for an overview of the major stages and steps in the framework.

1.5 Significance of the study

Clinical NLP applications benefit from extracting features for applied tasks from a deeper analysis of linguistic structure and semantics than basic features, such as n-grams, generated and extracted with general domain NLP tools (Albright et al., 2013). Current research in event extraction in the clinical domain mostly concerns the identification and relation of concepts and named entities from monolithic domain specific ontologies such as the National Institutes of Health (NIH)'s UMLS or the European GALEN Common Reference Model.[6] For example, the 2014 Sem-Eval Shared Task for clinical text[7] required participating teams to submit combined supervised and unsupervised systems that mapped entities and their relations to UMLS Concept Unique Identifier (CUI)s.

---

[6]http://www.opengalen.org/

[7]http://alt.qcri.org/semeval2014/task7/

Research Stage	Research Steps and Sub-steps
Event analysis	1. build or replicate a baseline system 2. repeat experiments 3. analyze the errors from replicated experiments 4. design a preliminary model for events in consultation with domain experts
Corpus development	1. Corpus preparation (a) prepare corpus (b) identify annotators (c) amend institutional review board (IRB) agreement (d) identify annotation tool (e) design annotation schema 2. Corpus annotation (multiple rounds) (a) create annotation guidelines (b) train annotators (c) annotate corpus (d) measure inter-annotator agreement (IAA) 3. Corpus finalization (a) finalize gold standard (b) publish annotations
Event detection	1. develop event detection modules 2. evaluate modules
Event-based feature extraction	1. extract features 2. build classifiers

Table 1.2: Overview of research methodology

The event analysis in this study is less concerned with identifying named entities and concepts in the domain but rather their intersection with the clinical language in reports concerning changes in state, the diagnosis of a disease or symptom, or a critical incidental

finding in a radiology image that requires immediate follow-up.

The research in this study contributes a potential framework for modeling similar types of events across genres. For example, does a change-of-state event differ when the report corpus describes a different disease? If so, can the change-of-state analysis be easily adapted for another disease detection task? Can the models of change-of-state and diagnosis, which were developed and trained on pulmonary X-ray report corpora, be applied to different types of reports outside of the domain of imaging?

In this study, I adapt a single general framework of research methods for all three applied tasks in order to reduce the start-up time for each task and increase the opportunity to reuse strategies, tools, and software components across tasks. This research framework may be adaptable to other NLP research in the clinical domain and provide a template/framework for conducting new studies and experiments.

There are a small number of annotated clinical text corpora available that are de-identified and free of licensing or legal restrictions. The scarcity of available annotated corpora in the clinical domain hampers progress in the field. I intend to release the annotated pneumonia report classification rationale snippet corpus I describe in this study to the Web for other researchers to access and use in their own research.

To summarize, in this chapter I began with the exploration of previous research that motivates my research questions and goals (Section 1.1). I then posed the two fundamental research questions of my study in Section 1.2, and described the four research goals I used to structure my research in Section, 1.3. In Section 1.4, I briefly described and explained the set of research methods I use to structure my research and how they generalize across the approaches I have taken to the three applied NLP tasks I defined in my research goals. I then addressed the potential contribution and significance of my research in Section, 1.5.

The rest of the dissertation is organized as follows. Chapter 2 reviews related literature and previous research on events in the clinical domain, using the four phases of the research framework I introduce in Section 1.4 as a organizing structure. Chapter 3 describes the four phase research framework and provides a more in-depth exposition of each phase. Chapter 4

contains a study of the development of change-of-state and diagnosis events in the context of the pneumonia report classification applied task. In Chapter 5, I study the adaptability of change-of-state and diagnosis event models and tools to the task of ALI report classification and in Chapter 6, I review my study of the development of a critical follow-up recommendation event and the classification system created to detect and identify critical follow-up recommendation in the narrative of imaging reports. Chapter 7 discusses the results of all three studies and the final chapter, Chapter 8 provides a summary and conclusion of the dissertation.

Chapter 2
LITERATURE REVIEW

In this chapter, I organize my literature review around the four research framework phases I introduced in Section 1.4 and later describe in detail in Chapter 3:

1. event analysis
2. corpus development
3. event detection
4. event-based feature extraction

In Section 2.1, I examine different approaches to representing events and provide an overview of previous research that motivates my decision to model change-of-state and diagnosis events as dependency trees. In Section 2.2, I review both traditional approaches to NLP corpus development as well as characteristics and considerations of creating and distributing NLP corpora in the clinical domain. Section 2.3 describes state-of-the-art event detection for both biomedical literature and clinical narratives and in Section 2.4, I discuss how event-based features have been applied in clinical NLP studies.

2.1 Event analysis

In general domain NLP, events are represented in many ways: they can be realized as collections of simple entity pairs connected by a single binary relation, an n-tuple of slots for named entities or trigger words, a narrative multi-sentence summary, a topical cluster of documents, or a hierarchy of relations and connected entities represented as a graph (Luo et al., 2016). In the biomedical NLP (bioNLP) or clinical NLP domain, events can be complex phenomena, such as the interaction of molecules and genes in biomedical literature,

or important actions and interactions in the clinical narrative, such as the diagnosis of a patient or an adverse drug effect. Luo et al. (2016) argued that events and relations discussed in biomedical literature and clinical narratives, are part of a continuum of descriptions of semantic structure from its simplest form, a binary relation between two named entities, to its most complex, a graph of nested event relations involving many entities and labeled relations, arranged in a hierarchical structure. The more structure an event has along this continuum, the more informative it is about the underlying semantics of the words it describes.

Luo et al. (2016) provided a comprehensive overview of the last decade of innovation in bioNLP and clinical NLP relation extraction and state-of-the-art NLP systems for both biomedical literature and clinical narratives. Shared tasks in biomedical literature NLP, such as BioNLP'09, served as a catalyst for the development of NLP systems to detect and extract events. The events in BioNLP'09 were defined as bio-molecular interactions in scientific literature and were characterized as *bio-events* (Kim et al., 2009). For evaluation, the structure of the bio-event was defined as a n-tuple relation of two or more entity-based fields or slots. At a very high level, event detection in bioNLP is a combined named entity recognition (NER) and relation extraction (RE) task. For BioNLP'09, participating teams were given the same collection of identified named entities at the start of the shared task to focus on relation extraction rather than both NER and RE in the same task.

Although bio-events in the BioNLP'09 Shared Task were represented as n-tuples for evaluation, many of the highest scoring participating teams incorporated graph representation of events in their actual systems. Luo et al. (2016) argued that the graph representation of events is the most generalized and that almost all state-of-the-art methods for extracting events or relations are graph-based.

Vanderwende et al. (2013) introduced change-of-state as an event n-tuple, similar in structure to the BioNLP09 Shared Task bio-event (Kim et al., 2009). In my pneumonia report classification study, I introduce an alternate dependency tree representation of one or more change-of-state and diagnosis events. An algorithm converts between the two representations: One or more change-of-state or diagnosis events can be represented as a dependency tree of

entities and relations (directed, labeled arcs) or one or more n-tuples as illustrated in Section 1.1.2. In my critical follow-up recommendation identification study, the event is a collection or template of properties and not a dependency tree. See Section 6.4.2 for a listing of the properties that make up the critical follow-up recommendation template.

2.2 Corpus development

In the clinical domain, corpora consist of electronic health record (EHR) artifacts selected for a cohort of patients. A patient's EHR is made up of medical information, including both structured data, such as laboratory results and vital signs, and unstructured narrative clinical text, such as discharge summaries, radiology notes, and progress reports.

Clinical text makes up the largest portion of patient information in an EHR (Chapman and Cohen, 2009) and in the last ten years, to improve access to the information captured in clinical text, there has been an increase in interest in NLP systems designed to work with EHRs. However, the clinical domain poses challenges for general domain NLP tools and approaches (Demner-Fushman et al., 2009). Clinical text is comprised of notes, reports, and summaries that are often meant only for communication between medical professionals. Reports can be technical in nature, exhibiting a varied level of grammaticality and employing short fragmentary telegraphic phrases. Templates used to assist in the generation of free-text reports lead to repeated cut-and-paste terms and phrases. Technologies like speech-to-text, used to make reports and notes easier to create, add their own noise and errors to documents. Most importantly, whereas innovation and progress in general domain NLP was spurred on by open and accessible articles, journals, annotated corpora, and tools, the inherent limitations on access to these types of publications and research data in the clinical domain has made it much more difficult for researchers to work on shared tasks and corpora. Clinical text is usually governed by strong privacy policies and the legal requirement to protect patient information. In the United States, patient information is protected by laws such as the

Health Insurance Portability and Accountability Act (HIPAA).[1] Annotated clinical corpora are typically only available to sanctioned institutional researchers.

Manually annotating clinical text corpora is an expensive and time consuming yet important process if progress is to be made by clinical NLP researchers in the clinical domain. Xia and Yetisgen-Yildiz (2012) summarized the rationale for manual annotation discussed in Roberts et al. (2009): (1) creating annotation schema serves to focus and clarify the information requirements of the text processing task and the domain of interest, (2) annotated data serves as a gold standard to assess the performance of text processing systems, and (3) annotated data serves as a resource for developing rule-based systems or creating statistical models by the application of machine learning approaches.

2.2.1 An overview of biomedical and clinical corpora

Corpora in the biomedical domain are divided into two genres, biomedical literature and clinical text. Examples of corpora of the first genre, biomedical literature, include the GENIA corpus (Kim et al., 2003), PennBioIE corpus (Kulick et al., 2004), Yapex corpus (Franzén et al., 2002), and GENETAG (Tanabe et al., 2005). Corpora representative of the second genre, clinical text, is typically institutional and governed by privacy and legal policies that limit open access. To overcome the challenge of conducting open, shared research on clinical narrative corpora, groups such as the US National Institutes of Health (NIH) Clinical and Translational Science Awards (CTSA)[2] and Informatics for Integrating Biology and the Bedside (i2b2)[3] provide de-identified corpora and hosted shared tasks to promote research on clinical text. Each NLP challenge hosted by i2b2 contributes a corpus of de-identified clinical records. See Table 2.1 for a description of i2b2 Shared Tasks and related publications.

---

[1] http://www.hhs.gov/hipaa/for-individuals/guidance-materials-for-consumers/index.html

[2] https://www.ctsacentral.org

[3] https://www.i2b2.org/

Year	Shared Task	Task Description	Citation
2006	De-identification and Smoking Challenge	De-identification of clinical records and smoking history	(Uzuner et al., 2008)
2008	Obesity Challenge	Obesity and co-morbidities	(Uzuner, 2009)
2009	Medication Challenge	Medication named entity extraction	(Uzuner et al., 2010)
2010	Relations Challenge	Assertion and relation extraction	(Uzuner et al., 2011)
2011	Co-reference Challenge	Co-reference resolution	(Uzuner et al., 2012)
2012	Temporal Relations Challenge	Temporal relations	(Sun et al., 2013)
2014	De-identification and Heart Disease Risk Factors Challenge	De-identification and heart disease risk factors	(Stubbs et al., 2015)
2015	No task issued	-	-
2016	De-identification and RDoC Classification Challenge	De-identification and psychiatric symptom severity	-

Table 2.1: i2b2 Shared Tasks

2.2.2 Three approaches to corpus development

The traditional approach to annotating corpora for NLP research involves a team of guideline designers, professional annotators, language or domain experts, and technical support staff and can take years to complete (Xia and Yetisgen-Yildiz, 2012). Examples of corpora developed with a traditional annotation approach include the Prague Dependency Treebank (Bejček et al., 2013), the English/Chinese/Arabic Penn Treebank (Marcus et al., 1993; Xia et al., 2000; Maamouri and Bies, 2004), the English PropBank (Palmer et al., 2005), and the Penn Discourse Treebank (Miltsakaki et al., 2004).

Other approaches to corpus development have emerged in recent years in response to the long development time and costs of the traditional approach. They have developed as a result of modern global communication systems, enabling annotators to collaborate in real-time over geographic distances, and open source software, which reduces the cost of developing

a custom annotation tool for each project. Two of these new approaches are *crowd-sourced* annotation, for example named-entity extraction tasks in the biomedical domain (Yetisgen-Yildiz et al., 2010), and *community-based* annotation, such as the evaluation corpus used in the 2009 i2b2 medication challenge (Uzuner et al., 2010), and the Groningen Meaning Bank (Basile et al., 2012) online.[4] Crowd-sourced annotation projects take advantage of global online knowledge work marketplaces, such as Amazon's Mechanical Turk (AMT) where tasks and workers are matched in a simple Web-based interface. Community-based annotation projects harness the expertise and knowledge of a research community where there is no opportunity to create a centralized, trained group of core annotators in one location.

The traditional annotation process is limited by cost, time, and the need for expert annotators or domain experts. Crowd-sourcing works well when the annotation does not require domain expertise and can be quickly applied without extensive training (Yetisgen-Yildiz et al., 2010). Community-based annotation is typically distributed over geography and time and requires strong coordination and communication systems as well as the enthusiastic support of the research community (Uzuner et al., 2010).

2.2.3 Challenges of the clinical domain

Xia and Yetisgen-Yildiz (2012) adapt the traditional annotation process to accommodate the unique constraints of the clinical domain. They discuss three challenges of annotation projects in the clinical domain:

1. the need for medical expert annotators
2. privacy issues
3. legal considerations

[4] http://gmb.let.rug.nl/

They suggest strategies to adapt the traditional annotation process in the face of these challenges by:

1. change the role of the NLP researcher by transitioning responsibilities to medical expert annotators
2. involve the medical expert annotators in the development of annotation guidelines, training annotators, and anticipating legal considerations and privacy issues
3. structure time commitments from expert annotators
4. involve NLP researcher early in the process

I integrate the strategies from Xia and Yetisgen-Yildiz (2012) with a traditional annotation process for corpus development in my critical follow-up recommendation identification study. I involve the medical expert annotators early in the discussion of the critical follow-up recommendation event annotation model and give them the responsibility of defining categories and labels. I also simplify and streamline the annotation process to maximize the impact of their limited time commitment.

2.3 Event detection

BioNLP shared tasks on event mining,[5] BioCreative shared tasks on PPI creation,[6] and DDIExtraction challenges on DDI extraction[7] have encouraged progress and innovation in event detection for biomedical literature over the last ten years. Of the top performing teams in these community shared tasks challenges, the majority have embraced graph-based methodologies (Luo et al., 2016). In the following subsection, I highlight state-of-the-art graph-based approaches to event detection developed and evaluated on the BioNLP '09 Shared Task and data previously discussed in Section 2.1.

---

[5]http://www.nactem.ac.uk/tsujii/GENIA/SharedTask/

[6]http://biocreative.sourceforge.net/

[7]http://labda.inf.uc3m.es/DDIExtraction2011/index.html

2.3.1 BioNLP '09 shared task

Events in BioNLP'09 were defined as bio-molecular interactions in scientific literature and were characterized as *bio-events* (Kim et al., 2009). For evaluation, the structure of the bio-event was defined as a n-tuple relation of two or more entity-based fields or slots. Participating teams were provided with a collection of named entities and expected to extract relations between entities in the form of event-tuples from a shared corpus.

The best performing system from the BioNLP '09 Shared Task, submitted by the University of Turku, relied on a multi-stage process, combining state-of-the-art machine learning techniques, features based on a full dependency analysis of each sentence, and rule-based post-processing (Björne et al., 2009). Their novel insight was that grammatical relations output by the Stanford dependency parser had a strong structural relation to the semantic labels and edges of bio-events. This motivated event trigger and edge detection as a two-stage classification task utilizing multiclass SVM and a rich set of features from the dependency parse. The resulting semantic graphs were normalized in a rule-based post-processing step to deal with abnormalities in argument generation (zero or too many). Björne et al. (2010) applied the same system with improvements to generalized event extraction of un-annotated text at scale against a 1% representative sample of the PubMed database. Miwa et al. (2010) developed a similar system, based on the same pipeline architecture as the Turku system, but treating all three stages as machine learning classification tasks. They also used deep parsing technologies based on the HPSG formalism to leverage new types of structural features—Predicate Argument Structure (PAS)—similar to but different than the grammatical relations of the Stanford dependencies. Their system outperformed Björne et al. (2009) when applied to the BioNLP '09 Shared Tasks and data.

Another notable and well-performing submission of BioNLP '09 was Riedel et al. (2009). Their system was notable for integrating three approaches:

1. A joint discriminative model which identities the complete event structure for a sentence in one pass.
2. Markov Logic, which leverages a statistical relational learning language enabling a declarative global model.
3. The representation of events as relational structures over tokens of a sentence rather than abstract entities.

The system submitted by Riedel et al. (2009) placed 4th overall on Task 1 and 1st overall on Task 2. Poon and Vanderwende (2010), who also implemented a joint approach with Markov Logic, improved on results of Riedel et al. (2009) and were able to achieve state-of-the-art on the BioNLP '09 tasks by expanding the scope of joint inference to include individual argument edges.

In McClosky et al. (2011) an alternate joint approach to event extraction is described with competitive results on BioNLP '09 tasks and data. The strength of their approach is its generalizability across domains and genres. The process of transforming the bio-events in the BioNLP '09 data into dependency trees for training a dependency parser is a simple and straightforward task. It enabled their system to match state-of-the-art performance with very little additional domain-specific tuning. Their system is premised on the insight that the BioNLP '09 n-tuple-based representation of bio-events can be realized as a semantically-labeled dependency tree and used to train any dependency parser to extract bio-events. Entities (proteins pre-identified for the task) and event anchors (defined by the task) are mapped from the bio-event representation to a dependency tree with a virtual root node. Edges are then constructed based on the semantic arguments of the event anchor, which are identified by the labeled slots of the bio-event n-tuple. Only the words that participate in the event are labeled in the dependency tree. A dependency parser is trained on the annotated corpus of semantically-labeled dependency trees and its results are re-ranked to determine

the best final representation.

The BioNLP '09 Shared Task was a catalyst for the development of state-of-the-art approaches to event detection and extraction for biomedical literature NLP. The majority of the state-of-the-art systems evaluated and compared based on BioNLP '09 Shared Task data, are graph-based (Luo et al., 2016). Of those systems discussed in this section, McClosky et al. (2011) stands out as an example of an approach that performs very well when evaluated against BioNLP '09 data and is straightforward to train and evaluate using existing dependency parsers and evaluation tools.

2.3.2 Event extraction in the clinical domain

In the clinical domain, event detection and extraction tasks can be more specific and limited in scope than in the biomedical literature domain. This is largely due to the differences between target corpora as discussed in Section 2.2. Clinical text consists of a mixture of many common words for describing observations and patient state and complex domain-specific medical terms. Event detection in this context is focused on entity and relation extraction tasks with a narrow focus, for example, the detection of a specific disease or drug interaction. Systems developed for the clinical domain share common approaches (Luo et al., 2016) and a state-of-the-art configuration of a two-stage statistical NLP system for event detection and extraction might include a Conditional Random Fields (CRF)-based entity detection module for concepts and other types of controlled vocabulary and a Support Vector Machine (SVM)-based classification module to extract relations between entities based on patterns or ontology mappings.

In recent years, groups such as i2b2 and the NIHs Shared Annotated Resources (ShARe) project have put together shared tasks including event extraction to promote the development of shared resources and evaluation frameworks. The following paragraphs highlight i2b2 shared tasks and corpora.

The 2009 i2b2 Medication Extraction Challenge[8] (Uzuner et al., 2010) was organized around the extraction of medication-related information from narrative patient records. The corpus of discharge summaries offered as a resource to participants was not annotated, but annotation guidelines and sample annotated records were provided. Participants were encouraged to submit either supervised or unsupervised systems. Of the top 10 systems, six were self-described as rule-based and the other four systems, as hybrid approaches. Supervised machine learning approaches included CRF for named entity recognition and SVM classification for relation extraction. The University of Sydney combined CRF, SVM, and rules to attain the best performance.

The 2010 i2b2/VA Challenges in Natural Language Processing for Clinical Data[9](Uzuner et al., 2011) shared task included three tasks:

1. extraction of medical problems, tests, and treatments
2. classification of assertions made on medical problems
3. relations of medical problems, tests, and treatments

The data for the challenge was made up of de-identified discharge summaries and progress notes from multiple institutions. All of the patient records were manually annotated for concept, assertion, and relation information. Similar to the 2009 medication challenge, CRF-based systems performed best at concept identification and SVM-based classification systems for assertion and relation tasks. There were some novel approaches, including a semi-supervised CRF solution that used distributional semantics features (Jonnalagadda and Gonzalez, 2010) and multiple systems that used ensemble approaches.

[8]https://www.i2b2.org/NLP/Medication/

[9]https://www.i2b2.org/NLP/Relations/

The 2012 i2b2 Temporal Relations Challenge[10] (Sun et al., 2012) focused on temporal relations in clinical narratives. The task was broken down into three subtasks:

1. clinically significant events
2. temporal expressions
3. temporal relations between clinical events and temporal expressions

The corpus was made up of discharge summaries annotated with two types of annotations:

1. events and temporal expressions
2. temporal relations

For the first subtask, event detection, statistical machine learning approaches performed best. Similar to previous years, span-based events were best identified with CRF-based approaches, and event attributes best identified with SVM-based classification approaches. The system with the best performance in event detection was a collaboration between Beihang University, Microsoft Research Asia, Beijing, and Tsinghua University (Xu et al., 2013).

The ShARe/CLEF eHealth 2013 Shared Task[11] included two information extraction tasks:

1. identifying disorder mention spans and associating them with relevant UMLS CUIs
2. identifying acronym/abbreviation spans and associating them with relevant UMLS CUIs

The overviews of these tasks describe high performing systems, but neither detail the specific approaches of those teams (Pradhan et al., 2013; Murtola et al., 2013).

[10] https://www.i2b2.org/NLP/TemporalRelations/Main.php

[11] https://sites.google.com/site/shareclefehealth/

The event detection method I implement to extract the graph structure of change-of-state and diagnosis event dependency trees is based on a two-stage supervised learning approach. In the first stage, change-of-state and diagnosis event entities are annotated within the sentence boundaries of unique rationale snippets and used to train a CRF-based NER module. In the second stage, directed labeled arcs (relations) are drawn between entities labeled by the NER module in the first stage and one or more dependency trees are created. The dependency trees are used to train a dependency parser, MaltParser, to automatically extract events from entity-labeled rationale snippet sentences.

Although my approach uses a dependency parser and a graph representation, it does not use syntactic dependency information to construct the graph. The dependency parser is simply used as a graph parser trained on domain-specific entities and directed, labeled arcs. It does not use any of the complex dependency and relation disambiguation or graph pattern matching that typifies state-of-the-art approaches to event-based relation extraction, such as McClosky et al. (2011) and Björne and Salakoski (2011), and other systems discussed in (Luo et al., 2016). I represent change-of-state and diagnosis events as graphs and extract the event graphs with a dependency parser, but my approach, using domain-specific entities and relations, is a basic supervised learning task. Unlike McClosky et al. (2011) and Björne and Salakoski (2011), I do not need to map an n-tuple representation to a dependency tree because the annotators in the VAP project explicitly label unique rationale snippets with the entities and relations of the change-of-state and diagnosis events.

2.4 Event-based feature extraction

In this dissertation, I explore integrating event-based features in applied NLP classification tasks and evaluating their impact on the performance when compared to and combined with baseline n-gram experiments replicated from previous studies. In the pneumonia report classification and ALI studies, event-based features are extracted from the semantic representations of change-of-state and diagnosis events which have been annotated by medical experts or event detection modules trained on pneumonia report classification annotations.

Unlike word ngrams, which are constructed based on surface strings using general domain NLP tools, event-based features in this study are generated from the semantic labels of event structures using tools trained on pneumonia report classification change-of-state and diagnosis events annotations.

In the clinical informatics literature, there are few examples of features extracted from semantic representations. Cai et al. (2016) present a comprehensive survey of NLP in research and clinical applications and find that n-gram-based features are the most common source for features in statistical machine learning approaches to clinical NLP tasks. They do, however, highlight two semantic features types especially important to clinical narrative NLP—chronicity and location—which have been extracted from temporal and anatomic location relations in recent studies of radiology reports (Yu et al., 2014; Chapman et al., 2011). Bejan et al. (2013a) extract features from intensive care unit (ICU) reports based on a patient timeline in order to predict the point on the timeline where the patient is positive for pneumonia. The timeline was designed on an eight-day scale, and a data instance, used in their pneumonia prediction experiments, was made up of reports for a specific day on the timeline and a *look-back period* of n days. Features were then extracted from the reports bound by the timeline parameters to create a feature vector for each patient in the pneumonia prediction experiments.

One of the benefits of graph-based event detection comes in the latter stages of the NLP processing pipeline when downstream components can mine the event graph for semantically enriched features for applied tasks (Luo et al., 2016). In my pneumonia report classification and ALI report classification studies I mine the event dependency graphs for features that I integrate into my experiments. These include patterns of relations between `Cos`, `Dhead`, `Attr`, `Val`, `Loc`, and `Ref` entities and their values. I have included in oracle experiments the attributes of `Cos` and `Dhead` entities, which are boolean values generalizing change-of-state and diagnosis values, such as *increased*, *changed*, and *hedge*. I use the negation and coordination entities at the appropriate scope in the hierarchy of relations in the event graph to negate and combine the values of entity and relation-driven feature types. See Appendix

A for a complete listing and description of all entities, relations, and attributes in the change-of-state and diagnosis event and diagnosis event schemas.

2.5 Summary

In this chapter, I described previous research and the state-of-the-art in bioNLP and clinical NLP by presenting a literature review organized around the four phases of my research framework I initially described in Section 1.4 and detail in Chapter 3:

1. event analysis (Section 2.1)
2. corpus development (Section 2.2)
3. event detection (Section 2.3)
4. event-based feature extraction (Section 2.4)

Section 2.1 reviewed how events are represented in the clinical domain and concluded that current state-of-the-art systems implement graph-based approaches to represent interconnected entities and relations as events. Section 2.2 discussed efforts to develop and distribute clinical NLP corpora, concluding with strategies for corpus development in the clinical domain. Section 2.3 surveyed approaches to event extraction in both the general and clinical domain, highlighting systems participating in the series of BioNLP Shared Tasks on event extraction that represent current state-of-the-art. Section 2.4 discussed the limited use of event-based features in clinical NLP studies and how I propose to implement event-based features in my applied task experiments. The next chapter presents the general research framework I implemented to address the methodology requirements of the three applied NLP tasks I address in my dissertation.

Chapter 3
RESEARCH METHODS

In Section 1.4, I introduced the overall methodology framework I use to structure the research in my dissertation and in Table 1.2, I provided a high-level overview of its four phases. In this Chapter, I describe the three applied tasks I target in my research and then present in detail each phase of my research framework, beginning with an overview in Section 3.2, and followed by a description of event analysis in Section 3.3, corpus development in Section 3.4, and event detection in Section 3.5. I conclude by discussing event-based feature extraction in Section 3.6. Each section outlines how each phase of the research framework addresses the methodology requirements of the three applied tasks.

3.1 Overview of the three applied tasks

In this dissertation, I create event-based NLP applications for three applied tasks: (1) pneumonia report classification, (2) acute lung injury (ALI) report classification, and (3) critical follow-up recommendation identification. All three tasks use features extracted from text and supervised learning to train classification models that categorize text components of radiology reports. See Table 1.1 for a list of publications and previous research related to the three tasks and Table 3.1 for a detailed description of each task.

Task	Description
Pneumonia report classification	Ventilator-associated pneumonia (VAP) is a pneumonia acquired by patients when they are connected to mechanical ventilation systems in a healthcare setting. The pneumonia report classification study classifies the narrative X-ray report text of a radiology corpus in order to detect disease. Supervised learning is used to train and test an NLP report classification system that labels reports with pneumonia (PNA) and Clinical Pulmonary Infection Score (CPIS) labels.
ALI report classification	Acute lung injury (ALI) is a disease of the lungs where they become acutely hypoxemic and a patient can suffer respiratory failure (Yetisgen-Yildiz et al., 2013a). . The ALI report classification study classifies the narrative X-ray reports of a cohort of patients to detect evidence consistent with a diagnosis of ALI. The study uses supervised learning to test and train an NLP system that labels patients with a negative or positive diagnosis of ALI.
Critical follow-up recommendation identification	A critical follow-up recommendation is a sentence in a narrative X-ray report that reflects a statement made by a radiologist in a radiology report to advise the referring clinician to further evaluate an imaging finding by either other tests or further imaging (Xia and Yetisgen-Yildiz, 2012). The critical follow-up recommendation study classifies the sentences in a range of radiology reports across modalities to identify critical follow-up recommendation. The study uses supervised learning to test and train an NLP system that labels sentences as critical follow-up recommendations. Additionally, it categorizes the recommendation sentence as being one of four types.

Table 3.1: Descriptions of the three applied NLP tasks

3.2 Introduction of a general research framework

In this study, I integrate event-based features into classification experiments by:

1. defining event models
2. adding event-based annotations to corpora
3. training and testing event detection modules
4. extracting event-based features for classification experiments

To accomplish these steps, I adapted a general framework of research methods to the specific needs of each study. The four basic phases in my general framework are: (1) event analysis, (2) corpus development, (3) event detection, and (4) event-based feature extraction. These phases are intended to be executed in sequential order and are further elaborated in Figure 3.1 and Figure 3.2.

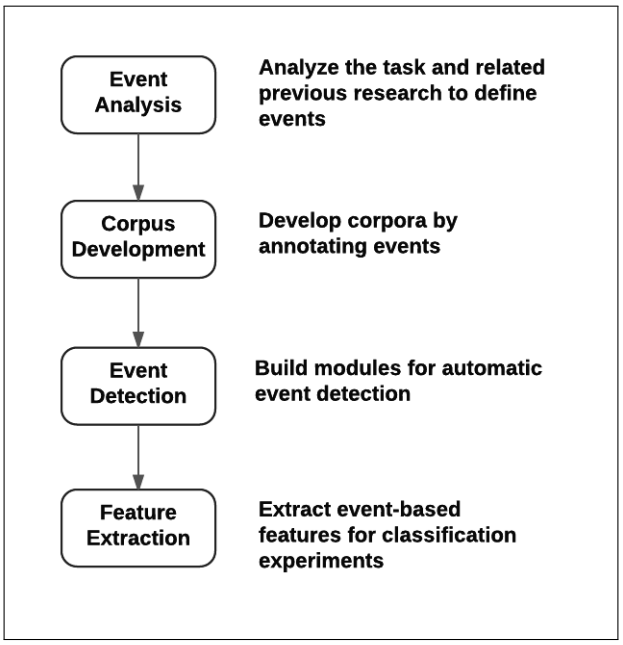

Figure 3.1: Overview of the major stages of the general research framework for developing event-based features

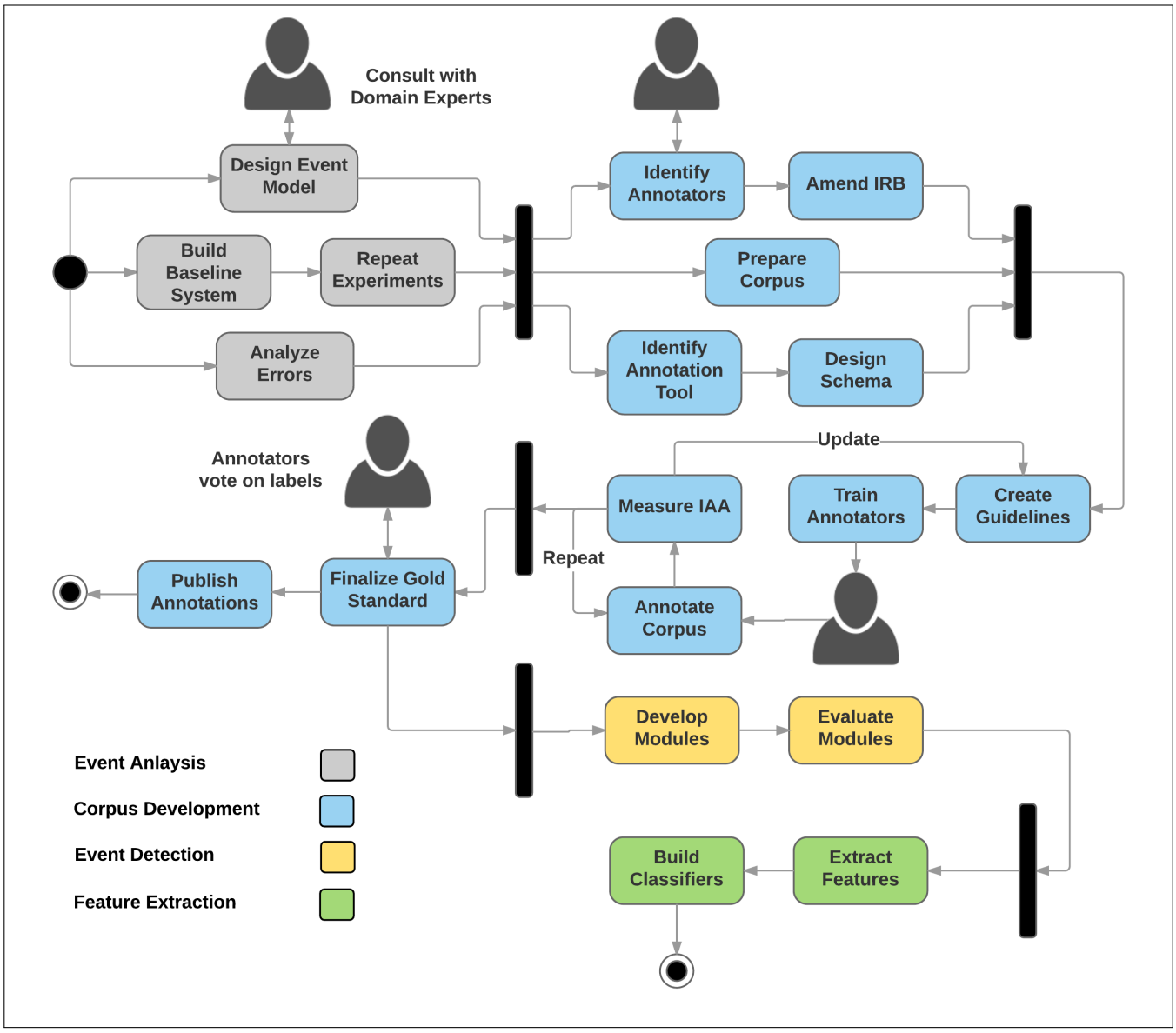

Figure 3.2: Methodology overview

The four stages in Figure 3.1 are described in more detail in Figure 3.2. Not depicted in the diagram is how the event-based features were integrated into a series of classification experiments specific to each NLP applied task. In all three studies, baseline n-gram and event-based features were each evaluated separately in a series of experiments and then

added together as a joint set of features for comparison.

3.3 Event analysis

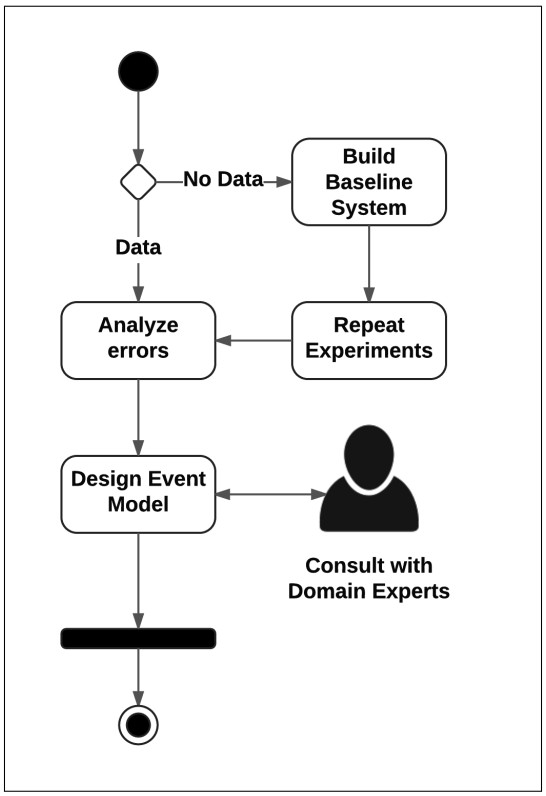

Figure 3.3: Event analysis workflow diagram

In the clinical NLP literature, events are represented in numerous ways—as dependency trees, tuples, slotted templates, or as simply a bag of properties. During the event analysis phase, previous studies, experimental results, error analysis, and domain experts contribute to the definition of events for specific applied clinical NLP tasks. The first step in defining events is gaining access to corpora, feature sets, software modules, and experiment settings of previous studies. In most cases this requires replicating, to some extent, the NLP system described in the original study. The original papers and articles of the three studies I describe

in this dissertation provided the necessary detail required to replicate results, including comprehensive descriptions of software, configuration files, and corpora.

Event analysis defines the logical event models that are translated into machine readable formats in subsequent phases of the framework. The main tasks of event analysis are:

1. **Build or replicate a baseline system** All corpora, NLP application modules, experiment settings, and dependencies on third party software packages for the three applied tasks in my study were well described in Tepper et al. (2013) for pneumonia report classification, Yetisgen-Yildiz et al. (2013b) for critical follow-up recommendation identification, and Yetisgen-Yildiz et al. (2013a) for ALI report classification.

2. **Repeat experiments** Using the system developed in the previous step I was able to recreate the configuration and settings from previous studies and replicate experiments for all three applied tasks. I used the replicated experimental results as a baseline to evaluate the impact of event-based features on classification performance.

3. **Analyze the errors from replicated experiments** Examining experimental results, especially detailed error analysis, such as an inventory of values and frequencies for false positives and false negatives, helps uncover events and their properties. I leveraged the analysis of classification errors in pneumonia report classification experiments described by Tepper et al. (2013) and Vanderwende et al. (2013) in order to define graph-based models for change-of-state and diagnosis events.

4. **Design a preliminary model for events in consultation with domain experts** A strong logical representation of an event model with documentation and examples translates into machine-readable representations for annotation and system processing. Previous research, annotation guidelines and early discussions with annotators and domain experts informed the change-of-state and critical follow-up recommendation event models I developed during the preliminary phase of each study.

Xia and Yetisgen-Yildiz (2012) documented the challenges of integrating medical experts into the annotation process. Domain expertise was especially impactful on the critical follow-up recommendation identification study. The medical expert annotators contributed a complex set of four category labels for the critical follow-up recommendation identification task that reflected their deep understanding of the domain.

See Figure 3.3 for a depiction of the interdependency and sequence of the event analysis methods. A description of these methods and how they relate to specific events is explored in the methodology sections of the individual studies detailed in Chapters 4 - 6.

3.4 Corpus development

In my research, two of the the three studies, pneumonia report classification and critical follow-up recommendation identification, required a corpus development phase. The third study, ALI report classification, explored the adaptability of change-of-state and diagnosis events defined for the pneumonia report classification task to a similar disease detection task. A corpus development phase was not required for the ALI task and therefore no additional annotations were created for this study.

The corpora developed for the pneumonia report classification and critical follow-up recommendation identification tasks each required an unique development process modified from the more general framework described in this section. The critical follow-up recommendation corpus was new and it was necessary to implement a complete end-to-end process involving medical expert annotators whereas the pneumonia report classification corpus was labeled by medical expert annotators in a previous study and for my research, a single NLP researcher alone annotated the pneumonia report classification rationale snippet sub-corpus for change-of-state and diagnosis events.[1]

[1]I performed the role of the single NLP researcher in the final round of annotation.

There are three stages of corpus development in the general research framework:

1. preparation
2. annotation
3. finalization

The following sections detail the three corpus development stages.

3.4.1 Corpus preparation

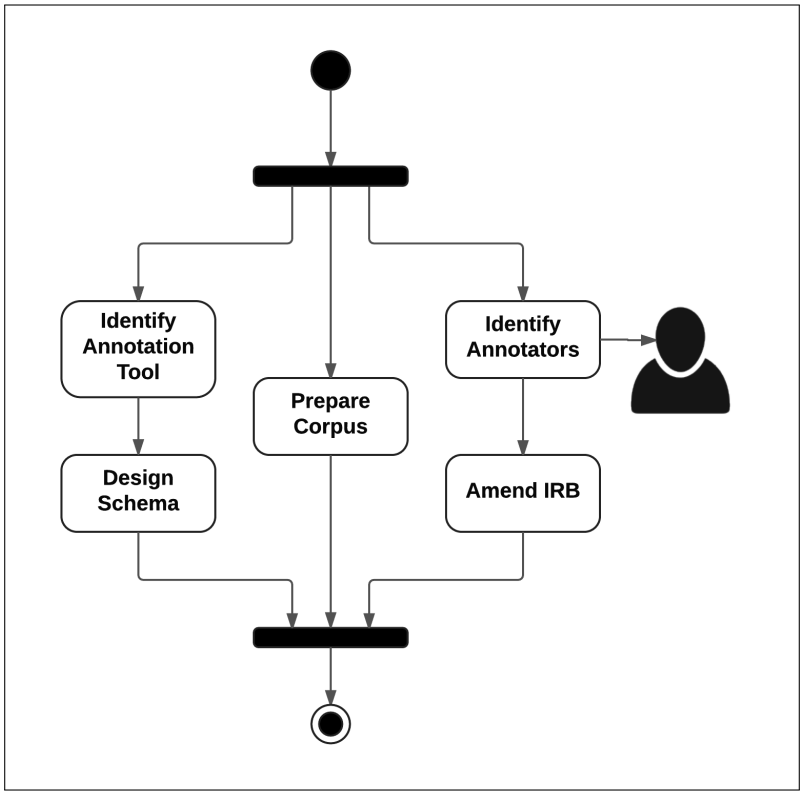

Figure 3.4: Corpus preparation workflow diagram

The corpus preparation stage consists of five main tasks:

1. **Prepare corpus** The original files and annotation data for pre-existing corpora can be stored in many different types of formats and infrastructure. Preparing the corpus for a new annotation environment or storage infrastructure requires pre-processing and conversion.

   The pneumonia report classification corpus was originally annotated for CPIS and PNA labels and rationale snippets in the GATE[2] (Cunningham et al., 2011) integrated development environment (IDE) for NLP. GATE stores annotations as mark-up islands in an XML file based on their own XML schema for annotation. I converted these files into two separate files types, a plain text file for the original report text, and a `.ann` annotation file formatted for the BRAT[3] (Stenetorp et al., 2012) annotation environment. A subsequent conversion utility was required to import and export the BRAT report and annotation files to and from a Microsoft SQL Server® database.

2. **Identify annotators** Selecting medical expert annotators requires matching domain knowledge with task and corpus. The original annotation process and medical expert annotator profiles for pneumonia report classification and ALI task were described in Xia and Yetisgen-Yildiz (2012), Tepper et al. (2013), and Yetisgen-Yildiz et al. (2013a). No additional medical expert annotators contributed to my pneumonia report classification and ALI studies.

3. **Amend institutional review board (IRB) agreement** Adding new members to a clinical informatics project may also require an amendment to the IRB agreement. All medical expert annotators for the critical follow-up recommendation identification study were recruited from the University of Washington Medical school and hospital

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

[3]http://brat.nlplab.org/index.html

system. They were all approved to participate by the human subjects committee (HSC) of the IRB.

4. **Identify annotation tool** Selection of an annotation tool should emphasize annotation format, usability, portability between operating systems, and ease of installation/configuration. I selected the BRAT[4] (Stenetorp et al., 2012) web-based annotation tool for all annotation development in this study.

5. **Design annotation schema** Annotation tools that are extensible for user-designed annotation schemas provide formats to describe schemas. BRAT uses a simple format of character offsets and labels to represent entities (text spans) and relations (labeled, directed arcs between entities). See Appendix B for detailed listings of multiple versions of the pneumonia report classification and critical follow-up recommendation BRAT schemas.

[4]http://brat.nlplab.org/index.html

3.4.2 Corpus annotation (multiple rounds)

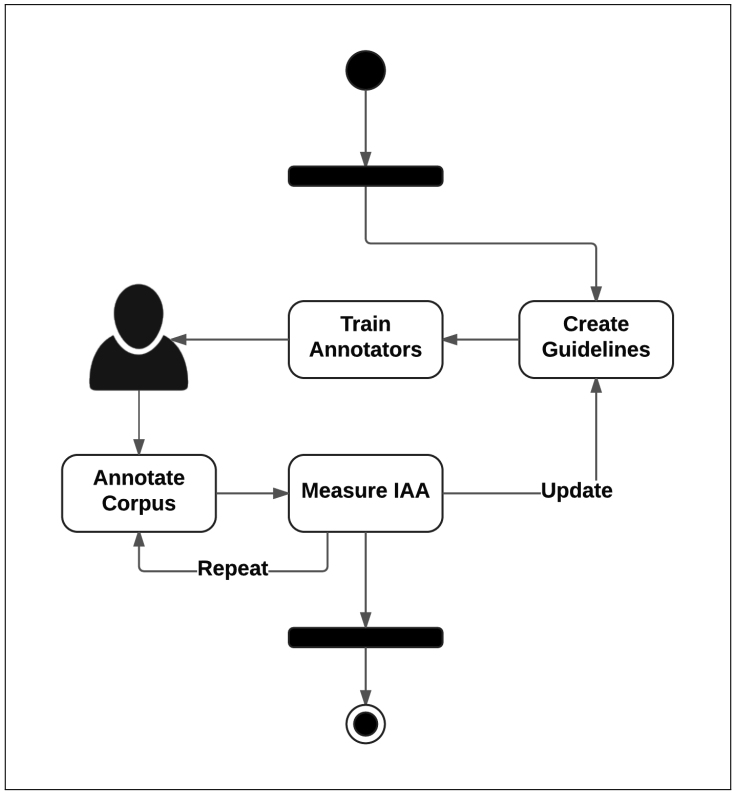

Figure 3.5: Corpus annotation workflow diagram

The corpus annotation stage consists of four main tasks:

1. **Create annotation guidelines** I followed the traditional annotation process with adaptations for the medical domain as described in Xia and Yetisgen-Yildiz (2012). Guidelines were integrated into the process from the beginning of event analysis and revised and edited after each inter-annotator agreement (IAA) round until finalized. In the critical follow-up recommendation study, the medical expert annotators owned the categories and labels for critical follow-up recommendations. Based on the findings of Xia and Yetisgen-Yildiz (2012), annotation guidelines are an important support document for the annotation process. In the critical follow-up recommendation identification

study, reports were annotated in short bursts of activity over extended periods of time due to the time commitment constraints of the medical expert annotators. The guidelines assisted the annotators in maintaining consistency, especially given the subtle differences in the critical follow-up recommendation category labels.

2. **Train annotators** Annotators were trained using a combination of training materials developed to document the computing infrastructure required to host the annotation tool as well as training materials bundled with the annotation tool itself. All annotators initially worked one-on-one with an NLP researcher to familiarize themselves with the computing infrastructure, annotation tool, and annotation guidelines.

3. **Annotate corpus** Annotators annotated the corpus in the BRAT annotation environment. It could be accessed either through a virtual desktop or, when the corpus access was not constrained by privacy or legal reasons, on the annotator's local system.

4. **Measure IAA** To finalize guidelines and develop a consensus on how annotations should be applied, two or three annotators annotated a small sampling of the overall corpus in one or more rounds of double or triple annotation. An IAA measure was selected to determine the reliability of the annotations after each round. In this study, two IAA measures were used to compare annotators pairwise after each round: (1) a precision, recall, and F-score for the two annotation sets, one acting as system, the other as gold, and (2) Cohen's Kappa. After each round of annotation and IAA, NLP researchers reviewed the annotation schema and guidelines with annotators and made changes when required.

3.4.3 Corpus finalization

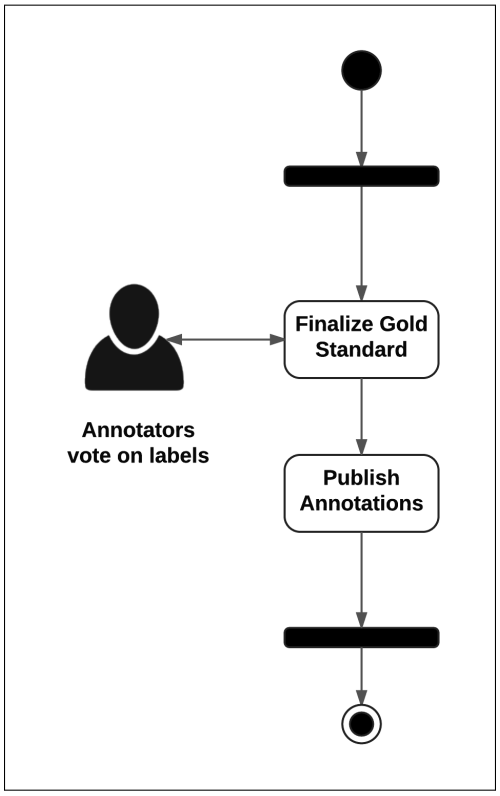

Figure 3.6: Corpus finalization workflow diagram

After the annotation guidelines are complete, annotators finish annotating the remaining unlabeled portion of the corpus. Typically, the work of annotating an entire corpus is divided up among multiple annotators. Each annotator is solely responsible for annotating an assigned section of the remaining unlabeled corpus. The final stage of annotation creates the gold standard or final version, which is used to train, test, and evaluate systems that generate or extract the same annotations.

The corpus finalization stage consists of two main tasks:

1. **Finalize gold standard** Finalizing the annotation guidelines does not guarantee that annotators will be consistent in their annotation tasks. Additional rounds of double or

triple annotation on randomly selected portions of the corpus can improve the quality of the gold standard. Similar to IAA comparison rounds that shape the annotation guidelines and schema, comparison of annotations when creating the gold standard can help clarify how to apply annotation consistently among annotators and update the annotation guidelines.

2. **Publish annotations** Clinical corpora require additional treatment before they are released. Privacy and legal considerations require vetting the release with the IRB and ensuring all clinical records are de-identified so that no patient information is revealed. I plan to release the pneumonia report classification unique rationale snippet corpus of 1008 text snippets annotated with change-of-state and diagnosis events to the Web at the conclusion of this study.

3.5 Event detection

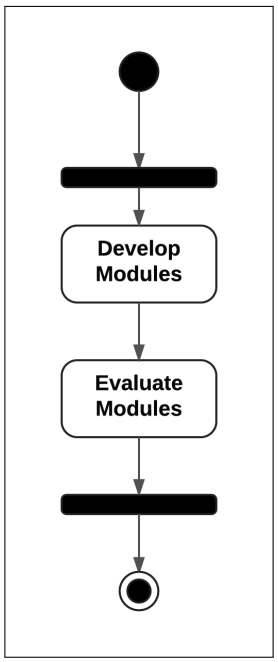

Figure 3.7: Event detection workflow diagram

Once an event model has been defined and translated into an annotation schema, it can be used to annotate a corpus, which is then used to train and test event detection modules using statistical NLP tools. The approach to event detection depends on the structure of the event. The pneumonia report classification and ALI studies both represented events as dependency trees, where as the critical follow-up recommendation identification study defined an event as a template of properties. Event detection was handled differently in each case.

The main tasks of creating an event detection system are to:

1. **Develop event detection modules** In the pneumonia report classification study, change-of-state and diagnosis events were represented as dependency trees. To extract a dependency tree from a text file, a two-stage process was used. The first stage identified change-of-state and diagnosis events entities in text using a named entity recognition

(NER) module, while the second stage used a graph-aware dependency parser to create graph structures over the named entities identified in the first stage.

In the critical follow-up recommendation study, critical follow-up recommendations were identified in a multi-stage process:

1. Sentences were identified as critical follow-up recommendation using a maximum entropy (MaxEnt) sentence classifier trained on the gold standard recommendation sentences.
2. A conditional random fields (CRF) NER module trained on `Reason`, `Time`, and `Test` annotations extracted named entities from the recommendation sentence.
3. A rule-based module generated the computed values *default recommended follow up test* and *default recommended follow timeframe*.
4. A MaxEnt classifier trained on the above properties plus report text features and metadata, assigned one of four critical recommendation labels to the event.

2. Evaluate modules All experiments designed to evaluate NER and relation extraction (RE) modules developed for both the pneumonia report classification and critical follow-up recommendation studies incorporated 5-fold cross-validation in order to train, test, and measure the performance of event detection components.

The performance of the pneumonia report classification and critical follow-up recommendation identification named entity recognizer was measured by precision, recall, and F-score based on the exact match of BIO labeled words in the system output to BIO labeled words from the gold standard. BIO labeling prefixes a word's label with either B if it is the beginning word in a named entity sequence, or I for a word inside a named entity sequence, and O for all words outside of a named entity labeled sequence. In both pneumonia report classification and critical follow-up recommendation studies, NER was evaluated at the word (token) and entity (BIO) level.

Pneumonia report classification dependency parsing results were evaluated using the

evaluation tool, MaltEval, which reports: (1) Labeled Attachment Score (LAS), calculating the accuracy of both a dependency label and its attachment in the dependency graph, (2) Unlabeled Attachment Score (UAS), calculating the accuracy of a dependency node and its attachment in the dependency tree without respect to its label, and, (3) Label Accuracy (LA), calculating the accuracy of all labels in the dependency graph including leaf-level entities. The critical follow-up recommendation identification category classification task was evaluated on exact match and reports precision, recall, and F-score.

3.6 Event-based feature extraction

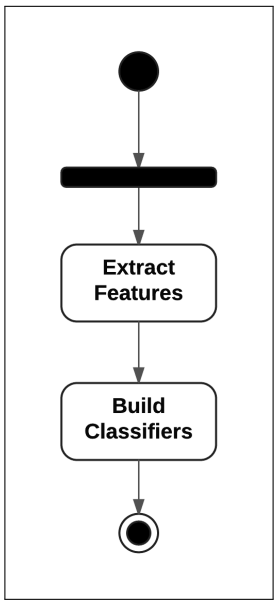

Figure 3.8: Event feature extraction workflow diagram

The event-based feature extraction phase has two tasks:

1. **Extract features** For pneumonia report classification and ALI experiments, event-based features were extracted from the entity, attribute, and relation annotations and their

values in the generated output from the event detection module. The remaining types of event-based features were created from combinations of tuple fields extracted from the tuple representation of the event, which was derived algorithmically from the event tree. For critical recommendation events, features were extracted from the critical follow-up recommendation sentence and report metadata. In all three applied tasks, event-based features were evaluated in combination with all other features in χ^2 feature selection. The best performing features, based on threshold values, were integrated into applied task experiments.

2. **Build classifiers** I used supervised statistical machine learning approaches for classification in all three applied tasks. Feature sets were generated with a common software library and the MALLET[5] (McCallum, 2002) machine learning tool kit was used for training and testing MaxEnt models. All features were represented in MALLET-style feature vectors. I used χ^2 statistical feature selection and threshold values to select the optimal number of best performing features for experiments. See Sections 4.4, 5.4, and 6.4 for detailed descriptions of my approach for each task.

3.7 Summary

In this chapter, I described the general framework of research methods I adapted to the three applied clinical NLP tasks in my dissertation. In Section 3.3, I described my approach for defining and analyzing events and in Section 3.4, I documented my approach to creating corpora in the clinical domain. In Section 3.5, I reviewed how I adapted information extraction techniques to the task of event identification and detection based on the the structure and representation of an event and in 3.6, I described how I extracted features from event representations and applied those features to NLP tasks. In Sections 4.3, 5.3, and 6.3 I detail how I adapt the general research framework described in this chapter to each individual applied task.

---

[5]http://mallet.cs.umass.edu/

Chapter 4

PNEUMONIA REPORT CLASSIFICATION

In my literature review, I discussed previous work targeting disease detection with NLP methods (Vanderwende et al., 2013; Yetisgen-Yildiz et al., 2013a; Tepper et al., 2013). A common goal of disease detection studies in clinical NLP is to integrate with larger institutional systems to provide realtime feedback and monitoring of patients' state for early warning of critical illness. Integrated surveillance systems for disease detection and prevention are an important component of long range planning for next generation healthcare platforms (Shortliffe and Blois, 2006). Given that narrative free-text reports represent the largest portion of data in the electronic health record (EHR), disease surveillance systems of the future will require applications that meaningfully process natural language in order to unlock all of the information about a patient's state. Albright et al. (2013) go further, suggesting the subtleties of natural language in clinical narratives require a comprehensive syntactic and semantic approach in order to capture difficult aspects of meaning, such as the agent or patient of a clinical event, the level of certainty about events, and references to an event before and after an event's occurrence in text. Vanderwende et al. (2013) propose an event-based disease detection system that, by monitoring a patient's state over time in the clinical narrative of the EHR, can prompt physicians with early warning and improve patient healthcare.

In this chapter, I implement a system for pneumonia report classification by using NLP and supervised learning techniques to extract event-based features from text and train a maximum entropy (MaxEnt) machine learning classifier. Section 4.1 describes the applied task I explore in this case study, classifying an X-ray report corpus labeled for suspicion

of pneumonia (PNA) and Cardio Pulmonary Infection Score (CPIS)[1] to support the larger goal of ventilator-associated pneumonia (VAP) phenotype detection. Section 4.2 details previous research, including annotated corpora and NLP systems developed for pneumonia report classification. In Section 4.3, I discuss how I adapt my general research framework to the specific goals of the case study and in Section 4.4, I outline how I implement system components and experiments to support my research into the applied task. In Section 4.5, I report the results of my component evaluations and pneumonia report classification experiments, and in Section 4.6, I discuss my results. I present my conclusions in the final section of the chapter, Section 4.7.

4.1 Task overview

VAP is a type of pneumonia that occurs in hospital to patients that are placed on mechanical ventilation due to critical illness or traumatic injury. The early detection and treatment of VAP is important as it is the most common healthcare-associated infection in critically ill patients and even short-term delays in appropriate treatment for patients with VAP are associated with high mortality rates, longer-term mechanical ventilation, and excessive hospital costs (Zilberberg and Shorr, 2010).

The presence of VAP in a patient cannot be established by a single test and diagnosis is often complicated by other co-occurring symptoms due to the critical illness or traumatic injury that is the primary cause of mechanical ventilation. Diagnosing VAP involves weighing many different types of evidence, such as chest X-rays, blood work, or respiratory secretions, at multiple points in time (Zilberberg and Shorr, 2010). Of the various types of evidence that contribute to a diagnosis, the narrative chest X-ray report is a core test for suspicion of VAP. It is used to monitor the change in health of patients over time and the progress of disease at a detailed level (See Figure 4.1 for an example of a chest X-ray). In addition to providing information about a patient's state at a particular point in time, clinical reports

[1]CPIS is used to predict which patients will benefit from the invasive, and preferably avoidable procedure to obtain pulmonary cultures.

make explicit or implicit references to previous reports and form chains of information about the state of the patient at many points of time. Capturing this level of detail, and connecting it to other information in a patients' EHR, is critical for the detection of diseases such as pneumonia.

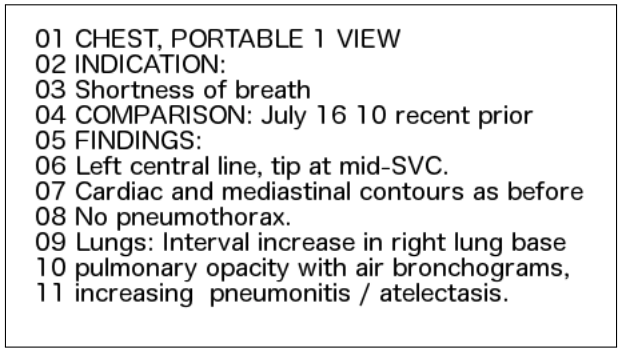

Figure 4.1: A sample chest X-ray report

In my case study, I implement a pneumonia report classification system that classifies X-ray reports for pneumonia using supervised machine learning methods and features extracted from change-of-state and diagnosis events. The long term goal of the system is to integrate with a larger VAP phenotype detection system as described in Xia and Yetisgen-Yildiz (2012) and Tepper et al. (2013).

4.2 Previous research

Xia and Yetisgen-Yildiz (2012) developed an annotated corpus of 1344 pulmonary chest X-ray reports from the University of Washington Harborview Medical Center to train and test a pneumonia report classification system as a component of an overall VAP phenotype detection system. To develop the corpus, medical expert annotators labeled pulmonary X-ray reports with two categories of labels: (1) PNA and (2) CPIS. See Table 4.1 for the PNA and CPIS category labels.

VAP Category	VAP Category Label
CPIS	1A (no infiltrate)
	1B (diffuse infiltrate or atelectasis)
	1C (local infiltrate)
PNA	2A (no suspicion of PNA)
	2B (suspicion of PNA)
	2C (probable PNA)

Table 4.1: VAP category labels for PNA and CPIS

4.2.1 Rationale snippets

Tepper et al. (2013) extended the work of Xia and Yetisgen-Yildiz (2012) by asking the same medical expert annotators who participated in the original labeling task to highlight one or more text spans in the X-ray reports that provided concise rationale for the report's PNA and CPIS labels. They called these text spans *rationale snippets*. In similar work, Zaidan et al. (2007) and Zaidan and Eisner (2008) used annotator rationale to improve performance in classification tasks for the general domain, and Yu et al. (2011) in the clinical domain. For Tepper et al. (2013), rationale snippets indicate passages within narrative clinical reports that are relevant for pneumonia report classification.

The system in Tepper et al. (2013) performs two related and cascaded tasks for classifying X-ray reports, (1) rationale snippet prediction, and (2) pneumonia report classification. In their study, they observed that annotations for rationale snippets align with sentence boundaries at a rate of over 95%, and because of this, they modeled the first task in their cascaded approach, rationale snippet prediction, as a two-stage sentence classification and sequence labeling task. They trained a sentence classifier and sequence labeler to predict the locations of CPIS and PNA rationale snippets in X-ray reports using an *inside, outside* tag-set for annotation, a MaxEnt model for classifying candidate rationale sentences, and a beam search for *inside, outside* sequence labeling of the candidate sentences. See Section 4.5.1 for

their results.

For the second task in their cascaded report classification system, Tepper et al. (2013) trained and evaluated, using 5-fold cross-validation, two SVM classifiers, one for CPIS and the other for PNA. Multiple sets of classification experiments were run with features extracted from four different scenarios:

1. the whole document
2. gold standard annotated oracle rationale snippets
3. system-predicted rationale snippets
4. combinations of the whole document with an oracle or predicted rationale snippet

The same feature sets were used and compared for all experiments:

- unigrams and bigrams filtered for stop words, digits, and punctuation
- UMLS concepts
- alternate proposing conjunctions (*versus*, *or*, *vs*, etc.)

Tepper et al. (2013) reported that extracting features from rationale snippets alone performed better than extracting features from the whole document or the whole document in combination with predicted snippets, and that word unigrams, filtered for stop words, digits, and punctuation, in combination with alternate proposing conjunctions were the best performing types of features to extract. See Section 4.5.1 for results.

Error analysis in Tepper et al. (2013) revealed the limitations of using only basic features in the pneumonia report classification experiments. A good example can be seen in the X-ray corpus rationale snippet, *The previously noted right upper lobe opacity consistent with right upper lobe collapse has resolved.* The system made an error and mislabeled this snippet because the snippet supports one category entirely up to but not including the crucial words *has resolved*. The classification system's basic word unigram feature set only provides words and their frequencies to the classifier. It does not capture the importance of the words

has resolved to the meaning of the sentence, and their long distance relationship to the observation, *right upper lobe opacity.*

Another example of a similar type of error can be seen in the X-ray corpus rationale snippet in Figure 4.2.

No change in diffuse lung disease consistent with edema.
Patchy bilateral lung consolidation has diminished.
No new focal abnormalities

Figure 4.2: A sample chest X-ray report snippet

The rationale snippet belongs to a chest X-ray report labeled in the gold standard as *2B (suspicion of PNA)*. The pneumonia report classification system mislabeled this report as *2A (no suspicion of PNA)*, because the system uses a simple n-gram approach, which in this case puts more emphasis on the occurrence of negating word unigrams, such as *no* and *diminished*, rather than the more meaningful complete statement *No change in diffuse lung disease*. As stated in Section 1.1.2, these types of errors motivated the change-of-state event proposal in Vanderwende et al. (2013).

4.2.2 Change-of-state events

Vanderwende et al. (2013) reviewed the results of experiments and error analysis in Tepper et al. (2013) and concluded that the majority of text in rationale snippets described observations of change in a patient's state (for example, *minimal patchy atelectasis in the right lung is seen, mildly improved since the prior study*) which they defined as a change-of-state event. In order to capture and label a change-of-state event in text, they proposed an n-tuple annotation model, based on the bio-event tuple described in Kim et al. (2008).

The n-tuple consisted of five fields, `Loc`, `Attr`, `Val`, `Cos`, and `Ref`. Figure 4.3 is an example of a change-of-state n-tuple with values for all five fields: `Loc` is the anatomical location (e.g.,

right lung), `Attr` is an attribute of the location that the event is about (e.g., atelectasis), `Val` is a possible value for the attribute (e.g., minimal patchy), `Cos` indicates the change of state for the attribute value compared to some previous reports (e.g., mildly improved), and `Ref` is a link to the report(s) that the change of state is compared to (e.g., since the prior study) (Vanderwende et al., 2013). Not all tuples have values for all five fields. A field can be unspecified and inferred from the context of the surrounding snippet text or from the collection of snippets that have been extracted from the sequence of a patient's X-ray reports.

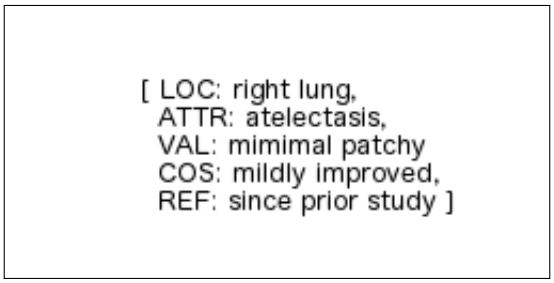

Figure 4.3: An example of an event tuple

In Figure 4.4 the fields of the event tuple are represented as labeled entities in the annotation of the snippet text. Annotators labeled unique snippets using the Web-based BRAT[2] (Stenetorp et al., 2012) annotation tool.

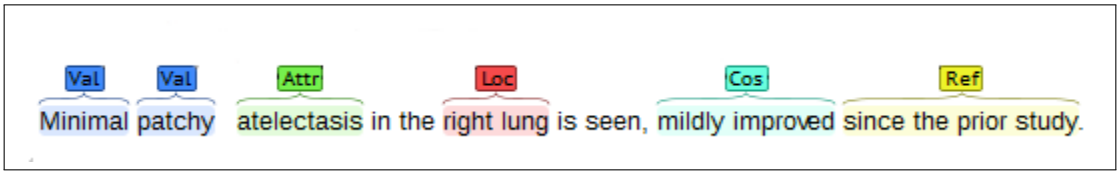

Figure 4.4: A snippet featuring the annotation of change-of-state event n-tuple fields as entities in BRAT

[2]http://brat.nlplab.org/index.html

4.3 Methodoloy

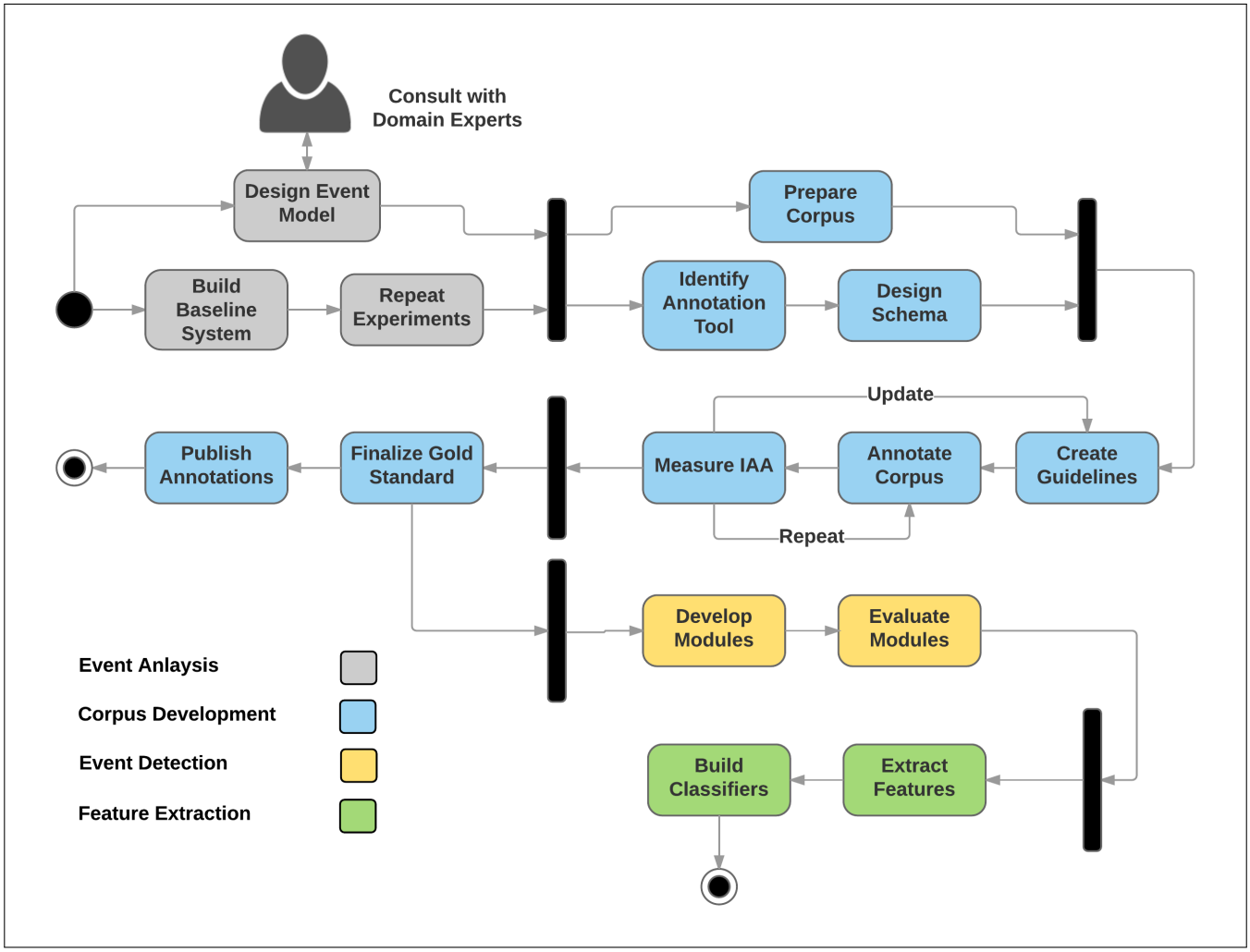

Figure 4.5: Pneumonia report classification methodology

I approached the pneumonia report classification; task by adapting the four phases of my proposed research framework, described in Chapter 3. The four phases are:

1) **Event analysis**—analyze the task and related previous research to define events
2) **Corpus development**—develop corpora by annotating events
3) **Event detection**—build systems for automatic event detection
4) **Event-based feature extraction**—extract event-based features for classification

Figure 4.5 provides a view of the four phases of the framework adapted for the study of pneumonia report classification.

4.3.1 Event analysis

During the initial phase of research, event analysis, I defined change-of-state and diagnosis events. Two of the four main tasks of event analysis adapted from the general framework are applicable to the change-of-state and diagnosis event study:

1. analyze results of previous research or replicate system and repeat experiments
2. define a preliminary model for events

In this study, I replicated the two-stage cascaded snippet prediction and pneumonia report classification system of Tepper et al. (2013) in order to establish a baseline set of n-gram experiments. The baseline experimental results and error reports informed my extensions to the original change-of-state n-tuple proposed in Vanderwende et al. (2013). I introduced a preliminary directed acyclic graph (DAG) model for the change-of-state event in order to connect labeled entities in an overarching event structure that can link descriptions of change of state with observations that precede or follow it in a sentence. The new event structure addressed the limitations of the n-gram feature model described in Tepper et al. (2013) and Vanderwende et al. (2013) and its graph structure simplified event annotation and extraction.

4.3.2 Corpus development

I adapted the list of corpus development tasks to the needs and requirements of this study. My adapted corpus development process consists of the following steps:

Corpus preparation In the corpus preparation stage of the corpus development phase, both the X-ray corpus annotated for PNA and CPIS labels and the unique rationale snippet corpus were imported into the system database and converted into table columns and rows based on a relational model. Additional tasks during this stage included:

1. identifying and selecting an annotation tool
2. developing a new schema to support the change-of-state and diagnosis event models

Corpus annotation (multiple rounds) The annotation stage of the corpus development phase occurred over two rounds. In the first round, three NLP researchers annotated the unique rationale snippet corpus with BRAT entities and relations based on an initial design of the change-of-state event model. A `Diagnosis` entity was also included in the annotation scheme to label phrases in the snippet that propose or suggest a diagnosis. The second round of annotation used an extended model which added coordination and negation entities to the change-of-state events and decomposed the previously annotated `Diagnosis` text span into the entities and labeled arcs of a diagnosis event model. After each round, inter-annotator agreement (IAA) measures were calculated and the annotators met to discuss differences in annotation and update guidelines. A single NLP researcher added all second stage annotations to the unique rationale snippet corpus.

The following steps were adapted from the canonical methodology discussed in Chapter 3:

1. create an initial set of annotation guidelines and a small dictionary of special terms to guide annotators
2. train annotators and update annotation guidelines based on their feedback
3. annotate a small sample of rationale snippets in triplicate over two rounds
4. compare annotations using inter-annotator agreement measures
5. review results of IAA with annotators and finalize guidelines
6. apply the final annotation guidelines to the entire corpus of unique rationale snippets

Corpus finalization In Vanderwende et al. (2013) a change-of-state n-tuple represented a single event. In my study, a change-of-state and diagnosis event structure includes one or more events. In order to generate individual event features it is necessary to extract n-tuples from the change-of-state and diagnosis event tree. I created an algorithm to extract one or more n-tuples from the change-of-state and diagnosis event dependency tree representation created during the event annotation stage and used the same algorithm to extract tuple-based features for pneumonia report classification experiments.

In the final step of corpus development, I released the gold standard version of the unique snippet rationale corpus and accompanying change-of-state and diagnosis event annotation guidelines to the Web at http://depts.washington.edu/bionlp/datasets.htm. In addition to the annotated corpus and guidelines, I also bundled the snippet prediction and change-of-state and diagnosis event extraction modules as standalone NLP tools.

The two steps of the general research framework that I applied to the pneumonia report classification study were:

1. develop an algorithm to derive event tuples from the change-of-state and diagnosis event annotations
2. release annotated unique rationale snippet corpus and BRAT schemas to the Web

4.3.3 Event detection

In the event detection phase, I followed the general framework by designing and evaluating modules for a two-stage event detection process, the first stage consisting of a named entity recognition (NER) module, and the second, a relation extraction (RE) module based on dependency parsing. For evaluation, I conducted 5-fold cross-validation experiments using the change-of-state and diagnosis event annotations developed for the unique snippet rationale corpus in the corpus development phase.

4.3.4 Event-based feature extraction

In the final phase of my research framework, I extracted event-based features and integrated those features into pneumonia report classification experiments. I defined six types of event-based features and created modules to extract and integrate the event-based features into the replicated baseline pneumonia report classification system.

4.4 Implementation

The implementation section details the components developed during the four phases of research. Table 4.2 describes the individual components I implemented and the evaluation metrics I used to measure their performance. Figure 4.6 provides a system diagram of all of the components and their interaction in the final application implemented for the pneumonia report classification task.

Research Phase	Impl. Step	Description	Evaluation
Event Analysis	Baseline system	Replicate the two-task cascaded snippet prediction and pneumonia report classification system as described in Tepper et al. (2013)	Snippet sentence matching performance measure and replication of baseline pneumonia report classification results from original study
	Event models	Model the change-of-state event, introduce a diagnosis event and entities for negation and coordination	Lossless transformation between dependency tree and n-tuple representations
Corpus annotation	Annotation	Annotate the unique rationale snippet corpus with change-of-state and diagnosis events, including negation and coordination entities	IAA measures including by token, entity, and event tuple
Event Detection	Named entity recognition (NER) module	Train and test an NER module to identify and label named entities in the unique rationale snippet corpus	P,R, and F1 measures of NER labeling by entity and token
	Relation extraction (RE) dependency parser module	Train and test RE module based on dependency parsing	CoNLL Shared Task on Dependency Parsing measures (LA, UAS, LAS). Merge algorithm for aligning entity-only system-generated NER/dependency parsing results with gold standard.
Event Extraction	N-Tuple generation algorithm	Algorithm for lossless transformation of dependency tree representations to n-tuple representations of change-of-state and diagnosis events	Selectively test and manually evaluate against n-tuple gold standard
	Feature extraction module	Implement methods for extracting event-based features	Selectively test and manually evaluate against ad-hoc gold standard
	Pneumonia report classification system	Run CPIS and PNA pneumonia report classification experiments using baseline features, event-only features, and all features, with and without χ^2 feature selection	P, R, and F1 measured and evaluated for whole document, oracle snippet, and predicted snippet. Compared against baseline results generated using system replicated from descriptions in Tepper et al. (2013).

Table 4.2: A breakdown of the overall task into steps and evaluation metrics

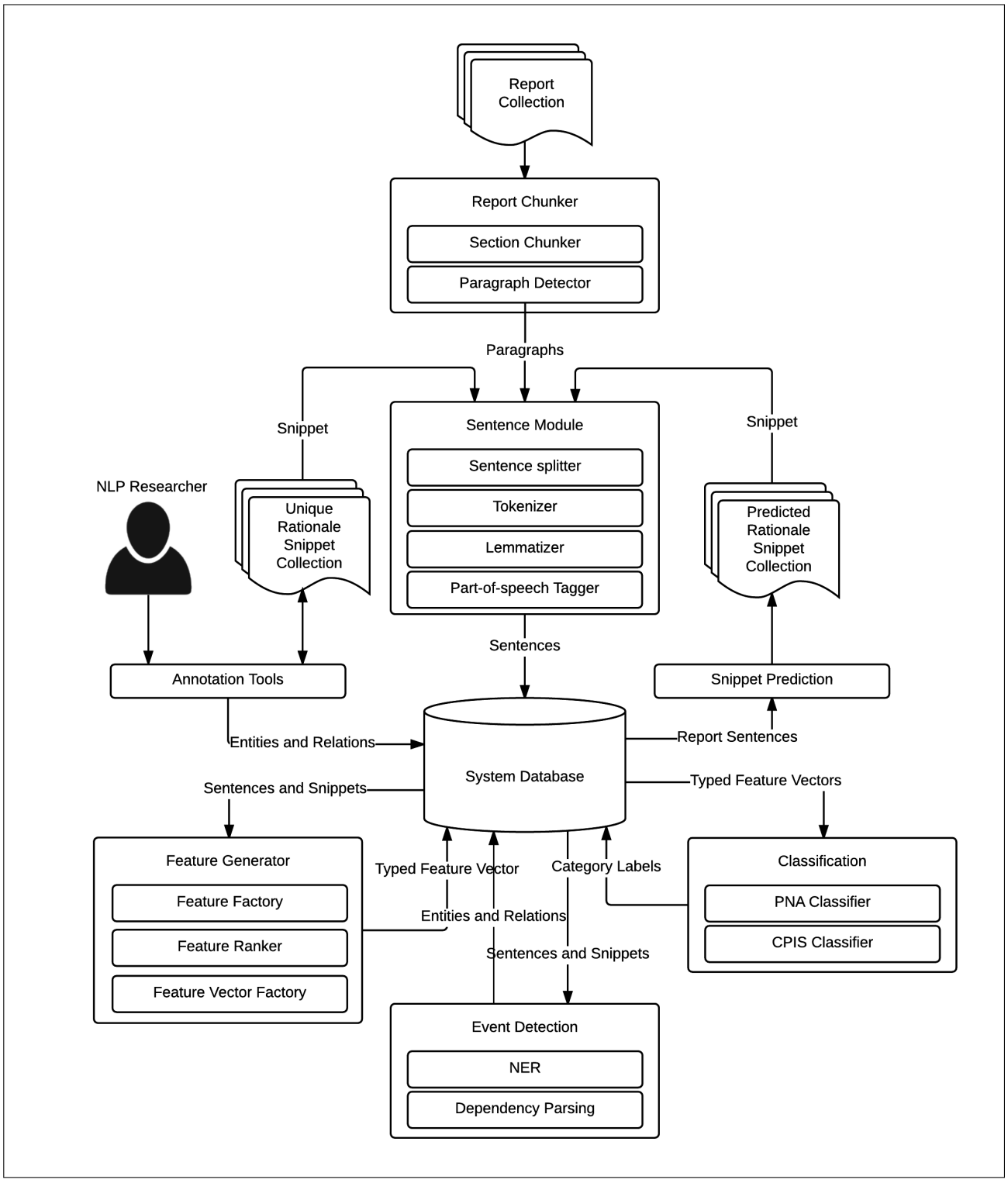

Figure 4.6: Pneumonia report classification system diagram

4.4.1 Baseline system

In order to replicate the pneumonia report classification experiments described in Tepper et al. (2013), I recreated the classification system and snippet prediction module and imported the X-ray report corpus, annotated with both PNA and CPIS labels, and the unique rationale snippet corpus, into my system database environment.

The classification experiments were conducted in three scenarios:

1. whole document (the text of an entire report)
2. a report's oracle rationale snippets only
3. a report's predicted rationale snippets only (generated by the snippet prediction module)

Whole document boundaries are the complete text of one report in the overall X-ray corpus, the oracle rationale snippet boundaries are the complete text of all of the unique rationale snippets that can be traced back to one report, and the predicted rationale snippet boundaries are the complete text of one or more predicted snippets generated by the snippet prediction module for one report.

To obtain (1) and (2) described in the preceding paragraph, I imported existing versions of the pneumonia report classification X-ray corpus and unique rationale snippet corpus into my system database. There are 1344 X-ray reports and 1065 snippets in the X-ray report corpus. As noted in the original study, due to annotator error, CPIS and PNA labeling was not applied to every report in the corpus. There are 3 reports where no CPIS label has been applied and 1 report where no PNA label has been applied, resulting in a total set of 1341 and 1343 reports for CPIS and PNA, respectively (Tepper et al., 2013). Table 4.3 details the distribution of labels across the X-ray corpus.

Tepper et al. (2013) described a sub-corpus of 1065[3] unique rationale snippets extracted from the chest X-ray corpus as a de-duplicated collection of all of the embedded rationale snippets throughout the X-ray corpus, irrespective of CPIS and PNA labels. By normalizing spaces and line endings in snippets, I further reduced the unique snippet sub-corpus to 1008 snippets. Each unique snippet is associated with one or more traces back to the original reports. Most of the report files contain two snippets, one for PNA and one for CPIS, which share the same text. Other reports feature only one PNA snippet or one CPIS snippet with different text, and a small number of reports have no snippet at all. It is possible for a report to have more than one rationale snippet for both PNA and CPIS, but it is rare in the overall corpus.

Label	**Category**	**Number (%) of reports**	**Number (%) of reports with rationale snippets**
CPIS	1A no infiltrate	178 (13%)	25 (2%)
	1B diffuse infiltrate...	1011 (75%)	1004 (75%)
	1C local infiltrate	152 (11%)	152 (11%)
	Total	1341 (100%)	1181 (88%)
PNA	2A no suspicion PNA	856 (64%)	362 (27%)
	2B suspicion of PNA	294 (22%)	290 (22%)
	2C probable PNA	193 (14%)	192 (14%)
	Total	1343 (100%)	844 (63%)

Table 4.3: Statistics of the X-ray report corpus as reported in Tepper et al. (2013)

[3]I further reduced the number of unique snippets to 1008 by eliminating line breaks and extra whitespace in the original snippet text.

In order to replicate all of the baseline experiments in Tepper et al. (2013), I created a new snippet prediction module based on the description in the original study. I trained and tested the snippet prediction module and used χ^2 to select the top 250 word unigram and alternate conjunction features for experiments. This set of feature types I called the *baseline set*. See Table 4.15 for the results of the snippet prediction experiments. Table 4.4 lists the overall total reports and unique snippets in the X-ray corpus and Table 4.4 summarizes the statistics for reports by label.

X-Ray corpus reports	1344
Unique rationale snippet corpus snippets	1008

Table 4.4: X-ray corpus and unique rationale snippet corpus totals

	CPIS	PNA
Reports with valid categories	1341	1343
Reports with rationale snippets	1181	844

Table 4.5: X-ray corpus rationale snippet statistics by label

Once the snippet prediction module was completed and the unique rationale snippet annotations were merged with the 5-folds of X-ray reports for both CPIS and PNA, I replicated the baseline experiments described in Tepper et al. (2013). I used the machine learning toolkit MALLET[4] (McCallum, 2002) instead of the LIBSVM Java API[5] (Chang and Lin, 2011) used in the original study. I chose to use maximum entropy (MaxEnt) in the MALLET package because I preferred the transparency of MALLET's models for feature engineering and MaxEnt has comparable performance to LIBSVM's support vector machine (SVM) algorithm for multiple label classification tasks. See Table 4.20 for a comparison of results

[4]http://mallet.cs.umass.edu/

[5]http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

published in Tepper et al. (2013) and the results from experiments run from my replicated baseline system.

4.4.2 Event models

Once the baseline feature set experiments were replicated, I extended the n-tuple change-of-state event model proposed in (Vanderwende et al., 2013). I developed graph-based models for change-of-state and diagnosis events over two stages of analysis and multiple rounds of annotation. To model a change-of-state event in the first stage of analysis, I defined it as a tuple of five fields or slots: ⟨`Cos, Attr, Loc, Val, Ref`⟩. See Section 4.2 for an example of a tuple and how each field is mapped to its value in Figure 4.3.

As a sentence can contain multiple events, I grouped the fields of an event together in a cohesive, separate structure. I used directed, labeled arcs to link two fields. There are four arc labels:

1. Label *State* connects a `Cos` entity with an `Attr` or a `Loc` entity
2. Label *Location* connects an `Attr` entity with a `Loc` entity
3. Label *Value* connects an `Attr` or `Loc` entity with a `Val` entity
4. Label *ReferencedBy* connects a `Cos` or an `Attr` entity with a `Ref` entity

I defined a total order between fields, `Cos` ≺ `Attr`≺ `Loc` ≺ `Val` ≺ `Ref` and required that, in an arc $A \rightarrow B$, A must precede B according to the total order. As a result, the annotation of an change-of-state event is a directed acyclic graph (DAG), as shown in Figure 4.7.

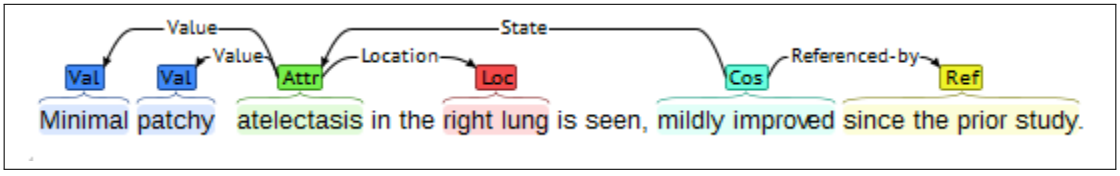

Figure 4.7: A snippet featuring the annotation of a change-of-state event

I also defined a phrase level entity `Diagnosis` which labels statements in the snippet that remark on change-of-state events. `Diagnosis` is a separate entity and not part of the change-of-state DAG.

During the first stage of annotation, three annotators applied the initial change-of-state and diagnosis entity model in two rounds of annotation. Their annotations were compared with IAA measures and differences between annotators were resolved. The feedback from these comparison rounds was incorporated into the annotation guidelines. A single NLP researcher completed the annotation of the entire unique rationale snippet corpus based on the guidelines and the first version of the annotated change-of-state pneumonia report classification corpus was used to test and train prototype event detection modules.

4.4.2.1 Extending the annotation schema

In the second stage of analysis, I extended the model by expanding the `Diagnosis` entity into an event DAG similar to change-of-state. In addition, I added coordination and negation entities which, when applied to the annotations from the first stage of analysis, transformed the original DAG structure into a dependency tree.

The diagnosis event

Rationale text snippets can include statements (e.g., *likely right pleural effusion*), which indicate a possible or likely diagnosis of a previously mentioned change-of-state. Because such a statement has the potential to be an important cue for disease detection, I extended my event analysis by expanding the definition of the previously defined entity `Diagnosis` to mark it with a new event type called *diagnosis*. The diagnosis event is very similar to a change-of-state event except that the `Cos` field is replaced by a new field called `Dhead` which is the head of a diagnosis statement (e.g., *likely* in *likely right pleural effusion*). The `Dhead` can contain words of possibility, conditional terms, modal verbs, and hedging language, such as, *likely*, *possibly*, *could*, and *might*. In addition to `Dhead`, the tuple for a diagnosis event may include the four fields (i.e., `Attr`, `Loc`, `Val`, and `Ref`); an example is given in Figure 4.8.

I also created a new arc label, *Diag*, that connects `Dhead` with `Attr`.

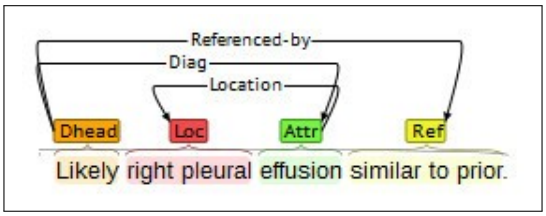

Figure 4.8: Annotation of a diagnosis event

Coordination and negation entities

Coordination and negation are common in snippets. Adding features in the schema for negation and coordination improved the representation in three ways:

1. Connecting entities together in a coordinated structure disambiguates how properties and values are distributed.
2. The single-headed constraint of a dependency tree can be realized when multiple connected entities are treated as a single entity.
3. Negation can be explicitly placed within the tree to address its intended scope over connected entities.

For example, in Figure 4.9, the three `Attr` fields are connected by conjunction words *or* and *and*; the `Cos` field *change* is negated by the word *No*.

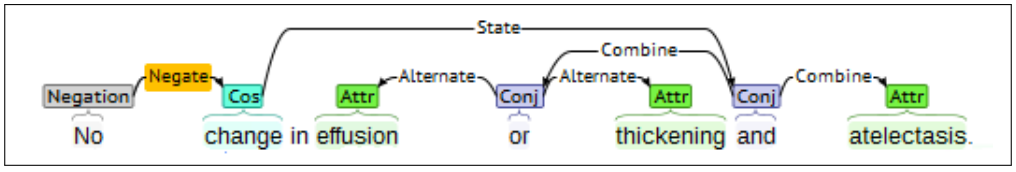

Figure 4.9: An example that includes both coordination and negation

In order to add coordination and negation to the change-of-state and diagnosis event model, additional labeled text spans and labeled arcs, or *entities* and *relations* as defined in the BRAT schema language, were added to the schema. To annotate coordination, I marked conjunctions such as *and*, *or*, and *but* as `Conj` and words such as *versus* and *vs.* as `Versus`. I distinguished five types of coordination, as illustrated in Figure 4.10, and used arc labels such as *combine* and *exclude* to indicate different relations between the conjuncts. Distinguishing the type of coordination could be important for disease detection; for instance, coordination with multiple *Alternate* arcs could indicate hedging in a diagnosis event. Not all of the labeled `Conj` entities in the corpus are strictly English syntax conjuncts or coordinators, for example the preposition *without* in the last example in Figure 4.10. I have allowed other parts of speech, such as the prepositions *with* and *without* to be labeled as `Conj` when the intent of the report author is to describe a semantic inclusion or exclusion relationship between two or more entities.

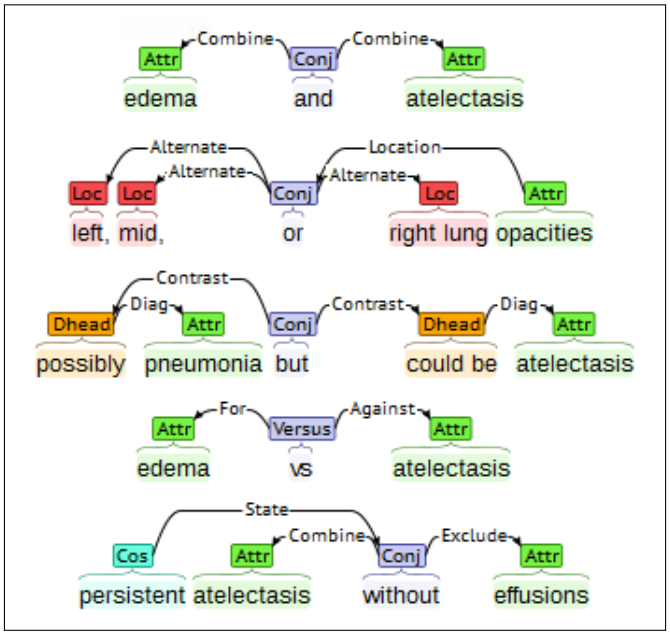

Figure 4.10: Five types of Coordination

I used the arc label *Negate* to link a negation word (`Negation`) with the word it negates. See Section 4.4.5 for an explanation of how the addition of coordination and negation entities to the model helped transform the representation from a DAG to a dependency tree.

The new annotation schema marks nine entity types: six as fields in a change-of-state or a diagnosis event (`Cos`, `Attr`, `Loc`, `Val`, `Ref`, and `Dhead`), two for coordination (`Conj` and `Versus`), and one for negation (`Negation` for negation words). The arc label set has 12 members: five for arcs connecting two event fields (*State*, *Location*, *Value*, *ReferencedBy*, and *Diag*), six for arcs connecting a conjunction and a conjunct (*Combine*, *Alternate*, *Contrast*, *Exclude*, *For*, and *Against*), and one for arcs connecting a negation word and a field (*Negate*).

The model also includes binary attributes[6] that can be applied to the head entities `Cos` and `Dhead` to generalize the specific text value of the entity. For example, there are 8 attributes that can be used to indicate a more general change-of-state, such as *Increased*, *Decreased*, *Improved*, *Worsened*, *New*, *Stable*, *Persistent*, and *Changed*. A *Hedge* attribute can be applied to the `Dhead` entity to indicate that it uses hedging language in its description of a diagnosis. There are no actual constraints in the annotation schema to limit the number of attributes that can be applied to an entity, however, some of the attributes are logically mutually exclusive, such as *Increased* versus *Decreased* and are not expected to be applied to the same entity. For a complete description of the model and its translation into a event tree schema for the BRAT annotation environment, please see Appendix A and the corpus development discussion in the next section.

4.4.3 Annotation

The original report annotations described in Tepper et al. (2013) were created using the GATE[7] (Cunningham et al., 2013) annotation environment and exported as XML documents based on the default GATE schema for annotation. The original rationale snippet

[6]These attributes are BRAT name and value pairs that can be applied in the annotation tool, they are not related to the semantic entity label, `Attr` in the actual change-of-state model.

[7]https://gate.ac.uk/

annotations were XML element text spans embedded as data islands within the report XML document. The change-of-state event and diagnosis annotations were represented as a DAG in the first round of annotation and as a dependency tree in the second. Although GATE enables hierarchical XML annotation, it does so in a limited and indirect way, requiring an ontology-based schema (OWL) for hierarchical relations between annotation elements. The tool I selected for the annotation task, BRAT, allows direct hierarchical relations to be drawn between entities and does so in a simple way, see Appendix B for BRAT schemas used in the pneumonia report classification study. I chose the BRAT tool over continuing to annotate in GATE for this specific reason. Additional contributing factors were its ease of deployment, simple web-based user interface, user documentation and training, and bundled examples.

BRAT uses a very concise and straightforward format: source documents are plain text files that are associated with a annotation file with the same name but different suffix (`.ann`). The annotation file uses an unique id, label, and character offsets to indicate the id, name, and start and end of text span entities in the original plain text source file. Relations, or labeled arcs, are listed as a label associated with an Arg1 and Arg2, which point to the unique ids of the associated entities.

A BRAT schema uses a limited syntax to describe the meta-rules for how its four essential elements should interact in the annotation environment. The four elements in the BRAT schema are:

1) entity a labeled text span
2) relation a labeled arc between two entities
3) attribute a name/value pair which can be applied to an entity[8]
4) note an extensible mechanism for inserting arbitrary notes and metadata into annotations

[8]The BRAT attribute is a type of schema element/rule and does not refer to the `Attr` entity in the change-of-state and diagnosis event schema.

The possible labels for these elements and the named relations between them are described in the BRAT schema file `annotation.conf`, which resides in a local or parent directory of the associated BRAT plain text and annotation files. I converted the logical model described in the event analysis phase and an initial version of the annotation guidelines into a BRAT schema for the change-of-state and diagnosis event models. See Appendix A for a complete description of the change-of-state and diagnosis model and Appendix B for versions of the BRAT schema developed during the course of the study.

To convert GATE-based XML documents to BRAT-friendly documents, I created a Java library of conversion utilities. Included in the Java library are BRAT to GATE readers and writers, BRAT source and annotation file readers and writers, an evaluation library for IAA based on the BRAT format, and sorting and filtering methods to extract custom subsets of BRAT annotations based on annotation properties, types, names, and custom search strings.

4.4.3.1 Annotation guidelines

The annotation guidelines provided annotators with instructions on how to install BRAT and some sample annotations to learn how to manipulate the user interface (e.g., adding, deleting, and editing annotations). In the first round of annotation, to use BRAT to annotate a change-of-state tuple, annotators marked five types of entities, corresponding to the five fields in an event tuple, `Cos`, `Attr`, `Val`, `Loc`, and `Ref`. An additional entity, not linked to the change-of-state tuple, called `Diagnosis`, labels any text span outside of an event tuple that contains a diagnostic phrase. See Figure 4.11 for an example of an annotated rationale snippet featuring an event tuple with all five fields.

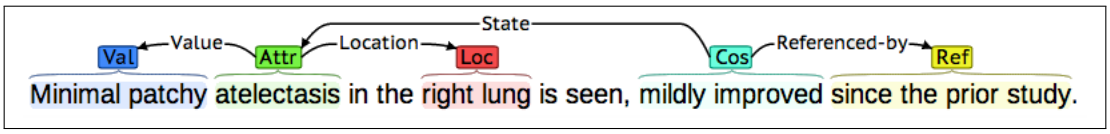

Figure 4.11: A rationale snippet featuring a change of state event annotation connecting all five fields of the change-of-state tuple

In the change-of-state model, some fields are shared by multiple event tuples, which in the graph-based model, are represented by directed, labeled arcs linking multiple entities. Once the entities in a snippet are connected to one another by labeled arcs, one or more connected, directed graphs are formed by these arcs. From the graph, event tuples can be derived algorithmically. Figure 4.12 shows the event tuple generated from the graph previously illustrated in Figure 4.11.

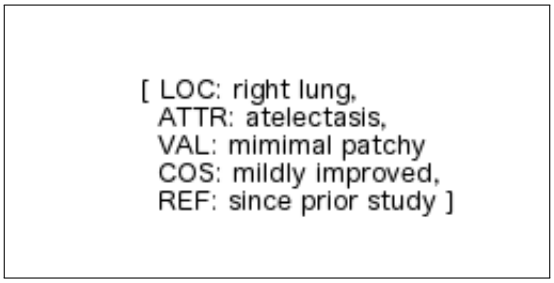

Figure 4.12: A derived event tuple.

A change-of-state and diagnosis event n-tuple represents a single event. The change-of-state and diagnosis event graph-based model can contain one-or-more event n-tuple events in a single connected dependency tree. In order to extract features for individual change-of-state and diagnosis events, one or more n-tuple representations are algorithmically extracted from the change-of-state and diagnosis event dependency tree.

Note that in the first stage of annotation, events can share entities between them and entities can have multiple parent entities. The graph in Figure 4.13 features two child `Attr` entities connected to a single, parent `Cos` entity. An event tuple is generated from the graph for each of the `Attr` entities and both tuples feature the same shared `Cos` entity.

In the second stage of annotation, the addition of new entities for the diagnosis event, negation, and coordination transformed the original DAG structure of the model into a dependency tree. The more constrained dependency tree structure of the revised change-of-state and diagnosis event model enables it to be parsed by a dependency parser. See

Section 4.4.5 for an explanation of the change-of-state and diagnosis event as dependency tree analysis.

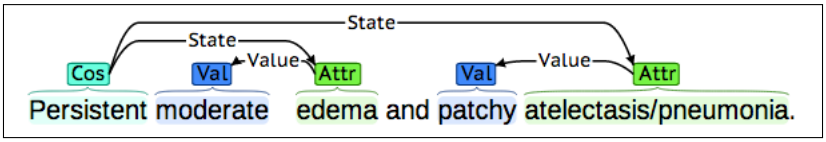

Figure 4.13: A snippet featuring shared entities between events.

A single NLP researcher adapted the annotation guidelines for the extended schema, providing additional documentation, explanations for new entities and labeled arcs, as well as screen captures from the annotation environment to provide concrete examples of all new features, including negation and coordination. The same researcher performed the final editing pass on the unique snippet corpus, ensuring all snippets conformed to the new change-of-state and diagnosis model described in the annotation guidelines.

The final versions of the annotation guidelines and BRAT schemas for change-of-state and diagnosis events represent the final form of the model. The two stages of annotation completed for this study would be unnecessary if a new corpus was targeted for annotation with change-of-state and diagnosis events.

4.4.3.2 Inter-annotator annotation process

In order to train annotators, calculate IAA, and finalize the annotation guidelines, 100 snippets were selected at random from the sub-corpus of 1008 unique text snippets. The snippets were annotated in two passes by three annotators. In the first pass, the annotators read the annotation guidelines, learned to use the annotation tools, and then annotated the first 20 snippets. IAA was measured at the word, entity, and event tuple level. The annotators then met and compared annotations, and the feedback from this discussion was used to revise and finalize the annotation guidelines. In the second pass, the annotators revised their annotation of the first twenty snippets following the revised guidelines, and completed annotating

the remaining 80 snippets. Again, IAA was calculated and the annotators reconvened to review their annotations and finalize the annotation guidelines. See Section 4.4.3.4 for the results of the two rounds of IAA measurement and a discussion of the results.

Once the initial guidelines were finalized in the first round of the annotation process, and after the IAA process was complete, a single annotator applied the new guidelines to the entire corpus. See Table 4.6 for corpus statistics of the entities before and after schema extensions, Table 4.7 for corpus statistics of labeled arcs before and after schema extensions, and Table 4.8 for attribute corpus statistics.

Entity	Annotation Stage 1 (Before Extensions)	Annotation Stage 2 (After Extensions)
Cos	1020	1067
Attr	1633	2404
Val	855	937
Loc	1657	1996
Ref	181	179
Dhead	0	517
Diagnosis	439	0
Conj	0	671
Versus	0	39
Negation	0	253
Total	5785	8063

Table 4.6: Corpus change-of-state and diagnosis event entity statistics before extensions (annotation stage one with six entity types) and after extensions (annotation stage two with nine entity types)

Labeled Arc	Annotation Stage 1 (Before Extensions)	Annotation Stage 2 (After Extensions)
State	1203	1074
Value	852	996
Location	1648	1829
Referenced-by	253	166
Diag	0	526
For	0	39
Alternate	0	430
Against	0	39
Combine	0	986
Contrast	0	17
Exclude	0	11
Negate	0	253
Total	3956	6366

Table 4.7: Corpus change-of-state and diagnosis event labeled arc statistics before extensions (annotation stage one with four labeled arc types) and after extensions (annotation stage two with twelve labeled arc types)

Attribute	Count
Changed	567
Decreased	37
Hedge	198
Improved	78
Increased	118
New	98
Persistent	231
Slash_Delimited	90
Stable	56
Worsened	65
Total	1538

Table 4.8: Corpus change-of-state and diagnosis event head-attribute (`Cos` and `Dhead`) statistics after extensions (annotation stage two with ten attribute types)

4.4.3.3 IAA agreement measures

Three agreement measures were used in the IAA process:

At word level To compare annotation at the word level, a standard BIO scheme to obtain word-level labels from entity annotations was used: If a text span is labeled as an entity of type *X*, the word-level label of the first word in the span is *B-X*, the label of other words in the span is *I-X*, and words that do not appear in any entity are labeled *O*. With the word-level labels, precision, recall, and F1-scores of *B-X* and *I-X* labels are calculated in a pairwise comparison of three annotators for both the first 20 and final 100 snippets. The results for the first 20 and the final 100 are listed in Table 4.9.

At entity level To compare annotations at the entity level, an exact match of entities is defined—two entities match exactly if their text spans are exactly the same and their entity types are identical. Table 4.9 show precision, recall, and F1-score in a pairwise comparison of three annotators for both the first 20 and final 100 snippets.

At event level From the labeled graph (See Figures 4.11 and 4.13), an algorithm (See Section 4.4.5.2) implemented as a Java module, extracts event n-tuples as described in Vanderwende et al. (2013). Two event tuples are considered an exact match if they have the same fields and the entity labels and values for those fields match exactly. This is the strictest measure among the three, because two events do not match if one field in one event does not exactly match the same field in the other event. See Table 4.9 which shows precision, recall, and F1-score in a pairwise comparison of three annotators for both the first 20 and final 100 snippets.

4.4.3.4 IAA results

In the first stage, annotations were compared pairwise amongst the three annotators after each of the two rounds of annotation. As discussed in the previous section, the first stage of annotation occurred before the schema was extended by the addition of a diagnosis event, and negation and coordination entities. Table 4.9 lists all of the pairwise IAA scores and averages for each type of measurement: word, entity, and event tuple and both rounds of comparison: the first 20 snippets, and all 100 snippets. Table 4.10 lists the pairwise and average Kappa scores for both rounds of comparison.

		A/B			**A/C**			**B/C**			**Averages**		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Word	20	86.1	84.3	85.2	85.3	85.8	85.6	78.6	83.3	80.9	83.3	84.5	83.9
	100	81.8	83.7	82.7	85.3	85.8	85.6	79.8	79.1	79.5	82.3	82.9	82.6
Entity	20	90.4	89.2	89.7	79.3	81.3	80.1	82.4	85.6	83.8	84.0	85.4	84.5
	100	87.3	88.5	87.9	81.9	83.2	82.5	84.5	84.5	84.5	84.6	85.4	84.9
Tuple	20	68.3	68.3	68.3	56.1	53.5	54.8	61.0	58.1	59.5	61.8	60.0	60.9
	100	68.00	69.4	68.7	72.4	70.7	71.5	76.4	73.1	74.7	72.3	71.0	71.6

Table 4.9: Macro-averaged IAA F-score at the word, entity, and event level for the first 20 and final 100 snippets comparing annotators A, B, and C.

		A/B	A/C	B/C	Avg.
Word	20	92.5	89.0	92.0	91.2
	100	90.0	88.3	86.6	88.3
Entity	20	94.9	92.0	95.3	94.1
	100	91.6	90.9	88.0	90.2
Tuple	20	36.6	09.6	19.1	21.8
	100	37.3	43.1	49.4	43.3

Table 4.10: IAA Kappa at the word, entity, and event level for the first 20 and final 100 snippets comparing annotators A, B, and C.

The average results at the word and entity level are good. The annotators IAA F-scores are between 82-85 for both the first 20 and the final 100 snippets, suggesting that labeling entities is relatively easy. Lower scores for the final 100 entities is due to the occurrence of new ambiguous words not seen in the initial 20 snippets. Examples include words such as *parenchymal*, labeled as part of a `Val` entity rather than a `Loc` entity, or *airspace disease* labeled as an `Attr` or broken into a `Loc` (airspace) and `Attr` (disease). Differences in text span offsets boundaries lead to general rules for labeling `Val` entities as individual text spans and `Loc` entities as multi-word spans. For example, the words *minimal patchy* are separated into two `Val` entity text spans whereas *upper right lobe* is combined into a single `Loc` text span.

Similarly, the final 100 snippets contain some unseen ambiguous text spans, which contributed to a minimal improvement in IAA F-scores. Examples of unseen ambiguous text spans in the final 100 include coordination constructions such as *consolidation versus atelectasis* and *atelectasis, effusions, or consolidation*, which annotators either treat as a single text span or label each entity as an individual text span and did not include the coordinating conjunctions *versus* and *or* as part of the entity. Many of these issues resolved in the second stage of annotation, when the extended schema was introduced, which includes explicit `Conj` entities for coordination.

Agreement at the event level is lower than the other levels due to its exact match requirement: two event tuples match only if all entity fields match. However, the event tuple agreement for the final 100 snippets is higher than the first 20 because the discussion in the first stage helped to clarify for annotators how to annotate events. One issue addressed in discussions was where to attach ambiguous entities to the graph, such as `Ref`, which could either attach to the overall change-of-state entity or its child attribute entities.

In this section, I described the development of a change-of-state and diagnosis event event model and annotation scheme over two stages of annotation. I reviewed the annotation process, highlighting IAA measures and unique snippet corpus statistics and concluded the section with a discussion of IAA results.

4.4.4 NER module

NER is the first stage of a two-stage event extraction process I developed for change-of-state and diagnosis events. In this section, I describe how I implemented an entity detection module trained and tested on the entity annotations of the unique rationale snippet subcorpus described in Section 4.3.2.

After consideration of state-of-the-art Open Source NER systems, I selected version 3.5.2 of the CRF-based Stanford Named Entity Recognizer[9] (Finkel et al., 2005). The Stanford CRF-NER application is a Java-based general implementation of (arbitrary order) linear chain conditional random field (CRF) sequence models. For training and testing, I used the recommended default settings for the NE Recognizer[10] and for evaluation, I used the evaluation methods integrated into the Stanford CRF NER package. See Appendix C for default settings for the Stanford NE Recognizer.

The annotated labeled entity text spans in BRAT format, created during the two rounds of change-of-state and diagnosis event annotation were used to train and test an NER module in Java. In the first round the entity set was made up of `Cos`, `Attr`, `Val`, `Loc`, `Ref`, and

[9]http://nlp.stanford.edu/software/CRF-NER.shtml

[10]http://nlp.stanford.edu/software/crf-faq.shtml#a

`Diagnosis` entities, and in the second round, `Cos`, `Attr`, `Val`, `Loc`, `Ref`, `Dhead`, `Conj`, `Versus`, and `Negation` entities. The fold configurations for training and evaluating the entity detection module were created by decomposing the sub-corpus of unique rationale snippets into five randomly sampled folds based on snippet. Each snippet in the unique snippet corpus can contain one or more sentences of arbitrary token length. To simplify the distribution of training and test data across folds in the 5-fold cross-validation process, folds were allowed to vary in token size and sentence count in order to maintain the integrity of a uniform unit of information—the snippet.

To train and test the NER module with the Stanford CRF NE Recognizer, all snippets were converted into token-per-line format, with an empty line representing sentence boundaries, and with the following features associated with each token: (1) the entity assigned by annotation (i.e. `Cos`, `Val`, etc.) or `O` if no entity label was assigned to the token; (2) a *B*, *I*, or *O* label depending if the token was the beginning of an entity, inside an entity, or outside an entity; and (3) a lemma[11] for the token. See Listing C.1 for complete list of properties (the default set) used to train and test NER models for the entity detector. See Section 4.5.2 for the results of the two rounds of NER training and testing.

4.4.5 RE dependency parsing module

As mentioned in the previous section, I approached change-of-state and diagnosis event detection as a two-stage process: (1) NER, and (2) (RE) as dependency parsing. After the first stage of annotation, while developing a strategy for extracting relations from the annotated corpus, I observed that the first-stage DAG representations of change-of-states were similar to natural language dependency trees output by a dependency parser. Nivre (2006) provides canonical definitions for dependency trees and summarizes in a simple list, the constraints commonly associate with dependency trees in the literature:

---

[11]Lemma were preprocessed using the `edu.stanford.nlp.process.Morphology class`.

1. each token in a sentence is represented by a node in the dependency graph with addition of a special node `0`, which is the `ROOT` of the graph
2. all nodes in the graph are connected
3. every node has at most one head
4. the graph is acyclic
5. the graph is projective[12]

Given the definition of change-of-state and diagnosis event DAGs in Section 4.4.2, the additional constraints of items 1 and 3 in the above list transform the event DAGs into dependency trees.

Based on the description of event extraction as dependency parsing in McClosky et al. (2011), I randomly selected six annotated change-of-state snippets from the unique rationale corpus and observed that entities such as `Cos`, `Dhead`, and `Attr` form head-like dependency relations when they are attached to `Val`, `Loc`, or `Ref` entities. This is due to the way change-of-state and diagnosis events are bound by the syntax of natural language. To determine the level of effort required to treat event extraction as a dependency parsing task, I extracted labeled entities in sequence from the six randomly selected change-of-state event annotations in the BRAT format, ignoring non-labeled words and punctuation, and manually mapped the labeled arcs in the BRAT annotations to the CoNLL 2007 dependency format for representing dependency graphs. I then trained a small model with the MALT parser and tested with the same file. The parser was able to reproduce all six graphs with correct labels.

I revisited the original schema for change-of-state events to determine if all annotations in the corpus could be represented as dependency trees and found that the exceptions were change-of-state events where specific labeled arcs did not adhere to dependency tree constraints. For example, in Figure 4.14, the coordinating conjunctions, *versus* and *or*, are not explicitly part of the model, and the annotator must create multiple parent connections

[12]Approximately 20% of change-of-state and diagnosis event event trees in the unique snippet corpus are non-projective but valid dependency trees based on the canonical definitions listed in Nivre (2006).

between the `Ref` entity *Again seen* and its `Attr` parent entities, *atelectasis*, *pneumonia*, and *aspiration*.

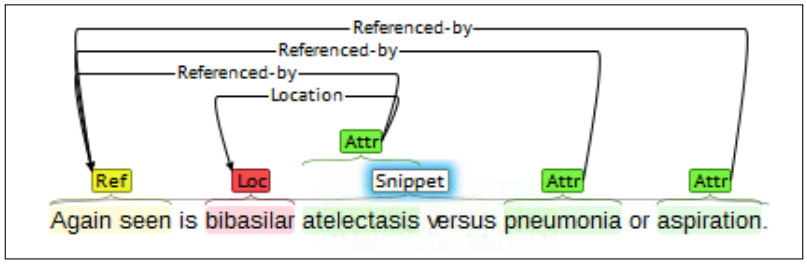

Figure 4.14: A first stage version of a change-of-state event that is a DAG but does not conform to the constraints of a dependency tree

Adding an explicit entity for coordination and labeled arcs for connecting coordinating conjunctions and the entities they coordinate allowed the DAG representation to be transformed into a dependency tree. See Figure 4.15 for a version of the same change-of-state event annotation updated using the second stage schema. In the updated example, the `Ref` entity *Again seen*, has only one parent or head, the `Versus` entity *versus*. The coordination entities, `Versus` with value *versus* and `Conj` with value *or* disambiguate the way the `Attr` entities, *atelectasis*, *pneumonia*, and *aspiration* are coordinated in the sentence and constrain these relations to be single-headed.

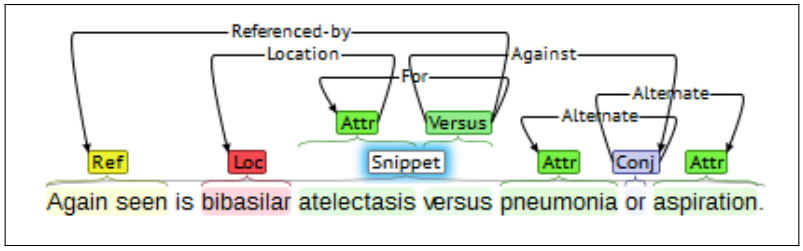

Figure 4.15: A second stage version of a change-of-state event that conforms to the constraints of a dependency tree

```
1  lungs         lung          Loc    Loc    T545 5  Location _  _
2  increased     increase      Cos    Cos    T546 0  ROOT     _  _
3  bibasilar     bibasilar     Loc    Loc    T547 4  DEP      _  _
4  pulmonary     pulmonary     Loc    Loc    T547 5  Location _  _
5  opacities     opacity       Attr   Attr   T548 2  State    _  _
6  likely        likely        Dhead  Dhead  T781 7  DEP      _  _
7  representing  represent     Dhead  Dhead  T781 0  ROOT     _  _
8  atelectasis   atelectasis   Attr   Attr   T785 10 Combine  _  _
9  aspiration    aspiration    Attr   Attr   T784 10 Combine  _  _
10 or            or            Conj   Conj   T782 7  Diag     _  _
11 infection     infection     Attr   Attr   T783 10 Combine  _  _
```

Listing 4.1: Example of a rationale snippet in CoNLL 2007 format

By adding a diagnosis event and entities for coordination and negation, I was able to convert all of the BRAT entities and relations from the unique rationale snippet corpus into CoNLL 2007 dependency format. See 4.1 for an example of an unique rationale text snippet converted to CoNLL 2007 format. See Appendix D for a brief description of the CoNLL 2007 format. Converting the DAG structures of the first round of annotation into dependency trees by adding additional entities for negation and coordination to the change-of-state schema was the primary reason for a second round of annotation. I maintained the two sets of annotation in order to evaluate the cost or benefit of additional entities to the performance of the NER extraction module.

I then trained and tested dependency parsing models based on the annotated pneumonia report classification corpus of 1008 unique rationale snippets, one version with all tokens and punctuation included in the token sequence and another version using only the sequence of entity tokens, and all other tokens and punctuation removed. I treated these experiments as oracle snippet experiments, where the named entities are not extracted by the NER module, but rather from the gold standard.

I also trained and tested a separate collection of dependency models based on the system-generated named entities extracted by my NER module. Similar to the oracle snippet experiments, these experiments have two versions, one, where all tokens and punctuation in the sequence are included in the model and output files, and a version where only system-generated entities are included in the model and output files. Due to the possibility that the system-generated named entities used in the entity-only experiments may not share the same token sequences as the gold standard, a merge algorithm re-integrates the entity-only token sequence back into a canonical sequence, including all tokens and punctuation, for evaluation purposes.

I used the 2.7.2 version of the MALT dependency parser[13] (Nivre et al., 2007), with default configuration properties, for training and testing dependency trees. I used MaltEval[14] (Nilsson and Nivre, 2008) for evaluation. MaltEval is a standalone Java application created to provide the dependency parsing community a new quantitative and qualitative tool to evaluate the performance of dependency parsers and analyze errors in place of the Perl script, `eval07.pl`, used for evaluation in the original CoNLL 2007 Shared Task on Dependency Parsing. The version of MaltEval I used for evaluation was released October, 2014.

For the conversion to CoNLL 2007 dependency format, I made non-final words of an entity depend on the final word with a dummy arc label `DEP`. Furthermore, the head of the entity at the root of an event tree depends on a dummy node with arc label `ROOT`. For instance, given a sentence *w1 w2 w3 w4 w5*, assume that *w1 w2* and *w4 w5* are two entities and the first entity depends on the second entity with the label `REL`. After the conversion, *w1* will depend on *w2* with a `DEP` label, and the same is true for *w4* and *w5*; *w2* will depend on *w5* with label `REL` and *w5* on a dummy word with label `ROOT`; *w3* is ignored since it is not part of any entity.

[13]http://www.maltparser.org/

[14]http://www.maltparser.org/malteval.html

```
1 2          3  4    5    6  7  8      9  10
1 minimal    _  Val  Val  _  2  DEP    _  _
2 patchy     _  Val  Val  _  3  REL_1  _  _
3 edema      _  Attr Attr _  4  DEP    _  _
4 improved   _  Cos  Cos  _  0  ROOT   _  _
```

Listing 4.2: Example of a rationale snippet converted to CoNLL 2007 format

A rationale snippet can contain multiple sentences and a sentence can in turn contain multiple event trees. In the unique snippet corpus, there are 1008 unique snippets, 1268 sentences, and 1743 `ROOT` arcs (one for each event tree).

4.4.5.1 Measuring performance with MaltEval

The default measurements of performance in MaltEval, described in the list below, compare CoNLL-formatted system-generated and gold-standard parses by token. MaltEval considers a token to be the collection of properties described by the ten fields in a single row of a file in the CoNLL format. MaltEval default token counts concern the values of two of the columns in the format, `HEAD` in column 7 and `DEPREL` in column 8. `HEAD`, identifies the head of a token by its `ID` column or, if the token is the root of the dependency relation, it is assigned `0`. `DEPREL` contains the label of the dependency relation. If the `HEAD` value of the token is `0`, and it is indeed the head of the dependency tree, its `DEPREL` will be labeled `ROOT`. A multi-word token, such as a named entity, can be described by labeling its non-head tokens with the dependency relation `DEP` and its `HEAD` value the `ID` of its head token. See Listing 4.2 for an example of a simple change-of-state event represented in the CoNLL format.

MaltEval offers additional types of measurements for dependency parsing, however, the measurements of performance used in my experiments are based only on counts of token properties. Three measures count and compare tokens by values in the `HEAD` and/or `DEPREL` columns and report an overall accuracy with no detailed breakdown of calculations:

Label attachment score (LAS) measures both `HEAD` and `DEPREL` values. If both are the same in system generated and gold standard tokens, the system token is considered a correct LAS match.

Unlabeled attachment score (UAS) measures only `HEAD` values. If system generated and gold standard tokens have the same `HEAD` value, the system token is considered a correct UAS match.

Labeled accuracy (LA) measures only `DEPREL` values. If system generated and gold standard tokens have the same `DEPREL` value, the system token is considered a correct LA match.

These three measures were developed for CoNLL 2006 and 2007 Shared Tasks on Dependency Parsing and have become a standard way to compare parsing performance. See Listing C.2 in Appendix C for an example of a MaltEval configuration file.

I reported LA, LAS, and UAS two ways, including `DEP` and `ROOT` relations, and without. `DEP` and `ROOT` relations are common to all dependency schemas and reporting the results two ways allows for a measure of labeling with only the labels from the change-of-state and diagnosis event schemas applied. I also included in the reports, a breakdown of the label accuracy (LA) calculations by dependency relation (`DEPREL`), reporting precision, recall, and F-score. See Table 4.27 for a complete evaluation report of dependency parsing for snippets with oracle NER entities including all tokens and punctuation, Table 4.28 for a report with oracle snippets and entities only, Table 4.29 for dependency parses based on system-generated named entities including all tokens and punctuation, and Table 4.30 for system-generated named entities only, merged with all tokens and punctuation for evaluation.

4.4.5.2 N-Tuple generation algorithm

Creating tuple representations from the change-of-state and diagnosis event graph demonstrates fidelity to the original event structure proposed by Vanderwende et al. (2013) and simplifies the creation of event-based features as described in Section 4.4.6. The change-of-state and diagnosis event n-tuple represents a single event whereas the change-of-state and diagnosis event dependency tree graph contains one or more events or n-tuples. The algorithm used to derive event tuples from the dependency tree graph created by the entities and their labeled arcs in change-of-state and diagnosis events is a simple recursive graph traversal from the root entity to the terminal nodes of the graph. Tuple slots are filled as the graph is traversed and if a new entity is found and its slot is filled, a new tuple is created using all of the original tuples values with the new entity filling the slot position of its similarly named entity in the previous step. All combinations of slots and entities are created with regard to duplication (duplicate tuples by entity id are removed in final step). See Figure 4.11 and Figure 4.12 for an example of a dependency tree and a derived change-of-state and diagnosis event tuple.

4.4.6 Event-based feature extraction

I introduce six types of event-based features that I integrated into the feature sets for pneumonia report classification experiments. Table 4.11 provides a description of the new event-based features and Table 4.12 describes the feature type membership in the canonical features sets used in my pneumonia report classification experiments.

Feature	Description
Entity	Name and value pair of a event-based named entity annotation Pattern: {Entity Type} = {Entity Value} Example: `VAL=CLEAR`
Attribute	Attribute of an event-based named entity annotation (only used in oracle experiments—no extractor was developed to label attributes, the annotations only exist in the gold standard) Pattern: { Attribute Value} Example: `INCREASED`
Tuple	N-Tuple extracted from a change-of-state or diagnosis event Pattern: [{`Cos`_Value},{`Attr`_Value},{`Val`_Value},{`Loc`_Value},{`Ref`_Value}] Example: `[NEW_NEG,ABNORMALITIES,FOCAL,LUNG,AS_PER_LAST_EXAM]`
Change-of-state	Specific `Cos` value replaced with a generic `COS`. `Val` fields in the n-tuple are normalized with an underscore if they are multi-word, and prefix the `Attr` value. `Loc` entities are added as a suffix to the `Attr` value with an @ symbol. Pattern: `COS`→{`Val_Value`}_{`Attr_Value`}_`@`_{`Loc_Value`} Example: `COS`→`PATCHY_EDEMA_@_LEFT_LUNG`
Diagnosis	Specific `Dhead` value replaced with a generic DIAGNOSIS, `Val` and `Loc` fields follow same pattern as Change-of-state feature above. Pattern: `DIAGNOSIS`→{`Val_Value`}_{`Attr_Value`}_`@`_{`Loc_Value`} Example: `DIAGNOSIS`→`PNEUMONIA_@_RIGHT_LUNG`
Observation	Specific `Attr` value replaced with a generic OBS (for observation), `Val` and `Loc` fields follow same pattern as Change-of-state feature above. Pattern: `OBS`→{`Val_Value`}_{`Attr_Value`}_`@`_{`Loc_Value`} Example: `OBS`→`PATCHY_ATELECTASIS_@_LUNGS`

Table 4.11: A description and example of event-based features generated from change-of-state and diagnosis event dependency trees

4.4.7 Pneumonia report classification experiments

In the final stage of the pneumonia report classification study, I integrated the event-based features I defined in the previous section, into the baseline experiments I replicated from Tepper et al. (2013). Table 4.12 describes the three sets of feature types that I used in my experiments.

Feature	Baseline	Event-only	All
Word Unigram	X		X
Alternating Conjunction	X		X
Entity Attribute Name & Value		X	X
Entity Name & Value		X	X
Event Tuple		X	X
Change-of-state		X	X
Diagnosis		X	X
Observation		X	X

Table 4.12: A table comparing the features that make up the three types of feature sets used in pneumonia report classification experiments

The results of the pneumonia report classification experiments, discussed in Section 4.5.4, suggested that the addition of event-based features to the baseline n-gram features and the use of a feature threshold of 250 improved the performance of the pneumonia report classifier for both categories, CPIS and PNA, when applied to the whole text of the report. Results for predicted and oracle snippets were mixed, however, with improvement over the baseline for the CPIS category but not for PNA. To explore the effect of feature selection across the feature sets, feature threshold experiments were run using increasing increments of features up to the maximum number of significant features ranked by χ^2. See Section 4.5.4.1 for the results of the feature threshold experiments for whole documents and Appendix E for the results of feature threshold experiments for predicted and oracle snippets.

4.5 Results

In this section, I detail the results of pneumonia report classification experiments and evaluation measures for corpus annotation and snippet prediction, NER, and RE module development. Table 4.13 describe the subsections of the results Section.

Section	Description
Baseline for snippet prediction	Rationale snippet prediction experiments and baseline VAP report classification experiments replicated from descriptions in Tepper et al. (2013)
Inter-annotator agreement	Two rounds of annotations by three NLP researchers during the first stage of change of state annotation of the unique rationale snippet corpus
NER	NER experiments on the unique rationale snippet corpus as a result of the two stages of annotation
Event detection	Event detection experiments using a dependency parser to draw labeled arcs between NER annotated entities from sentences in the unique rationale snippets corpus
Event-based classification	Pneumonia report classification experiments integrating features extracted from event annotations

Table 4.13: Results section overview

All experiments in this study were limited by the amount of data available for training and testing and implemented 5-fold cross-validation. Each of the five folds was broken down into a 80%/20% split of training and testing data. There was no data overlap between testing and training splits, and each fold's test split did not overlap with any other fold's test split. See Table 1 in the dissertation frontmatter for a list of the abbreviations and terms used in the result tables headers.

4.5.1 Baseline results

In this section, I report snippet prediction module performance and pneumonia report classification results generated by the baseline system I replicated from the system described in Tepper et al. (2013). See Section 4.4.1 for a detailed review of the baseline system implementation phase.

4.5.1.1 Snippet prediction results

Table 4.14 lists the results of snippet prediction in Tepper et al. (2013) using measures of sentence overlap and snippet overlap. I implemented an exact match sentence label score which is similar, but not exactly the same, as sentence overlap. I treated the snippet prediction task as a binary sentence labeling task and normalized the labels of all sentences in the corpus before training and testing. The label boundaries match the sentences exactly, so there is no overlap of a label across sentences. My measure is simply an exact sentence label to sentence label comparison, where a sentence can either be labeled *Inside* a snippet or *Outside* a snippet. Contiguous *Inside* sentences are considered part of a single multi-sentence snippet. The results of my replicated snippet prediction module are listed in Table 4.15.

My snippet prediction module is a binary sentence classifier, and its *trained on reports with snippets only* version performed with an F-score of 88.3 for CPIS, and 70.2 for PNA, compared to Tepper et al. (2013)'s snippet prediction module with *no prevTag* feature (equivalent to a binary classifier), trained on reports with snippets only, which performed with an F-score of 88.4 for CPIS, and 70.4 for PNA. The performance of the snippet prediction modules for both CPIS and PNA are very similar and suggest the exact match sentence label to sentence label measure I used to evaluate the performance of my snippet prediction module is equivalent to the sentence overlap measure used in Tepper et al. (2013).

Another observation regarding Table 4.15 is that a snippet prediction module trained on reports with snippets only does not outperform a snippet prediction module trained on all reports when compared using the exact sentence label match measure. The module trained

on all reports has a slightly higher F-score, however, for both CPIS and PNA, the *trained on reports with snippets only* module has higher recall. The higher recall leads to more false positive snippets predicted, but this in turn helps improve F-score performance for both CPIS and PNA report classification as demonstrated by the results in Table 4.16. Based on these results, I trained my snippet prediction module on reports with snippets only, because higher recall is preferred over precision for this task.

Type	Feature Model	Sentence Overlap			Snippet Overlap		
		P	R	F1	P	R	F1
CPIS	w/prevTag feature	91.1	84.0	87.4	95.6	89.9	92.7
	no prevTag feature	86.6	90.4	**88.4**	91.6	96.3	93.9
PNA	w/prevTag feature	61.2	88.9	72.5	67.6	94.2	78.7
	no prevTag feature	62.8	80.2	**70.4**	66.8	94.0	78.1

Table 4.14: Performance results for snippet prediction trained on reports with snippets only originally published in Tepper et al. (2013)

Type	Trained On	S	G	Θ	TP	TN	FP	FN	P	R	F1	Acc
CPIS	reports with snippets only	1735	1648	250	1494	11653	241	154	86.1	90.7	**88.3**	97.1
	All reports	1622	1648	250	1448	11720	174	200	89.3	87.9	88.6	97.3
PNA	reports with snippets only	1422	1135	250	898	11921	524	237	63.2	79.1	**70.2**	94.4
	All reports	858	1135	250	635	12222	223	500	74.0	55.9	63.7	94.7

Table 4.15: Performance results by exact sentence match (similar to sentence overlap measure) for replicated snippet prediction module trained on all reports or reports with snippets only

Type	Trained On	S	G	Θ	TP	FP	FN	P	R	F1	Acc
CPIS	Snippets only	1341	1341	0	1155	186	186	76.9	72.4	**74.2**	86.1
	All reports	1341	1341	0	1144	197	197	74.3	71.9	72.3	85.3
PNA	Snippets only	1343	1343	0	1093	250	250	74.4	72.2	**72.4**	81.4
	All reports	1343	1343	0	1083	260	260	73.9	71.3	71.7	80.6

Table 4.16: Baseline pneumonia report classification results for predicted snippets trained on reports with snippets only compared to trained on all reports.

As in the original system described in Tepper et al. (2013), a feature selection threshold (θ) of 250 χ^2 feature selection approach was used for all replicated CPIS and PNA snippet prediction experiments. Table 4.17 lists the top 10 word unigram features for both labels. The list contains word unigrams one would expect for the domain.

CPIS	PNA
ATELECTASIS	ATELECTASIS
OPACITIES	PNEUMONIA
LUNG	OPACITIES
DIFFUSE	LUNG
PNEUMONIA	DIFFUSE
EDEMA	EDEMA
LOBE	PATCHY
PATCHY	LOBE
FOCAL	CONSISTENT
BIBASILAR	LOWER

Table 4.17: Top 10 word unigram features (all feature values converted to uppercase for case insensitive ranking) ranked by χ^2 feature selection for replicated CPIS and PNA snippet prediction experiments

4.5.1.2 Baseline pneumonia report classification results

Table 4.18 lists the results of Tepper et al. (2013)'s baseline pneumonia report classification experiments for CPIS and PNA, with whole reports, oracle snippets, and predicted snippets using the baseline feature set (unigrams and alternating conjunctions). Table 4.19 lists the results of the same experiments run on the replicated baseline system including additional experiments incorporating χ^2 feature selection with a threshold (θ) of 250 features.

	Unit of Classification	S	G	Θ	TP	FP	FN	P	R	F1	Acc
CPIS	Whole Document	1341	1341	0	1150	191	191	79.3	66.9	72.6	85.8
	Predicted Snippets	1341	1341	0	1168	173	173	78.9	75.0	76.9	87.1
	Oracle Snippets	1341	1341	0	1232	109	109	88.0	83.9	85.9	91.9
PNA	Whole	1343	1343	0	1054	289	289	70.2	66.6	68.4	78.5
	Predicted Snippets	1343	1343	0	1103	240	240	75.4	72.6	74.0	82.1
	Oracle Snippets	1343	1343	0	1137	206	206	79.3	75.8	77.5	84.3

Table 4.18: Pneumonia report classification results from Tepper et al. (2013) with baseline features

	Unit of Classification	S	G	Θ	TP	FP	FN	P	R	F1	Acc
CPIS	Whole	1341	1341	0	1134	207	207	74.3	68.9	71.3	84.6
		1341	1341	250	1153	188	188	77.0	71.1	73.4	86.0
	Predicted Snippets	1341	1341	0	1155	186	186	76.9	72.4	74.2	86.1
		1341	1341	250	1154	187	187	77.6	71.1	73.4	86.1
	Oracle Snippets	1341	1341	0	1217	124	124	85.6	81.0	83.0	90.8
		1341	1341	250	1224	117	117	87.3	81.3	83.4	91.3
PNA	Whole	1343	1343	0	1047	296	296	70.1	68.8	69.3	80.0
		1343	1343	250	1085	258	258	74.1	71.7	72.5	80.8
	Predicted Snippets	1343	1343	0	1093	250	250	74.4	72.2	72.4	81.4
		1343	1343	250	1119	224	224	77.5	74.4	75.1	83.3
	Oracle Snippets	1343	1343	0	1139	204	204	78.6	77.1	77.6	84.8
		1343	1343	250	1147	196	196	79.3	77.8	78.3	85.4

Table 4.19: Replicated baseline pneumonia report classification results with baseline features.

Table 4.20 compares, side by side the results of experiments with baseline features in Tepper et al. (2013) (System A) and the replicated system, with feature selection (System B*), and without feature selection (System B).

Text Boundary	Type	System	θ	P	R	F1	Acc
Whole Document	CPIS	A	0	79.3	66.9	72.6	85.8
		B	0	74.3	68.9	71.3	84.6
		B*	250	77.0	71.1	73.4	86.0
	PNA	A	0	70.2	66.6	68.4	78.5
		B	0	70.1	68.8	69.3	80.0
		B*	250	74.1	71.7	72.5	80.8
Predicted Snippet	CPIS	A	0	78.9	75.0	76.9	87.1
		B	0	76.9	72.4	74.2	86.1
		B*	250	77.6	71.1	73.4	86.1
	PNA	A	0	75.4	72.6	74.0	82.1
		B	0	74.4	72.2	72.4	81.4
		B*	250	77.5	74.4	75.1	83.3
Oracle Snippet	CPIS	A	0	88.0	83.9	85.9	91.9
		B	0	85.6	81.0	83.0	90.8
		B*	250	87.3	81.3	83.4	91.3
	PNA	A	0	79.3	75.8	77.5	84.3
		B	0	78.6	77.1	77.6	84.8
		B*	250	79.3	77.8	78.3	85.4

Table 4.20: Comparison of pneumonia report classification experiments with baseline features

The charts in Figures 4.16 and 4.17 present a visual comparison of the three systems and demonstrate that although there is differences in performance, the systems all follow a similar pattern when features are extracted from whole reports, predicted snippets, and oracle snippets.

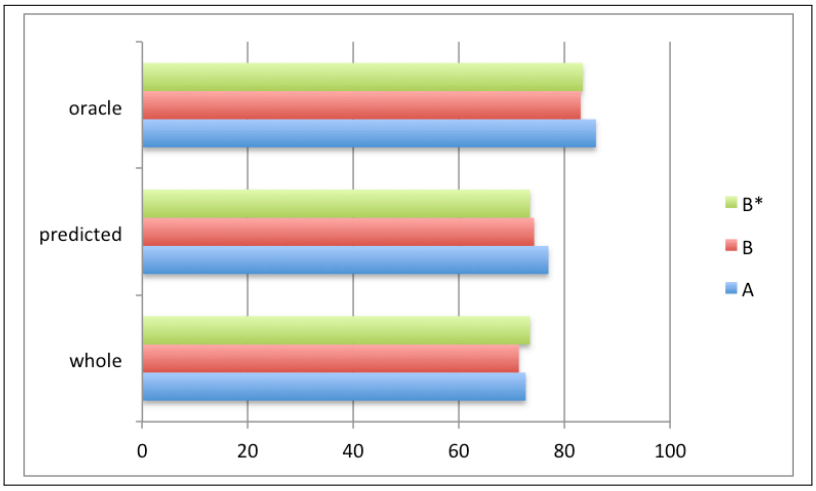

Figure 4.16: Comparison of F1 score in CPIS classification experiments with baseline features: Tepper et al. (2013) (System A), replicated baseline (System B), and replicated baseline with feature selection Θ=250 (System B*)

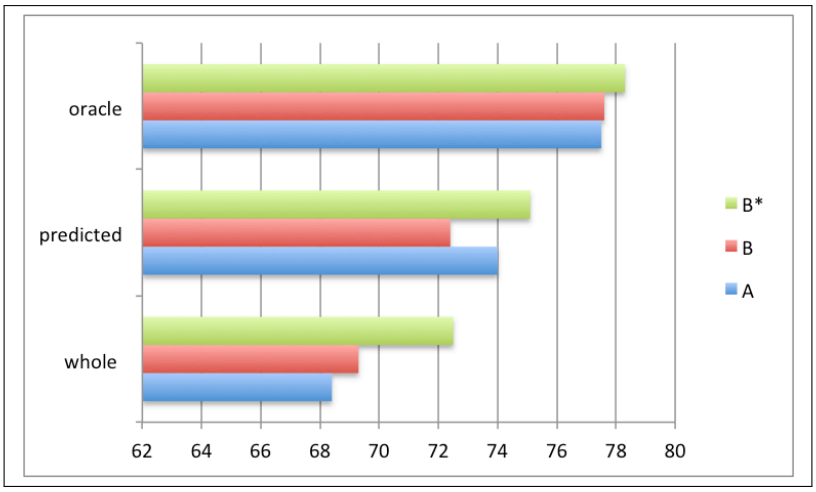

Figure 4.17: Comparison of F1 score in PNA classification experiments with baseline features: Tepper et al. (2013) (System A), replicated baseline (System B), and replicated baseline with feature selection Θ=250 (System B*)

Although the results of the replicated system (Systems B and B*) do not match the performance results published in Tepper et al. (2013), they do follow a similar pattern, both with and without feature selection, of improved performance when features are extracted from predicted and oracle snippets over features extracted from the entire report. For my pneumonia report classification experiments, I want to determine if event-based features can improve the performance of the pneumonia report classifier, and if so, by how much. Therefore, Tables 4.21 and 4.22 are the baseline F-scores I target in my experiments integrating event-based features, with and without feature selection.

Text Boundary	Type	F1
Whole Document	CPIS	71.3
	PNA	69.3
Predicted Snippet	CPIS	74.2
	PNA	72.4
Oracle Snippet	CPIS	83.0
	PNA	77.6

Table 4.21: Performance of the replicated pneumonia report classification system (System B) with baseline features

Text Boundary	Type	F1
Whole Document	CPIS	73.4
	PNA	72.5
Predicted Snippet	CPIS	73.4
	PNA	75.1
Oracle Snippet	CPIS	83.4
	PNA	78.3

Table 4.22: Performance of the replicated pneumonia report classification system (System B*) with baseline features, feature selection, and a feature threshold θ=250

4.5.2 NER results

The results of the NER performance evaluation, performed after both the first and second stage of annotation, are listed at the entity level for round one in Table 4.23, at the token level for round one in Table 4.24, at the entity level for round two in Table 4.25, and at the word level for round two in Table 4.26. The NER module depends on the Stanford CRF-NE Recognizer, version 3.5.2, for training models and labeling test files.

In general, the system performs very well after both stages. For the first stage of annotation, six named entities are included in the evaluation: `Cos`, `Attr`, `Val`, `Loc`, `Ref`, and `Diagnosis`. In the second stage, The lowest performing entities, `Dhead` and `Ref`, are the ones in the set that were most likely to contain general and productive non-domain specific language, such as terms of likelihood and possibility in the case of `Dhead`, and general descriptions of previous reports, conditions, or patient state in the case of `Ref`.

The improvement of performance with the addition of entities in the second round is unexpected. Additional entities add more complexity to the task, but may also, as the results demonstrate, restrict the tokens in the string that are labeled `O`. This may reduce the ambiguity of tokens representing non-change-of-state event entities such as the additional entities for coordination and negation.

Entity type	S	G	TP	FP	FN	P	R	F1
`Attr`	1616	1633	1527	89	106	94.5	93.5	94.0
`Cos`	1021	1020	935	86	85	91.6	91.7	91.6
`Val`	814	855	734	80	121	90.2	85.9	88.0
`Loc`	1651	1657	1529	122	128	92.6	92.3	92.4
`Ref`	174	181	138	36	43	79.3	76.2	77.8
`Diagnosis`	451	439	375	76	64	83.2	85.4	84.3
Totals	5727	5785	5238	489	547			
				macro average		88.6	87.5	88.0
				micro average		91.5	90.5	91.0

Table 4.23: Performance of Stanford CRF-NER on the unique rationale snippet corpus, at the **entity** level, after the first stage of annotation.

Entity type	S	G	TP	FP	FN	P	R	F1
`Attr`	1670	1704	1592	78	112	95.3	93.4	94.4
`Cos`	1548	1559	1467	81	92	94.8	94.1	94.4
`Val`	1113	1182	1036	77	146	93.1	87.7	90.3
`Loc`	2921	2948	2792	129	156	95.6	94.7	95.1
`Ref`	462	470	423	39	47	91.6	90.0	90.8
`Diagnosis`	2759	2626	2436	323	190	88.3	92.8	90.5
Totals	10473	10489	9746	727	743			
				macro average		93.1	92.1	92.6
				micro average		93.1	92.9	93.0

Table 4.24: Performance of Stanford CRF-NER on the unique rationale snippet corpus, at the **token** level, after the first stage of annotation.

For the second stage, nine entities are included in the evaluation: `Cos`, `Val`, `Attr`, `Val`, `Loc`, `Ref`,`Conj`, `Versus`, and `Negation`.

Entity type	S	G	TP	FP	FN	P	R	F1
`Attr`	2414	2404	2346	68	58	97.2	97.6	97.4
`Cos`	1067	1067	1011	56	56	94.8	94.8	94.8
`Dhead`	510	517	454	56	63	89.0	87.8	88.4
`Loc`	2004	1996	1899	105	97	94.8	95.1	95.0
`Ref`	176	179	155	21	24	88.1	86.6	87.3
`Val`	908	937	837	71	100	92.2	89.3	90.7
`Conj`	678	671	636	42	35	93.8	94.8	94.3
`Versus`	35	39	34	1	5	97.1	87.2	91.9
`Negation`	252	253	244	8	9	96.8	96.4	96.6
Total	8044	8063	7616	428	447			
					macro average	93.7	92.2	92.9
					micro average	94.7	94.5	94.6

Table 4.25: Performance of Stanford CRF-NER on the unique rationale snippet corpus, at the **entity** level, after the second stage of annotation.

Entity type	S	G	TP	FP	FN	P	R	F1
Attr	2492	2499	2444	48	55	98.1	97.8	98.0
Cos	1293	1306	1240	53	66	95.9	95.0	95.4
Dhead	1037	1026	962	75	64	92.8	93.8	93.2
Loc	3240	3249	3164	76	85	97.7	97.4	97.5
Ref	495	466	438	57	28	88.5	94.0	91.2
Val	1132	1160	1056	76	104	93.8	91.0	92.2
Conj	696	697	654	42	43	94.0	93.8	93.9
Versus	35	40	34	1	6	97.1	85.0	90.7
Negation	255	262	246	9	16	96.5	93.9	95.2
Total	10675	10705	10238	437	467			
				macro average		94.9	93.5	94.1
				micro average		95.9	95.6	95.8

Table 4.26: Performance of Stanford CRF-NER on the unique rationale snippet corpus, at the **token** level, after the second stage of annotation.

4.5.3 Oracle and predicted event extraction results

Event extraction for the unique rationale snippet is a dependency parsing task. The labeled arcs used to create connections between entities in the first stage annotation schema did not support the creation of true dependency trees for all change-of-state and diagnosis events in the unique rationale snippet corpus. After the second stage of annotation, the addition of a diagnosis event, and explicit coordination and negation entities, enabled annotators to create dependency trees with the labeled arcs available in the schema. There are twelve labeled arc types included in the performance evaluation of the dependency parsing module, not including the two special cases: `ROOT` and `DEP`, which describe the special node `0`, characteristic of a dependency tree, and the *dependent* relation, `DEP`, which describes a non-head token's relation to its head in a dependent dependency relation. Token nodes are connected in the dependency tree with either *head* relations, which are the names of labeled arcs in the annotation scheme, or *dependent* relations, which are named `DEP`, and only occur in the

dependency tree (they are the the tokens in the entity annotation text span to the left of the head token).

Version 2.7.2 of the MALT parser was used to train models and parse the NE annotated unique rationale snippet corpus output by the NER module. MaltEval was used to evaluate the output dependency trees in the CoNLL 2007 Shared Task for Dependency Parsing format.

The event extraction process was evaluated in four scenarios:

1) Oracle NER entities with all text and punctuation Unique rationale snippets with gold standard named entities and labeled arcs, created during the second stage of annotation were used to train and test. All non-entity tokens and punctuation were included in the annotated sentences used to test and train. See Table 4.27 for detailed, per dependency label, Label Accuracy (LA) results for this scenario.

2) Oracle NER entities with only entities The same snippets with gold standard named entities and labeled arcs, created during the second stage of annotation, were used to train and test. Only tokens that participated in a dependency relation were included in the sentences used to train and test. See Table 4.28 for detailed, per dependency label, Label Accuracy (LA) results for this scenario.

3) System-generated NER entities with all text and punctuation The same snippets with gold standard named entities and labeled arcs, created during the second stage of annotation were used to train the models. Snippet sentences with only system-generated NER entities were used to test the models. The testing sentences included all punctuation and non-entity tokens in the original sentence string. See Table 4.29 for detailed, per dependency label, Label Accuracy (LA) results for this scenario.

4) System-generated NER entities with only entities The same snippets with gold standard named entities and labeled arcs, created during the second stage of annotation were used to train the models. Snippet sentences with only system-generated NER entities were used to test the models. The testing sentences included only tokens that participated in dependency relations. An algorithm using trace metadata in column 7

of the CoNLL 2007 dependency format was used to merge the entity-only token string with the original complete sentence token string in the gold standards to evaluate. This was done to accommodate the scenario in which the tokens that participated in the entity-only test output differed from the tokens in the original string that participated in dependency relations. The merge algorithm ensures that there is a token-to-token mapping between strings for evaluation. See Table 4.30 for detailed, per dependency label, Label Accuracy (LA) results for this scenario.

The strategy of removing the non-participating tokens in the sentence string before training or testing improved both precision and recall as demonstrated in the comparison of results in all four tables below as well as summary accuracy scores in Table 4.31.

Arc label	System	Gold	TP	FP	FN	P	R	F1
Diag	417	526	398	19	128	95.4	75.7	84.4
Location	1443	1828	1395	48	433	96.7	76.3	85.3
ReferencedBy	102	166	101	1	65	99.0	60.8	75.4
State	602	1072	559	43	513	92.9	52.1	66.8
Value	926	994	898	28	96	97.0	90.3	93.5
Alternate	355	430	340	15	90	95.8	79.1	86.6
Against	38	38	34	4	4	89.5	89.5	89.5
Combine	809	984	718	91	266	88.8	73.0	80.1
Contrast	0	17	0	0	17	_	0.0	_
Exclude	8	11	7	1	4	87.5	63.6	73.7
For	37	39	34	3	5	91.9	87.2	89.5
Negate	249	252	244	5	8	98.0	96.8	97.4
Subtotal	4986	6357	4728	258	1629			
				macro average		86.0	70.4	76.9
				micro average		94.8	74.4	83.4
ROOT	3159	1769	1741	1418	28	55.1	98.4	70.7
DEP	2560	2579	2546	14	33	99.5	98.7	99.1
Total	10705	10705	9015	1690	1690			
				macro average		84.8	74.4	78.0
				micro average		84.2	84.2	84.2

Table 4.27: Performance of dependency parsing of **oracle** named entities from unique rationale snippets including **all tokens and punctuation**

Arc label	System	Gold	TP	FP	FN	P	R	F1
Diag	501	526	470	31	56	93.8	89.4	91.5
Location	1748	1817	1672	76	145	95.7	92.0	93.8
ReferencedBy	159	166	157	2	9	98.7	94.6	96.6
State	1000	1066	919	81	147	91.9	86.2	89.0
Value	985	987	962	23	25	97.7	97.5	97.6
Alternate	433	430	410	23	20	94.7	95.3	95.0
Against	37	38	32	5	6	86.5	84.2	85.3
Combine	884	984	781	103	203	88.3	79.4	83.6
Contrast	9	17	1	8	16	11.1	5.9	7.7
Exclude	6	11	6	0	5	100.0	54.5	70.6
For	40	39	34	6	5	85.0	87.2	86.1
Negate	252	252	247	5	5	98.0	98.0	98.0
Subtotal/	6054	6333	5691	363	642			
			macro average			86.8	80.4	82.9
			micro average			94.0	89.9	91.9
ROOT	2019	1745	1675	344	70	83.0	96.0	89.0
DEP	2584	2579	2538	46	41	98.2	98.4	98.3
Total	10657	10657	9904	753	753			
			macro average			87.3	82.8	84.4
			micro average			92.9	92.9	92.9

Table 4.28: Performance of dependency parsing of **oracle** named entities from unique rationale snippets including **entities only**

Arc label	System	Gold	TP	FP	FN	P	R	F1
Diag	428	526	391	37	135	91.4	74.3	82.0
Location	1445	1828	1367	78	461	94.6	74.8	83.5
ReferencedBy	95	166	92	3	74	96.8	55.4	70.5
State	581	1072	536	45	536	92.3	50.0	64.9
Value	893	994	830	63	164	92.9	83.5	88.0
Alternate	357	430	338	19	92	94.7	78.6	85.9
Against	35	38	31	4	7	88.6	81.6	84.9
Combine	831	984	678	153	306	81.6	68.9	74.7
Contrast	1	17	0	1	17	0.0	0.0	_
Exclude	8	11	4	4	7	50.0	36.4	42.1
For	34	39	32	2	7	94.1	82.1	87.7
Negate	249	252	241	8	11	96.8	95.6	96.2
Subtotal	4957	6357	4540	417	1817			
				macro average		81.2	65.1	71.7
				micro average		91.6	71.4	80.3
ROOT	3156	1769	1662	1494	107	52.7	94.0	67.5
DEP	2562	2579	2354	208	225	91.9	91.3	91.6
Total	10675	10705	8556	2119	2149			
				macro average		79.9	69.0	72.8
				micro average		80.1	79.9	80.0

Table 4.29: Performance of dependency parsing of **system-generated** named entities from unique rationale snippets including **all tokens and punctuation**

Arc label	System	Gold	TP	FP	FN	P	R	F1
Diag	504	526	453	51	73	89.9	86.1	88.0
Location	1758	1828	1636	122	192	93.1	89.5	91.2
ReferencedBy	154	166	142	12	24	92.2	85.5	88.8
State	1000	1072	895	105	177	89.5	83.5	86.4
Value	948	994	890	58	104	93.9	89.5	91.7
Alternate	430	430	407	23	23	94.7	94.7	94.7
Against	35	38	30	5	8	85.7	78.9	82.2
Combine	865	984	737	128	247	85.2	74.9	79.7
Contrast	6	17	0	6	17	0.0	0.0	_
Exclude	7	11	7	0	4	100.0	63.6	77.8
For	34	39	31	3	8	91.2	79.5	84.9
Negate	253	252	240	13	12	94.9	95.2	95.0
Subtotal	5994	6357	5468	526	889			
				macro average		84.2	76.7	80.0
				micro average		91.2	86.0	88.5
ROOT	2028	1769	1640	388	129	80.9	92.7	86.4
DEP	2493	2579	2407	86	172	96.6	93.3	94.9
Total	10515	10705	9515	1000	1190			
				macro average		84.8	79.1	81.6
				micro average		90.5	88.9	89.7

Table 4.30: Performance of dependency parsing of **system-generated** named entities from unique rationale snippets including **entities only**

Table 4.31 lists the summary Label Accuracy (LA) (head and dependency label), Unlabeled Attachment Score (UAS) (head only), and Label Attachment Score (LAS) (dependency label only) for the four scenarios described in tables above. Each row scenario lists the results when ROOT and DEP arcs and included in the evaluation, and when they are not. These are the standard scores used to compare dependency parsing systems and are based on the format of the results used in the CoNLL 2007 Shared Task on Dependency Parsing.

	ROOT and DEP arcs	LA	UAS	LAS
Oracle NER with all tokens and punctuation	w/o	94.8	94.6	93.6
	w	84.2	84.1	83.6
Oracle NER with entities only	w/o	94.0	93.3	92.2
	w	92.9	92.4	91.7
System-generated NER with all tokens and punctuation	w/o	91.6	91.5	89.9
	w	80.0	80.0	78.4
System-generated NER with entities only	w/o	91.1	90.6	88.8
	w	89.8	88.8	87.5

Table 4.31: Performance of dependency parsing of NER, measured by labeled accuracy (LA), unlabeled attached score (UAS), and labeled attachment score (LAS).

Figure 4.18 compares all four scenarios for training and testing dependency parsing: (1) oracle all tokens, (2) oracle entity-only, (3) system-generated all tokens, and (4) system-generated entity-only. When the entity-only approach is taken, the recall improves significantly in both oracle and system-generated tests. The merge algorithm guarantees that the original token sequence can be rebuilt from the CoNLL 2007 dependency representation, so there are no negative effects to the overall processing pipeline for entity-only parsing.

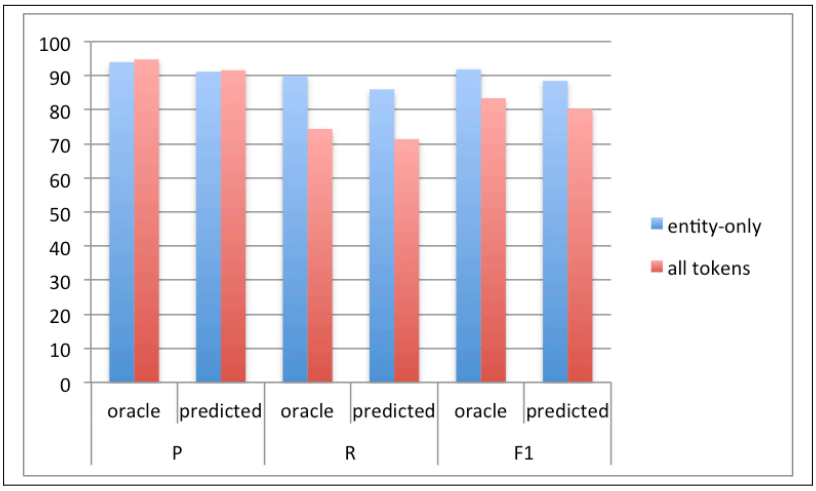

Figure 4.18: A comparison of dependency parsing approaches, all tokens versus entity-only

4.5.4 Pneumonia report classification final results

In this section, I present the final performance results for integrating event-based features into pneumonia report classification experiments. The predicted snippets referred to below are generated by the snippet prediction module, the whole document represents the text of the entire X-ray report from the pneumonia report classification chest X-ray corpus, and the oracle snippets are the annotated unique snippets traced back to the original report files in which they were originally identified—only text from the oracle snippets in a report (there can be more than one) are used for the classification task.

Table 4.32 represents the results of extracting event-only features from the whole document, predicted, and oracle snippets, while Table 4.33 lists the results where all features were used. Tables 4.34-4.36 list all summary results by whole document, predicted, and oracle snippet.

	Text Boundary	N	TP	FP	FN	P	R	F1	Acc
CPIS	Whole	0	1143	198	198	76.2	68.9	71.6	85.2
		250	1162	179	179	79.6	70.4	73.0	86.7
	Predicted Snippets	0	1152	189	189	77.6	70.4	72.7	85.9
		250	1151	190	190	79.7	67.7	71.8	85.8
	Oracle Snippets	0	1205	136	136	85.0	78.4	81.0	89.9
		250	1203	138	138	85.4	77.8	81.0	89.7
PNA	Whole	0	1049	294	294	71.0	67.5	68.9	78.1
		250	1081	262	262	73.2	69.2	70.2	80.5
	Predicted Snippets	0	1075	268	268	74.1	70.0	71.5	80.0
		250	1093	250	250	75.2	71.0	72.2	81.4
	Oracle Snippets	0	1096	247	247	74.5	72.2	73.2	81.6
		250	1126	217	217	77.6	75.8	76.6	83.8

Table 4.32: Final pneumonia report classification results with **event-only** features.

	Text Boundary	N	TP	FP	FN	P	R	F1	Acc
CPIS	Whole	0	1150	191	191	76.7	70.3	72.8	85.8
		250	1175	166	166	79.7	75.5	77.2	87.6
	Predicted Snippets	0	1159	182	182	78.2	71.8	74.1	86.4
		250	1172	169	169	80.9	73.3	76.1	87.4
	Oracle Snippets	0	1225	116	116	87.1	81.8	84.0	91.3
		250	1225	116	116	87.0	82.0	84.2	91.3
PNA	Whole	0	1057	286	286	71.3	69.3	70.1	78.7
		250	1093	250	250	75.1	72.7	73.5	81.4
	Predicted Snippets	0	1077	266	266	73.6	70.9	71.8	80.2
		250	1099	244	244	75.6	73.2	73.8	81.8
	Oracle Snippets	0	1114	229	229	75.9	74.8	75.3	82.9
		250	1123	220	220	76.8	75.5	75.9	83.6

Table 4.33: Final pneumonia report classification results with **all** features.

Table 4.34 compares the final results for whole document feature extraction with the baseline systems, System B, with feature selection, and System B*, with feature selection $\theta = 250$. The best F-score performance values for both CPIS and PNA are highlighted in bold.

Text Boundary	Type	System	Features	θ	P	R	F1	Acc
Whole Document	CPIS	B	baseline	0	74.3	68.9	71.3	84.6
			event-only	0	76.2	68.9	71.6	85.2
			all features	0	76.7	70.3	72.8	85.8
		B*	baseline	250	77.0	71.1	73.4	86.0
			event-only	250	79.6	70.4	73.0	86.7
			all features	250	79.7	75.5	**77.2**	87.6
	PNA	B	baseline	0	70.1	68.8	69.3	80.0
			event-only	0	71.0	67.5	68.9	78.1
			all features	0	71.3	69.3	70.1	78.7
		B*	baseline	250	74.1	71.7	72.5	80.8
			event-only	250	73.2	69.2	70.2	80.5
			all features	250	75.1	72.7	**73.5**	81.4

Table 4.34: Comparison of the performance of baseline systems with feature selection (System B*) and without (System B) and final system with event-only or all feature sets over the **whole document**

Table 4.35 compares the final results for predicted snippet feature extraction with the baseline systems, System B, no feature selection, and System B*, with feature selection $\theta = 250$. The best F-score performance values for both CPIS and PNA are highlighted in bold.

Text Boundary	Type	System	Features	θ	P	R	F1	Acc
Predicted snippets	CPIS	B	baseline	0	76.9	72.4	74.2	86.1
			event-only	0	77.6	70.4	72.7	85.9
			all features	0	78.2	71.8	74.1	86.4
		B*	baseline	250	77.6	71.1	73.4	86.1
			event-only	250	79.7	67.7	71.8	85.8
			all features	250	80.9	73.3	**76.1**	87.4
	PNA	B	baseline	0	74.4	72.2	72.4	81.4
			event-only	0	74.1	70.0	71.5	80.0
			all features	0	73.6	70.9	71.8	80.2
		B*	baseline	250	77.5	74.4	**75.1**	83.3
			event-only	250	75.2	71.0	72.2	81.4
			all features	250	75.6	73.2	73.8	81.8

Table 4.35: Comparison of the performance of baseline systems with feature selection (System B*) and without (System B) and final system with event-only or all feature sets over the **predicted snippets**

Table 4.36 compares the final results for predicted snippet feature extraction with the baseline systems, System B, no feature selection, and System B*, with feature selection $\theta = 250$. The best F-score performance values for both CPIS and PNA are highlighted in bold.

Text Boundary	Type	System	Features	θ	P	R	F1	Acc
Oracle snippets	CPIS	B	baseline	0	85.6	81.0	83.0	90.8
			event-only	0	85.0	78.4	81.0	89.9
			all features	0	87.1	81.8	84.0	91.3
		B*	baseline	250	87.3	81.3	83.4	91.3
			event-only	250	85.4	77.8	81.0	89.7
			all features	250	87.0	82.0	**84.2**	91.3
	PNA	B	baseline	0	78.6	77.1	77.6	84.8
			event-only	0	74.5	72.2	73.2	81.6
			all features	0	75.9	74.8	75.3	82.9
		B*	baseline	250	79.3	77.8	**78.3**	85.4
			event-only	250	77.6	75.8	76.6	83.8
			all features	250	76.8	75.5	75.9	83.6

Table 4.36: Comparison of the performance of baseline systems with feature selection (System B*) and without (System B) and final system with event-only or all feature sets over **oracle snippets**

The results are mixed across all three tables, for CPIS and PNA. The highlighted F-score values in Table 4.34 for the whole document are higher than the predicted snippet results, and indicate that when event-based features are added to the feature set for whole documents, especially when feature selection is used, there can be a significant performance improvement over baseline. For CPIS classification, System B* had an F-score of 77.2 for *all features with feature selection* compared to the baseline System B *no feature selection* which had an F-score of 71.3 and for PNA classification, System B* had an F-score of 73.5 for *all features with feature selection* compared to the baseline System B *no feature selection* which had an F-score of 69.3. However, in all three tables, event-features alone never outperform unigram-based features, and it is clear that feature selection is the main catalyst to improved performance. In all three tables, regardless of feature set, feature selection is seen to improve

performance, if even only marginally in the case of oracle snippets.

4.5.4.1 Feature threshold experiments

In this section, I explore the effect of feature selection thresholds on the performance of the pneumonia report classifier for both the CPIS and PNA categories. Three sets of experiments for unit of classification: (1) whole document, (2) oracle snippet, and (3) predicted snippet were run. Whole document results are listed and discussed in this section and oracle and predicted snippets results are listed in Appendix E. The whole document experiments were further grouped into sets for baseline features, event-only features, and all features. Each set based on feature type included one or more experiments for a specific feature threshold. The same increments of feature threshold were applied across all experiment sets. The standard feature threshold increments for CPIS and PNA labels on whole documents were: 50, 100, 150, 200, 250, 500, 750, 1000, 1500, and 2500. Three other factors influenced the number of feature threshold experiments run for each experiment set: (1) if the maximum number of χ^2 significant features was less than one of the standard feature threshold increments, experiments were not run for thresholds greater than the maximum (e.g., there are only 519 significant features for CPIS experiments on whole documents, so the maximum feature threshold experiment for this set was 750), (2) if the maximum number of average features per fold in the MaxEnt model with no feature selection was less than a feature threshold, no experiments are run for feature threshold increments with a greater value, and (3) the F-scores for the larger feature threshold increments demonstrate a pattern of diminishing performance. The row at the top of each table with a hyphen as its value for θ represents an experiment with no feature selection.

The series of Tables 4.37 - 4.49 list the results of pneumonia report classification experiments for the CPIS category on whole documents for baseline, event-only, and all features. The Table 4.40 lists the highest performing experiment across the three feature sets.

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1134	207	207	74.3	68.9	71.3	84.6
750	1341	1341	1153	188	188	76.3	70.3	72.4	86.0
500	1341	1341	1144	197	197	74.8	69.1	71.0	85.3
250	1341	1341	1153	188	188	77.0	71.1	73.4	86.0
200	1341	1341	1163	178	178	78.5	72.4	74.7	86.7
150	1341	1341	1171	170	170	79.9	72.6	75.2	87.3
100	1341	1341	1168	173	173	79.2	72.2	74.7	87.1
50	1341	1341	1148	193	193	74.4	70.0	71.2	85.6

Table 4.37: CPIS feature threshold experiments, **baseline** features on **whole document** (Average number of features across folds in model with no feature selection = 1653/Number of significant χ^2 ranked features = 510)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1143	198	198	76.2	68.9	71.6	85.2
2500	1341	1341	1150	191	191	80.1	66.0	69.3	85.8
1500	1341	1341	1154	187	187	78.7	68.6	70.8	86.1
1000	1341	1341	1153	188	188	77.8	69.1	71.2	86.0
750	1341	1341	1151	190	190	76.8	69.8	71.8	85.8
500	1341	1341	1156	185	185	77.3	71.2	73.1	86.2
250	1341	1341	1162	179	179	79.7	70.2	72.9	86.7
200	1341	1341	1164	177	177	79.4	71.4	73.8	86.8
150	1341	1341	1163	178	178	78.5	72.4	74.3	86.7
100	1341	1341	1159	182	182	78.5	71.8	74.2	86.4
50	1341	1341	1111	230	230	73.2	66.5	67.9	82.8

Table 4.38: CPIS feature threshold experiments, **event-based** features on **whole document** (Average number of features across folds in model with no feature selection = 10,544/Number of significant χ^2 ranked features = 2500)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1150	191	191	76.7	70.3	72.8	85.8
2500	1341	1341	1165	176	176	80.8	69.2	72.2	86.9
1500	1341	1341	1166	175	175	79.5	70.6	73.4	87.0
1000	1341	1341	1168	173	173	79.6	71.7	74.4	87.1
750	1341	1341	1165	176	176	78.8	72.0	74.6	86.9
500	1341	1341	1174	167	167	79.4	74.5	76.4	87.5
250	1341	1341	1175	166	166	79.7	75.5	77.2	87.6
200	1341	1341	1172	169	169	79.0	74.9	76.5	87.4
150	1341	1341	1167	174	174	78.0	73.0	74.7	87.0
100	1341	1341	1167	174	174	78.0	73.2	74.7	87.0
50	1341	1341	1167	174	174	78.1	72.2	74.0	87.0

Table 4.39: CPIS feature threshold experiments, **all** features on **whole document** (Average number of features across folds in model with no feature selection =12,197 /Number of significant χ^2 ranked features = 2500)

Features	θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
baseline	150	1341	1341	1171	170	170	79.9	72.6	75.2	87.3
event-only	150	1341	1341	1163	178	178	78.5	72.4	74.3	86.7
all	250	1341	1341	1175	166	166	79.7	75.5	77.2	87.6

Table 4.40: The highest performing feature threshold experiments for the CPIS category

The feature threshold experiments for the CPIS category, summarized in Table 4.40, demonstrate a smaller margin of improvement of performance than the final classification experiments summarized in Section 4.5.4. The best **baseline** features experiment, with θ=150, had an F-score of 75.2 and the best **all** features experiment, with θ=250, had an F-score of 77.2. The difference between them was 2.0, with all features outperforming baseline features. The final classification experiments' best baseline experiment, with θ=250, had an

F-score of 73.4 and the best all features experiment, with θ=250, had an F-score of 77.2. The difference between them was 3.8, with all features outperforming baseline features. Furthermore, the difference in true positives was only 4 reports, which explains the .3 difference in accuracy, which for multi-category classification is the same as the micro-averaged F-score.

The series of Tables 4.41 - 4.49 list the results of pneumonia report classification experiments for the PNA category on whole documents for baseline, event-only, and all features. Table 4.44 lists the highest performing experiment across the three feature sets.

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1047	296	296	70.1	68.8	69.3	78.0
500	1343	1343	1081	262	262	73.5	71.4	72.1	80.5
250	1343	1343	1085	258	258	74.1	71.7	72.5	80.8
200	1343	1343	1088	255	255	74.0	72.1	72.6	81.0
150	1343	1343	1100	243	243	75.6	73.1	**73.8**	81.9
100	1343	1343	1097	246	246	75.0	72.1	73.0	81.7
50	1343	1343	1090	253	253	73.8	70.0	71.1	81.2

Table 4.41: PNA feature threshold experiments, **baseline** features on **whole document** (Average number of features across folds in model with no feature selection = 1653/Number of significant χ^2 ranked features = 306)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1049	294	294	71.0	67.5	68.9	78.1
1000	1343	1343	1097	246	246	76.0	72.5	**73.6**	81.7
750	1343	1343	1090	253	253	75.1	72.1	73.0	81.2
500	1343	1343	1095	248	248	75.4	72.3	73.2	81.5
250	1343	1343	1081	262	262	73.2	69.2	70.2	80.5
200	1343	1343	1085	258	258	74.6	70.5	71.7	80.8
150	1343	1343	1079	264	264	73.7	69.6	70.7	80.3
100	1343	1343	1082	261	261	74.6	69.9	71.1	80.6
50	1343	1343	1078	265	265	74.4	68.5	70.2	80.3

Table 4.42: PNA feature threshold experiments, **event-based** features on **whole document** (Average number of features across folds in model with no feature selection = 10,584/Number of significant χ^2 ranked features = 928)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1057	286	286	71.3	69.3	70.1	78.7
1500	1343	1343	1082	261	261	74.4	72.6	73.3	80.6
1000	1343	1343	1084	259	259	74.1	72.6	73.1	80.7
750	1343	1343	1086	257	257	74.6	72.7	73.5	80.9
500	1343	1343	1084	259	259	74.5	72.5	73.3	80.7
250	1343	1343	1093	250	250	75.1	72.7	73.5	81.4
200	1343	1343	1100	243	243	76.0	73.4	**74.3**	81.9
150	1343	1343	1095	248	248	74.9	72.2	72.8	81.5
100	1343	1343	1089	254	254	74.3	70.7	71.7	81.1
50	1343	1343	1079	264	264	71.8	67.2	68.8	80.3

Table 4.43: PNA feature threshold experiments, **all** features on **whole document** (Average number of features across folds in model with no feature selection = 12,238/Number of significant χ^2 ranked features = 1234)

Features	θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
baseline	150	1343	1343	1100	243	243	75.6	73.1	73.8	81.9
event-only	1000	1343	1343	1097	246	246	76.0	72.5	73.6	81.7
all	200	1343	1343	1100	243	243	76.0	73.4	**74.3**	81.9

Table 4.44: The highest performing feature threshold experiments for the PNA category

The feature threshold experiments for the PNA category (see Table 4.44) demonstrate a smaller margin of improvement of F-score performance than the final classification experiments summarized in Section 4.5.4. The best **baseline** features experiment, with θ=150, had an F-score of 73.8, and the best **all** features experiment, with θ=200, had an F-score of 74.3. The difference between them was only .5, with all features outperforming the baseline features. The final experiments' baseline features experiment, with θ=250, had an F-score of 72.5 and the best all features experiment, with θ=250 had an F-score of 73.5. There is only a difference of 1.0 between them. The difference in true positives is 0 reports, which explains the exact same accuracy score of 81.9.

4.6 Discussion

In this section I discuss results and observations of the different phases of my research framework and follow the same section heading and organization structure for discussion as my methodology and results sections:

1) **Event analysis**—analyze the task and related previous research to define events

2) **Corpus development**—develop corpora by annotating events

3) **Event detection**—build systems for automatic event detection

4) **Event-based feature extraction**—extract event-based features for classification

4.6.1 Event analysis

In the event analysis phase, I replicated the rationale snippet prediction module and pneumonia report classification experiments described in Tepper et al. (2013). In this section, I discuss the results of the snippet prediction module evaluation and replicated baseline feature experiments.

4.6.1.1 Snippet prediction module evaluation

My snippet prediction module's design differs Tepper et al. (2013) in three ways:

Classification model Tepper et al. (2013) achieved their best snippet prediction performance for CPIS and PNA with two different approaches. The two approaches were: (1) a MaxEnt model for binary classification and a beam search for sequence decoding , and (2) a *no prevTag* setting for the MaxEnt model trainer which excludes the previous sentence label tag from the feature set and separates the beam search sequence decoding from the binary sentence classifier. (1) performs best for the PNA class (but does so marginally, 72.5 F-score over 70.4 F-score) and has no positive effect on the CPIS class. (2) performs best for CPIS (93.9 F-score for snippet overlap) but not PNA. Given the performance difference between CPIS and PNA, and the small difference between PNA with and without *prevTag*, for simplicity, I implement a MaxEnt binary classifier (*inside* and *outside* snippet sentence categories) with no beam search. The choice to constrain snippet prediction to a binary classification task has a minimal effect on the performance of both PNA and CPIS snippet prediction in my experiments.

Evaluation measure The snippet overlap and sentence overlap measures in the original study are not a strict measure of sentence to sentence matches between the system and gold standard labels. I implemented a stricter, but simpler, binary classification evaluation per sentence for my snippet prediction performance. The performance of my snippet predictor using my measure is similar to the sentence overlap reported in

the original study, F-score of 88.3 vs 88.4 for CPIS, and an F-score of 70.2 vs 70.4 for PNA, but cannot be compared directly because it is based on exact label matches per sentence and not character offset overlap. See Table 4.14 for original sentence overlap results from Tepper et al. (2013), and Table 4.15 for performance results of the replicated snippet prediction module using the alternate sentence label to sentence label evaluation measure. In the design of my experiments, my goal was to measure the impact of event-based features on pneumonia report classification. Similar, but not exactly the same performance of the snippet prediction module in my evaluation versus that published in Tepper et al. (2013) did not effect the comparison of the overall patterns of performance between baseline and event-based feature selection in the final set of pneumonia report classification experiments.

Training folds In the original study, the snippet prediction model was trained on folds comprised only of reports with snippets, and tested on testing folds that included all reports. The folds that did not include a snippet are considered in the negative class automatically, and they are distributed evenly amongst the test folds. The approach was taken to reduce the imbalance of positive to negative sentences and improve recall. No performance results were provided in the original study to show the differences between these two approaches to training. I used both strategies to train and evaluate models for training. I compared the results in Table 4.15. After this comparison, the approach chosen by Tepper et al. (2013), training with reports with snippets-only, was proven to improve recall and resulted in higher classification results (see a comparison of the training folds' configuration on classification results in Table 4.16).

4.6.1.2 Replicated baseline experiments

The two baseline systems, System B and System B*, listed in the result tables in Section 4.5.1.2 do not match the performance of Tepper et al. (2013)'s System A in Table 4.14, however the numbers do follow a similar pattern, as seen in the charts in Figures 4.16 and

4.17. I chose to use System B and B* as my baseline results, to compare with final pneumonia report classification experimental results, because they demonstrated a similar performance pattern as System A in oracle, predicted, and whole document scenarios. The main goal of the case study was to measure the impact of event-based features on performance of pneumonia report classification in various configurations not to match the results of the original system in Tepper et al. (2013).

4.6.2 Corpus development

The results of the IAA measures during the first stage of annotation helped clarify basic rules for annotating the `Cos` entities and revealed the task as fairly easy for the annotators. In particular, discussions about how to mark up `Val`, where each token is labeled as a individual entity, and `Loc`, where multiple tokens are labeled with a single span, helped to improve IAA scores in the second round of annotation.

The second stage of annotation, after extensions were made to the schema for coordination and negation, was considerably more complicated. Additional rounds of IAA could have helped clarify model issues earlier on in the second stage, rather than relying solely on a single NLP researcher to apply all of the negation, coordination, and diagnosis event labeling. The single second stage annotator made several passes of annotation editing and revising the unique rationale snippet corpus, without the benefit of an IAA review or discussion with fellow annotators.

Maintaining the single-headed constraint of a dependency tree in the second stage of annotation was not supported by the annotation environment and had to be maintained by the human annotator. There were many cases where the scope of coordination and negation were interpreted by the annotator based on the context of the surrounding sentence, and not on a strict set of rules. In a small number of cases, where the dependency tree graph may have been constructed in a way that would lead to non-projectivity, the human annotator preferred a projective analysis to ensure that as many annotations in the corpus conformed to the constraints of a projective dependency tree. There remain a number of non-projective

dependency trees within the unique snippet corpus. See Figure 4.19 for an example.

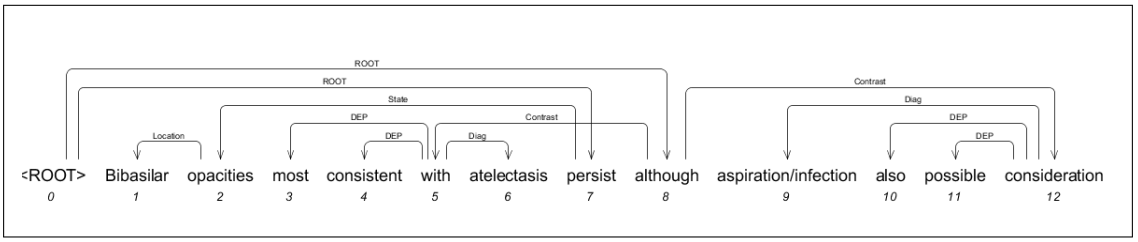

Figure 4.19: An example of a non-projective dependency in the unique snippet corpus

In the figure above, Figure 4.19, there are two reasons for non-projective arcs to exist in the dependency tree. The first reason is a symptom of the complexity of natural language itself. There is a change-of-state event *bipolar opacities persist* with head *persist*, and two diagnosis events, *most consistent with atelectasis*, headed by *with*, and *aspiration/infection also possible consideration*, with head *consideration*. The two diagnosis events are coordinated by the conjunction *although*. The projections for the events cross, causing non-projective arcs. The second reason is that the special node, `ROOT`, which provides the root of dependency tree itself, needs to be linearized when added to the tree and its arc crosses *into* the events to connect to their heads. Both reasons for non-projection are acceptable based on the constraints of non-projective dependency trees described in Nivre (2006).

4.6.3 Event detection

Both NER and RE with dependency parsing performed well in evaluation. The two-stage approach was simple and easy to implement with off-the-shelf tools, given the dependency tree constraints applied during the second stage of annotation. Without dependency constraints, the graph is much less predictable in its shape, and is more difficult to parse with a general solution, as was the case after the first stage of annotation, where the change-of-state event was represented simply as a DAG.

Parsing entity-only token strings in the dependency parser, ignoring non-entity labeled

tokens and punctuation, proved a good strategy, improving overall F-score, especially recall, for both oracle and system generated named entity configurations. Tracking the token identity and order in the original token sequence in the feature column (column 6) of the CoNLL 2007 dependency format allowed the original token sequence to be reconstructed or merged with the canonical gold standard trees for evaluation.

An analysis of the projectivity of the event trees in the unique snippet corpus and running experiments with the covington non-projective algorithm enabled in the Malt parser might improve the performance of the dependency parsing phase of the two-stage event detection process. In a preliminary analysis, approximately 13% of the arcs in the corpus are currently non-projective. Many of these arcs are `ROOT` arcs that are added by transformation to the CoNLL 2007 format, but a number of examples, like Figure 4.19, contain non-projective arcs for actual change-of-state and diagnosis events.

4.6.4 Event-based feature extraction

Table 4.45 compares the top 10 features selected by χ^2 ranking for *baseline* and *all feature* sets in the pneumonia report classification experiments configured for the whole document. The list represents only a small number of the total features selected in experiments, where the feature selection threshold is set to 250, but they do represent the highest ranking features by χ^2 value. The word unigram values in the baseline feature list, in the left column, are also found as the main value in the event-based feature constructions for the *all features* set, in the right column. For example, the clinical observation construct, `OBS→OPACITY`, and clinical attribute value pair, `ATTR=OPACITY`, rank above, but are clustered with, the word unigram `OPACITY`, which is the highest ranking word unigram in the baseline feature set. The same pattern repeats with the second ranking value in the baseline feature set, `ATELECTASIS`. Similar to `OPACITY`, it also is found as the value of a cluster of event-based feature constructions in the *all features* set. The pattern suggests that a number of event-based features are simply reinforcing the effect of existing n-gram features in the MaxEnt model, weighted by their distribution in corpus. Simple event features, such as entity value

pairs, do not add any novel predictive value to the experiments, but rather help to boost the weights of existing n-gram features. The more structured event features, such as a change-of-state and diagnosis event tuple with all fields populated, occur more infrequently than the less structured event features, but contribute more information as a feature.

Baseline Feature Set	All Feature Sets
OPACITY	OBS→OPACITY
ATELECTASIS	ATTR=OPACITY
OPACITIES	OPACITY
LOBE	ATELECTASIS
CONSOLIDATION	COS→OPACITY
DIFFUSE	OBS→ATELECTASIS
ASPIRATION	ATTR=ATELECTASIS
UPPER	OPACITIES
ABNORMALITIES	LOBE
TUBES	ATTR=CONSOLIDATION

Table 4.45: Top 10 features per feature set used in CPIS whole document experiments

A closer look at the 250 features selected in experiments for the *all features* feature set in whole document, predicted snippet, and oracle snippet configurations reveals that some of the more structural event-based features, such as a generalized change-of-state with attached value prefix and location suffix, do not rank high in the overall feature set. Table 4.46 lists a selection of ranked event-based features from the 250 all features set. The first structured event-based feature occurs at position 20 for CPIS and the first tuple with more than two slots occurs at rank 87.

Rank	Feature
20	EV_COS=NEG_COS→FOCAL_ABNORMALITIES
...	
55	EV_OBS=OBS→OPACITY_@_RIGHT_LOWER_LOBE
...	
81	EV_COS=NEG_COS->DIFFUSE_OPACITIES_@_LUNG
...	
87	EV_TP=[CHANGE_NEG,OPACITIES,DIFFUSE,LUNG,-]

Table 4.46: Ranked event features from combined all features feature set

The size of the overall dataset and the sparsity of event-based features in the χ^2 ranking is a limitation when determining if the features benefit an applied clinical NLP classification task. Expanding the corpus with additional reports and annotations could contribute more event-based structured features to the feature ranking and selection process. Expanding the scope of annotation also addresses the assumption that change-of-state and diagnosis events are unique to rationale snippets. Whole document experiments show that in reports with no snippets, which are automatically labeled negative in the snippet-only experiments, change-of-state and diagnosis events can be extracted. Expanding the annotation of change-of-state and diagnosis events to address the entire report, not only the rationale snippets, could contribute more to the negative class models for CPIS.

4.6.5 Pneumonia report classification error analysis

The distribution of reports across CPIS and PNA labels in the pneumonia report classification corpus is imbalanced as discussed in Tepper et al. (2013). Table 4.47 illustrates the characteristics of each category. The number of reports for the negative class *1A no infiltrate* of CPIS is small compared to the positive classes and only contributes 25 reports with rationale snippets to the model. Of the two positive classes, the number of reports for the *1B diffuse infiltrate or atelectasis* label greatly outnumbers *1C local infiltrate*. Many of

the signature features of the dominant class *1B diffuse infiltrate or atelectasis*, such as the ngrams *atelectasis* and *opacity*, are also words that occur frequently in reports labeled *1C local infiltrate*. This is a source of ambiguity between the two positive classes and is reflected in the poor results for label *1C local infiltrate*, as seen in examples of the detailed results of whole document CPIS experiments in Table 4.48 and Table 4.49. There is a similar imbalance in the distribution of reports across PNA labels, but instead of an under-represented negative class, the PNA negative class is the largest of the three label classes and has the most reports that contain a rationale snippet.

Label	**Category**	**Number (%) of reports**	**Number (%) of reports with rationale snippets**
CPIS	1A no infiltrate	178 (13%)	25 (2%)
	1B diffuse infiltrate...	1011 (75%)	1004 (75%)
	1C local infiltrate	152 (11%)	152 (11%)
	Total	1341 (100%)	1181 (88%)
PNA	2A no suspicion PNA	856 (64%)	362 (27%)
	2B suspicion of PNA	294 (22%)	290 (22%)
	2C probable PNA	193 (14%)	192 (14%)
	Total	1343 (100%)	844 (63%)

Table 4.47: Statistics of the X-ray report corpus as reported in Tepper et al. (2013)

4.6.5.1 Classification performance across category labels

When event-based features are added to the baseline feature set in whole document experiments for CPIS, there is a slight improvement in F-score across labels as can be seen in the comparison of results in Table 4.48 and Table 4.49. However, where labels *1A no infiltrate* and *1B diffuse infiltrate or atelectasis* see an increase in true positives, label *1C local infiltrate* sees its improvement mostly from a decrease in false positives. The pattern of low F-scores

for label *1C local infiltrate* repeats itself in all detailed experiment results for CPIS.

CPIS Label	**TP**	**FP**	**FN**	**P**	**R**	**F1**
1A	132	26	46	83.5	74.2	78.6
1B	942	123	69	88.5	93.2	90.8
1C	60	58	92	50.8	39.5	44.4
Total	1134	207	207	74.3	68.9	71.3

Table 4.48: Detailed results of CPIS whole document experiment with baseline features and no feature selection

CPIS Label	**TP**	**FP**	**FN**	**P**	**R**	**F1**
1A	140	29	38	82.8	78.7	80.7
1B	952	121	59	88.7	94.2	91.4
1C	58	41	94	58.6	38.2	46.2
Total	1150	191	191	76.7	70.3	72.8

Table 4.49: Detailed results of CPIS whole document experiment with all features and no feature selection

The same tables from the PNA experiment sets shows a different imbalance in the corpus. See Table 4.50 and Table 4.51 below and compare with the results in Table 4.48 and Table 4.49

PNA Label	TP	FP	FN	P	R	F1
2A	781	124	75	86.3	91.2	88.7
2B	127	118	167	51.8	43.2	47.1
2C	139	54	54	72.0	72.0	72.0
Total	1047	296	296	70.1	68.8	69.3

Table 4.50: Detailed results of PNA whole document experiment with baseline features and no feature selection

PNA Label	TP	FP	FN	P	R	F1
2A	791	128	65	86.1	92.4	89.1
2B	125	110	169	53.2	42.5	47.3
2C	141	48	52	74.6	73.1	73.8
Total	1057	286	286	71.3	69.3	70.1

Table 4.51: Detailed results of PNA whole document experiment with all features and no feature selection

The distribution of reports across labels for PNA, similar to CPIS, is also imbalanced based on the negative class, but for a different reason. For PNA, the largest, dominant class is the negative class *2A no suspicion of pneumonia*. Again, similar to CPIS, it is one of the positive classes, *2B suspicion of pneumonia* that has the poorest classification performance.

4.6.5.2 Examples of the negative class

How the CPIS and PNA models classify individual instances of the negative class based on their imbalanced distribution is almost exactly opposite. `Report_11` is an example of a report that is labeled as the negative class for both PNA and CPIS categories.

For CPIS, the label applied to `Report_11` by the medical expert annotators during the

annotation phase is the negative class *1A no infiltrate*. The snippet they select as rationale for the label is *Lung volumes are lower on today's examination increasing perihilar haziness and vascular congestion which may reflect volume status.* Without specific medical knowledge and access to the remainder of the report text, this appears to be an ambiguous snippet given its assigned category. Phrases such as *lung volumes are lower*, *increasing perihilar haziness*, and *vascular congestion*, especially when decomposed into n-gram features suggest one of the two positive labels rather than the negative class. In fact, in baseline feature experiments using only the report's oracle snippet to classify and no feature selection, the model ranks label probability as: *1B* 35.7%, *1C* 51.2%, and *1A* 13.1%. With the same settings and the addition of event-features the classifier again does not assign the correct label probabilities: *1B* 69.7%, *1C* 24.4%, and *1A* 5.9%.

Predicted snippet experiments assign the correct label to *Report_11*, however, not due to model features. The snippet prediction module does not predict a rationale snippet for the report and it is assigned the negative class by default in experiments.

The only experimental settings for the classifier that correctly predict the label for *Report_11*, are whole document experiments with all features, both baseline and event-based, without feature selection. The model generates the following label probabilities for the report: *1A* 70.8%, *1B* 29.0%, and *1C* .2%. Extracting additional events and event-based features from the complete report text, not just the rationale snippet, results in the correct label for `Report_11`. The number of whole document event-based features generated for `Report_11` exceeds the number of n-gram features generated, and in this specific example, produces the correct result. Experiments using feature selection, for example, CPIS with a 250 feature threshold on whole document with all features predicts the incorrect label probabilities: *1B* 52.9%, *1A* 39.9%, and *1C* 7.2%.

The PNA category is more balanced than CPIS and has a much larger number of negative class reports that contain one or more rationale snippets (362). A PNA baseline feature experiment with oracle snippets and no feature threshold results in the correct label for `Report_11`, *2A no suspicion of pneumonia* 57.4%, *2B suspicion of pneumonia* 40.3%, and

2C probable pneumonia 2.2%. Using the same oracle and baseline settings with no feature selection and the addition of event-based features, the classifier performs even better *2A no suspicion of pneumonia* 71.0%, *2B suspicion of pneumonia* 27.2%, and *2C probable pneumonia* 1.8%. The strongest support for the correct label comes from the PNA whole document experiments with all features and no feature selection, *2A no suspicion of pneumonia* 85.4%, *2B suspicion of pneumonia* 14.2%, and *2C probable pneumonia* .4%. Like CPIS, no rationale snippet is predicted when the snippet prediction module is run on `Report_11`, so it is automatically assigned the negative class in the classification experiments for predicted snippets.

4.6.5.3 MaxEnt model feature ranking vs χ^2 ranking

I used χ^2 for statistical feature selection in my experiments. The value of χ^2 for each feature in all feature types was calculated in a preprocessing stage. Significant features, those whose χ^2 value was greater than the critical value for $p = .05$ and 2 degrees of freedom, 5.99, were ranked and stored in the system database. These values were calculated for each feature globally across the entire corpus. The MaxEnt models I trained with MALLET in a 5-fold configuration for cross-validation, rank and weight features based on their distribution across training folds. I average the feature weights across folds and only rank the features that occur in all five test folds. Table 4.52, 4.53, 4.54, and 4.55 list the top 10 χ^2 ranked features and top 10 MaxEnt model ranked features for CPIS and PNA in order to evaluate whether the χ^2 rankings were similar to the MaxEnt model rankings, and how event-based features ranked when combined with baseline features.

MaxEnt ranked (label)	χ^2 **ranked**
U_O_N_v_ATELECTASIS (1B)	U_O_N_v_OPACITY
U_O_N_v_DIFFUSE (1B)	U_O_N_v_ATELECTASIS
U_O_N_v_CONSOLIDATION (1C)	U_O_N_v_OPACITIES
U_O_N_v_OPACITY (1C)	U_O_N_v_LOBE
U_O_N_v_PNEUMONIA (1C)	U_O_N_v_CONSOLIDATION
U_O_N_v_BIBASILAR (1B)	U_O_N_v_DIFFUSE
U_O_N_v_OPACITIES (1B)	U_O_N_v_ASPIRATION
U_O_N_v_PATCHY (1B)	U_O_N_v_UPPER
U_O_N_v_BILATERAL (1B)	U_O_N_v_ABNORMALITIES
U_O_N_v_INFECTION (1C)	U_O_N_v_TUBES

Table 4.52: Top 10 **CPIS** MaxEnt model ranked **baseline** features (by label) compared with Top 10 χ^2 ranked **baseline** features

MaxEnt ranked (label)	χ^2 **ranked**
U_O_N_v_ATELECTASIS (1B)	EV_OBS=OBS→OPACITY
U_O_N_v_BILATERAL (1B)	EV_EN_ATTR=OPACITY
U_O_N_v_DIFFUSE (1B)	U_O_N_v_OPACITY
EV_OBS=OBS→ATELECTASIS (1B)	U_O_N_v_ATELECTASIS
EV_EN_ATTR=ATELECTASIS (1B)	EV_COS=COS→OPACITY
U_O_N_v_OPACITY (1C)	EV_OBS=OBS→ATELECTASIS
EV_EN_VAL=DIFFUSE (1B)	EV_EN_ATTR=ATELECTASIS
U_O_N_v_PNEUMONIA (1C)	U_O_N_v_OPACITIES
U_O_N_v_OPACITIES (1B)	U_O_N_v_LOBE
U_O_N_v_BIBASILAR (1B)	EV_EN_ATTR=CONSOLIDATION

Table 4.53: Top 10 **CPIS** MaxEnt model ranked **baseline** and **event** features (by label) compared with Top 10 χ^2 ranked **baseline** and **event** features

MaxEnt ranked (label)	χ^2 **ranked**
U_O_N_v_PNEUMONIA (2C)	U_O_N_v_PNEUMONIA
U_O_N_v_INFECTION (2C)	U_O_N_v_INFECTION
U_O_N_v_CONSOLIDATION (2B)	U_O_N_v_ASPIRATION
U_O_N_v_ASPIRATION (2B)	U_O_N_v_EXPECTED
U_O_N_v_INFECTION (2B)	U_O_N_v_CONSOLIDATION
U_O_N_v_PNEUMONIA (2B)	U_O_N_v_PATCHY
U_O_N_v_EXTUBATED (2B)	U_O_N_v_SHOW
U_O_N_v_CLEAR (2A)	U_O_N_v_ABNORMALITIES
U_O_N_v_CATHETER (2A)	U_O_N_v_FOCAL
U_O_N_v_OPACIFICATION (2B)	U_O_N_v_MAY

Table 4.54: Top 10 **PNA** MaxEnt model ranked **baseline** features (by label) compared with Top 10 χ^2 ranked **baseline** features

MaxEnt ranked (label)	χ^2 **ranked**
U_O_N_v_PNEUMONIA (2C)	U_O_N_v_PNEUMONIA
EV_DIAG=DIAGNOSIS→PNEUMONIA (2C)	EV_OBS=OBS→ATELECTASIS/PNEUMONIA
EV_OBS=OBS→PNEUMONIA (2C)	EV_OBS=OBS→PATCHY_ATELECTASIS/PNEUMONIA
EV_DIAG=DIAGNOSIS→INFECTION (2C)	EV_OBS=OBS→PNEUMONIA
EV_EN_ATTR=PNEUMONIA (2C)	EV_EN_ATTR=PNEUMONIA
U_O_N_v_INFECTION (2C)	EV_DIAG=DIAGNOSIS→PNEUMONIA
U_O_N_v_PNEUMONIA (2B)	EV_TP=[-,EXPECTED,-,-,-]
U_O_N_v_INFECTION (2B)	EV_EN_ATTR=EDEMA/PNEUMONIA
EV_OBS=OBS→INFECTION (2C)	EV_DIAG=DIAGNOSIS→EDEMA/PNEUMONIA
EV_EN_ATTR=INFECTION (2C)	EV_OBS=OBS→EDEMA/PNEUMONIA

Table 4.55: Top 10 **PNA** MaxEnt model ranked **baseline** and **event** features (by label) compared with Top 10 χ^2 ranked **baseline** and **event** features

Tables 4.52 and 4.54 compare the list of χ^2 and MaxEnt model ranked features for the baseline n-gram feature set. Although the lists do not match exactly, or rank features in the same order, they do show that similar terms from the domain, such as *atelectasis*, *pneumonia*, *opacities*, and *infection* are shared by both lists. Tables 4.53 and 4.55 compare the lists of χ^2 and MaxEnt model ranked features for all features (baseline n-gram and event-based features combined). Again, the lists do not match exactly, but do demonstrate a common set of terms from the domain. One difference between the χ^2 and MaxEnt ranking for all features is that more event-based features are ranked in the top 10 for χ^2 for both CPIS and PNA labels.

4.6.5.4 Pneumonia report classification errors

In this section, I review the results for the poorest performing labels for both CPIS and PNA, and how event-based features and optimal feature threshold experiments effected their performance. Both labels are one of the two positive classes for their category.

Table 4.56 lists the detailed results for CPIS label *1C local infiltrate* in the best performing optimal feature threshold experiments for whole documents. The addition of event-based features improves the results of the *1C local infiltrate* class, specifically by increasing the number of true positives and reducing the number of false negatives for the class. The confusion tables featured in the series Table 4.57 - 4.59, show the breakdown of how false positives and false negatives breakdown for each row of Table 4.56. In the best performing configuration for CPIS, all features with $\theta = 250$, mislabeling of the poorest performing label, *1C* is reduced.

Features	θ	**TP**	**FP**	**FN**	**P**	**R**	**F1**
baseline	150	60	33	92	64.5	39.5	49.0
event-only	1000	61	31	91	66.3	40.1	50.0
all	200	72	41	80	63.7	47.4	**54.3**

Table 4.56: Performance of CPIS Label *1C local infiltrate* in feature threshold experiments

	1A	**1B**	**1C**
1A	148	27	3
1B	18	963	30
1C	7	85	60

Table 4.57: Confusion matrix for CPIS whole document **baseline** experiments with optimal feature selection threshold of θ=**150**

	1A	**1B**	**1C**
1A	147	29	11
1B	27	955	29
1C	11	80	61

Table 4.58: Confusion matrix for CPIS whole document **event-only** experiments with optimal feature selection threshold of θ=**150**

	1A	**1B**	**1C**
1A	151	23	4
1B	22	952	37
1C	5	75	72

Table 4.59: Confusion matrix for CPIS whole document **all** feature experiments with optimal feature selection threshold of θ=**250**

Table 4.60 lists the detailed results for PNA label *2B suspicion of PNA* in the best performing optimal feature threshold experiments for whole documents. The addition of event-based features marginally improves the results of the *2B suspicion of PNA* class, with a small increase in the number of true positives and small reduction in the number of false negatives for the class. The confusion tables featured in the series Table 4.61 - 4.63, show the breakdown of how false positives and false negatives breakdown for each row of Table 4.60. In the best performing configuration for PNA, all features with $\theta = 200$, mislabeling of the poorest performing label, *2B* is reduced.

Features	θ	**TP**	**FP**	**FN**	**P**	**R**	**F1**
baseline	150	137	79	157	63.4	46.6	53.7
event-only	1000	131	73	163	64.2	44.6	52.6
all	200	139	84	155	62.3	47.3	**53.8**

Table 4.60: Performance of PNA Label *2B suspicion of PNA* in feature threshold experiments

	2A	**2B**	**2C**
2A	813	39	4
2B	113	137	44
2C	3	40	150

Table 4.61: Confusion matrix for PNA whole document **baseline** experiments with optimal feature selection threshold of θ=**150**

	2A	**2B**	**2C**
2A	816	37	3
2B	122	131	41
2C	7	36	150

Table 4.62: Confusion matrix for PNA whole document **event-only** experiments with optimal feature selection threshold of θ=**1000**

	2A	**2B**	**2C**
2A	810	44	2
2B	115	139	40
2C	2	40	151

Table 4.63: Confusion matrix for PNA whole document **all** feature experiments with optimal feature selection threshold of θ=**200**

4.7 Summary

In this study, I tried to answer the question, *do event-based features impact the performance of a clinical NLP disease detection report classification task?*. I selected as a case study pneumonia report classification, and used a pre-existing corpus and the description of an

implemented system (Xia and Yetisgen-Yildiz, 2012; Tepper et al., 2013; Vanderwende et al., 2013) to create baseline results for n-gram feature-based experiments. I introduced a new model for annotating change-of-state and diagnosis events, including entities for negation and coordination, and constrained the model's graph representation to a dependency tree. I annotated a corpus of 1008 unique rationale snippets with a schema based on the change-of-state and diagnosis event event model in the BRAT annotation tool and developed a two-stage event detection system integrating off-the-shelf NER and dependency parsing tools. I introduced six event-based features that I integrated into pneumonia report classification tasks, and measured and compared the performance of a pneumonia report classifier trained on feature sets that included event-based features alone, baseline features alone (word unigram and alternating conjunctions), and a combination of all features. The pneumonia report classification experiments were run for both CPIS and PNA labels.

In conclusion, the pneumonia report classification experiments returned mixed results. CPIS experiments in oracle, predicted, and whole document configurations all improved when event-based features were added to the feature sets. Feature selection and feature selection threshold experiments demonstrated that this pattern held even when optimal threshold values were implemented for each feature set. Table 4.64 lists the best performing configuration of features and optimal feature threshold for each classification configuration: whole document, oracle snippet, and predicted snippet.

Config.	**Features**	θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
oracle	all	500	1341	1341	1228	113	113	87.4	82.2	**84.4**	91.6
predicted	all	150	1341	1341	1175	166	166	80.6	74.3	**76.6**	87.6
whole	all	250	1341	1341	1175	166	166	79.7	75.5	**77.2**	87.6

Table 4.64: The highest performing experiments for the CPIS category in oracle snippet, predicted snippet, and whole document configurations

For PNA, event-based features only boosted performance in the whole document configuration. In oracle and predicted snippet experiments, event-based features lowered performance when used both as a standalone feature set and in combination with n-gram features. In PNA predicted snippet experiments, feature selection caused a decrease in performance with baseline and event-only configurations. The poor performance of event-features and feature selection itself could be due to the mismatch of event-based feature ranking between the MaxEnt model and the external χ^2 feature ranking system for PNA, as described in Section 4.6.5.3.

Config.	**Features**	θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
oracle	baseline	100	1343	1343	1147	196	196	79.3	77.8	**78.3**	85.4
predicted	baseline	150	1343	1343	1119	224	224	77.5	74.4	**75.1**	83.3
whole	all	200	1343	1343	1100	243	243	76.0	73.4	**74.3**	81.9

Table 4.65: The highest performing experiments for the PNA category in oracle snippet, predicted snippet, and whole document configurations

Event-only experiments, even with feature selection, did not exceed baseline experiments in most of the experiment scenarios. As discussed in the previous section, event-based features appear to boost performance when combined with the baseline n-gram model, but do not outperform it when used as a standalone feature set.

Event attributes were added to the change-of-state and diagnosis event schema to generalize the language of change in change-of-state events and hedging in diagnosis events. Full description of the attributes and their values can be found in Section A.1.3.1 and Section A.1.3.2. Examples of change of state attribute include *increase*, *decrease*, *persist*, and *change*. I did not develop an attribute extractor during the course of the study and therefore attributes were only used in oracle experiments. There are, however, attributes that rank within the top 250 χ^2 feature rankings for oracle snippets and this potential predictive value suggests that creating an extractor for attribute event features in future experiments would

be helpful, especially given that attributes generalize over specific terms relating to change, which are already captured in the n-gram model.

Running the snippet prediction, NER, and RE modules against the text of the whole document reveals that change-of-state and diagnosis events exist outside of snippet boundaries in reports. It also emphasizes that even though non-snippet reports do not contain predicted rationale snippets, they can contain change-of-state and diagnosis events. Extending the annotation process for change-of-state to the complete corpus will add more examples of change-of-state to the existing modules and possibly contribute to improving the model for the negative class.

Chapter 5

ACQUIRED LUNG INJURY (ALI)

In the previous Chapter, I described my study of an NLP radiology report classification system to detect evidence of pneumonia in chest X-ray reports by integrating features extracted from change-of-state and diagnosis events. The system identified change-of-state and diagnosis events in clinical reports by using a three-stage process: (1) predicting rationale text snippets for pneumonia detection, (2) labeling named entities in rationale snippet sentences, and (3) creating a dependency tree by connecting entities to one another with labeled arcs. How constrained are these components by the language of the report classification task they were trained on and the relatively small set of reports and rationale snippets in the pneumonia report classification chest X-ray corpus?

In this study, I explore the adaptability of the NLP components I developed to detect pneumonia in chest X-ray reports to other disease scenarios and I evaluate whether modules trained on a corpus annotated for suspicion of pneumonia (PNA) and Clinical Pulmonary Infection Score (CPIS) can be applied to similar diseases and related corpora. In Section 5.1, I provide an overview of a report classification task for acute lung injury (ALI), similar to the pneumonia report classification task I explored in Chapter 4, and describe why it is a candidate for demonstrating the adaptability of the pneumonia report classification NLP components to a related disease report classification task. In Section 5.2, I review the previous research of Yetisgen-Yildiz et al. (2013a) and their development of an ALI patient classification system and annotated corpus. I then describe the methodology I will use to replicate the patient classification system and experiments described in Yetisgen-Yildiz et al. (2013a) in Section 5.3, and detail my implementation of snippet prediction and change-of-state and diagnosis event components for ALI report classification in Section 5.4. I compare

the ALI report classification experimental results after integrating event-based features with the baseline results I generate with the replicated system in Section 5.5, and discuss the outcome of my experiments and my conclusions in Section 5.6.

5.1 Task overview

ALI and acute respiratory distress syndrome (ARDS) are critical lung illnesses that together represent 7% of intensive care admissions (Rubenfeld et al., 2005). Early detection is important given that the appropriate treatment, lung protective ventilation (LPV), is often applied too late or not at all, increasing the chance the patient will not recover (Ferguson et al., 2005; Rubenfeld et al., 2004). The identification of ALI requires recognition of a precipitating cause, either due to direct lung injury from trauma or pneumonia or secondary to another insult such as sepsis, transfusion, or pancreatitis. The consensus criteria for ALI include the presence of bilateral pulmonary infiltrates on chest radiograph, representing non-cardiac pulmonary edema as evidenced by the absence of left atrial hypertension (Pulmonary Capillary Wedge Pressure $<$ 18 mmHg (2.4 kPa)) or absence of clinical evidence of congestive heart failure, and oxygenation impairment as defined by an arterial versus inspired oxygen level ratio (PaO2/FiO2) $<$ 300 mmHg (40 kPa)) (Articas et al., 1998; Dushianthan et al., 2011; ARDS Task Force, 2012).

Yetisgen-Yildiz et al. (2013a) annotated a corpus of chest X-ray reports and developed an NLP-based patient classification system to identify patients with lung injury by classifying a patient's X-ray reports as being consistent or non-consistent with a diagnosis of ALI. They selected X-ray reports that represent a cohort of patients who meet the minimum oxygenation criteria for ALI, and extracted an n-gram feature set to train and test a maximum entropy (MaxEnt)-based classification system that used χ^2 feature selection and assertion classification to optimize performance through feature space reduction. In their study, they addressed the limitations of their n-gram approach by suggesting that a richer set of features, based on the semantics of change-of-state as described in Vanderwende et al. (2013) may improve the performance of their classifier.

The chest X-ray reports I developed into an annotated unique rationale snippet corpus for change-of-state and diagnosis events in the previous chapter are from the same institution and electronic health record (EHR) system, Harborview Medical Center, as the reports used to develop an ALI extractor in Yetisgen-Yildiz et al. (2013a). Although the exact disease description for ALI and pneumonia differ, a random sampling of the report text reveals similar observations and findings are used to develop consensus diagnosis by radiologists and clinicians. Table 5.1 compares side-by-side, the top 15 unigram features, ranked by χ^2 score, in the ALI and pneumonia report classification corpus. Although not exactly the same, many of the listed terms come from the same domain—lung injury or disease.

ALI Unigram Features	VAP Unigram Features
DIFFUSE	OPACITY
ATELECTASIS	ATELECTASIS
PULMONARY	OPACITIES
EDEMA	LOBE
OPACITIES	CONSOLIDATION
CONSISTENT	DIFFUSE
ALVEOLAR	ASPIRATION
DAMAGE	UPPER
BILATERAL	ABNORMALITIES
DISEASE	TUBES
WORSENING	LEFT
LUNG	RADIOLOGISTS
SEVERE	ADVERSE
CLEAR	RIGHT
PHYSICIANS	FOCAL

Table 5.1: A comparison of the top 15 unigram features from the ALI and pneumonia report classification corpora, measured by χ^2

In order to explore the adaptability of the pneumonia report classification rationale snippet prediction module and change-of-state and diagnosis event named entity recognition (NER) and relation extraction (RE) event detection modules, I applied them as a case study to the task of ALI report classification in the chest X-ray report corpora annotated by Yetisgen-Yildiz et al. (2013a). I pivoted the data from the ALI database to reshape the classification task as report-based rather than patient-based to align more with the original pneumonia report classification task.

5.2 Previous research

Yetisgen-Yildiz et al. (2013a) developed an ALI extractor, which identified patients with ALI symptoms from an annotated corpus comprised of 1748 chest x-ray reports. The corpus was generated for 629 patients (average number of reports=2.78, min=1, max=3) and annotated by three medical expert annotators. The 629 subjects were selected from a cohort of intensive care unit (ICU) patients at Harborview Medical Center, previously described in Glavan et al. (2011), who met the oxygenation criteria for ALI (PaO2/FiO2<300 mmHg). Three Critical Care Medicine specialists reviewed the chest radiograph images for each patient and annotated the radiographs as consistent (positive) or not-consistent (negative) with ALI. Yetisgen-Yildiz et al. (2013a) assigned ALI status for each subject based on the number of physician raters calling the chest radiographs consistent or not. 254 patients were assigned to the positive set (2 or more physicians agreeing on ALI positive) and 375 patients to the negative set (2 or more physicians agreeing on ALI negative). See Table 5.2 for a summary of agreement counts between annotators.

Yetisgen-Yildiz et al. (2013a) used a MaxEnt classifier in the MALLET[1] (McCallum, 2002) machine learning package to classify ALI patients and evaluated performance using 10-fold cross-validation on the corpus of 1748 X-ray reports. Three types of experiments were performed: (1) n-gram experiments using different combinations of n-gram features,

[1]http://mallet.cs.umass.edu/

(2) feature selection experiments using different thresholds of n-gram features ranked by χ^2 statistical feature selection, and (3) assertion analysis experiments using assertion classes as described in (Bejan et al., 2013b). Their best performing system was an n-gram model of combined unigram, bigram, and trigrams, with a feature selection threshold of $\theta = 800$ features. They reported system performance of 81.7 precision, 75.6 recall, and an F-score of 78.5.

Based on error analysis and a close examination of the language of false negative and false positive reports, Yetisgen-Yildiz et al. (2013a) suggested in the conclusion of their study that additional semantic information, such as the change-of-state and diagnosis events proposed in Vanderwende et al. (2013) may contribute to better performance in classification.

Annotation	**Agreement**	**Patient Count**
ALI positive patients	3	147
	2	107
ALI negative patients	3	205
	2	170

Table 5.2: Agreement levels between medical expert annotators on ALI status

5.3 Methodology

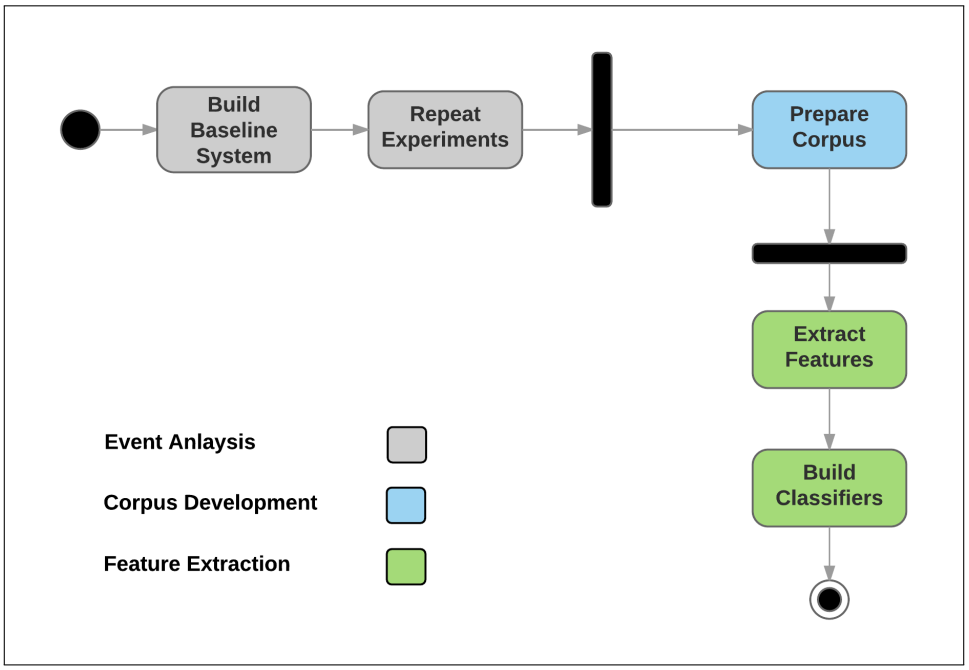

Figure 5.1: ALI methodology

In Chapter 3, I propose four phases to structure my research into clinical events:

1) **Event analysis**—analyze the task and related previous research to define events

2) **Corpus development**—develop corpora by annotating events

3) **Event detection**—build systems for automatic event detection

4) **Event-based feature extraction**—extract event-based features for classification

In this section I adapt my general research framework to the study of the adaptability of change-of-state and diagnosis event events and extraction tools trained on the pneumonia report classification corpus, to an alternate disease report classification task, ALI. The research framework is dramatically simplified—there are no annotation or event detection phases.

5.3.1 Event analysis

In the event analysis phase, I replicate the ALI NLP system and experiments described in Yetisgen-Yildiz et al. (2013a). I randomly distribute reports from the 1748 X-ray report corpus into five folds for 5-fold cross-validation.

5.3.2 Corpus development

I made no changes to the existing ALI corpus, with the exception of file and format changes importing the ALI corpus from a text-based MySQL bulk export to a table-based representation in the system database. The corpus contains only ALI labels used in the report classification task. I do not add any additional annotation to the corpus—all named entities, events, and rationale snippets are extracted using components trained on the pneumonia report classification corpus.

5.3.3 Event extraction

I used the pneumonia report classification NER and dependency extraction modules, trained on the pneumonia report classification unique snippet corpus to extract events as CoNLL 2007 Shared Task on Dependency Parsing formatted files. I parsed the dependency trees in these files with a Java-based import tool and converted them to a table-based representation in the system database. A final Java-based conversion tool, extracted tuples and event-based features as feature vector strings from the system database to create feature vector strings for training in MALLET.

5.3.4 ALI classification applied task

Table 5.3 briefly summaries the event-based features generated by the pneumonia report classification tuple and feature extraction module. The only event type previously included in the pneumonia report classification task but not the in the ALI experiments, is the Attribute event type, which was only used in oracle experiments for pneumonia report classification.

Feature	Description
Entity	Name and value pair of a event-based named entity annotation Pattern: {Entity Type} = {Entity Value} Example: `VAL=CLEAR`
Tuple	N-Tuple extracted from a change-of-state or diagnosis event Pattern: [{`Cos`_Value},{`Attr`_Value},{`Val`_Value},{`Loc`_Value},{`Ref`_Value}] Example: `[NEW_NEG,ABNORMALITIES,FOCAL,LUNG,AS_PER_LAST_EXAM]`
Change-of-state	Specific `Cos` value replaced with a generic `COS`. `Val` fields in the n-tuple are normalized with an underscore if they are multi-word, and prefix the `Attr` value. `Loc` entities are added as a suffix to the `Attr` value with an @ symbol. Pattern: `COS`→{`Val_Value`}_{`Attr_Value`}_`@`_{`Loc_Value`} Example: `COS`→`PATCHY_EDEMA_@_LEFT_LUNG`
Diagnosis	Specific `Dhead` value replaced with a generic DIAGNOSIS, `Val` and `Loc` fields follow same pattern as Change-of-state feature above. Pattern: `DIAGNOSIS`→{`Val_Value`}_{`Attr_Value`}_`@`_{`Loc_Value`} Example: `DIAGNOSIS`→`PNEUMONIA_@_RIGHT_LUNG`
Observation	Specific `Attr` value replaced with a generic OBS (for observation), `Val` and `Loc` fields follow same pattern as Change-of-state feature above. Pattern: `OBS`→{`Val_Value`}_{`Attr_Value`}_`@`_{`Loc_Value`} Example: `OBS`→`PATCHY_ATELECTASIS_@_LUNGS`

Table 5.3: A description and example of event-based features generated from change-of-state and diagnosis event dependency trees

5.4 Implementation

To measure the impact of change-of-state and diagnosis event features on the performance of ALI report classification, I replicated the NLP system and experiments described in Yetisgen-Yildiz et al. (2013a) to establish a baseline for comparison and a system to implement new experiments. I used four extraction modules and their associated models trained on the pneumonia report classification corpus to extract event-based features. See Table 5.4 for a summary of the modules, models, and the description of the task and Table 5.5 for a description of change-of-state and diagnosis event features. Figure 5.2 provides an diagrammatic view of the overall system and data flow.

Module	Model	Task Description
Snippet Prediction	CPIS/PNA snippet	Predict rationale snippets with the pneumonia report classification snippet prediction module using models trained on CPIS and PNA labels snippets from the pneumonia report classification corpus
NER	Unique snippet corpus	Extract named entities from predicted snippets and whole reports from the ALI corpus using the pneumonia report classification NER module and models trained on the change-of-state and diagnosis entities annotated in the pneumonia report classification unique rationale snippet corpus
Dependency	Unique snippet corpus	Extract change-of-state and diagnosis dependency trees from predicted snippets and whole reports from the ALI corpus based on the named entities output by the pneumonia report classification NER module, and models trained on the pneumonia report classification unique rationale snippet corpus.
Tuple and Event Feature	[Rule-based]	Extract change-of-state and diagnosis event features from the tuples output by the pneumonia report classification tuple extraction module.

Table 5.4: Pneumonia report classification modules and models used in ALI report classification study

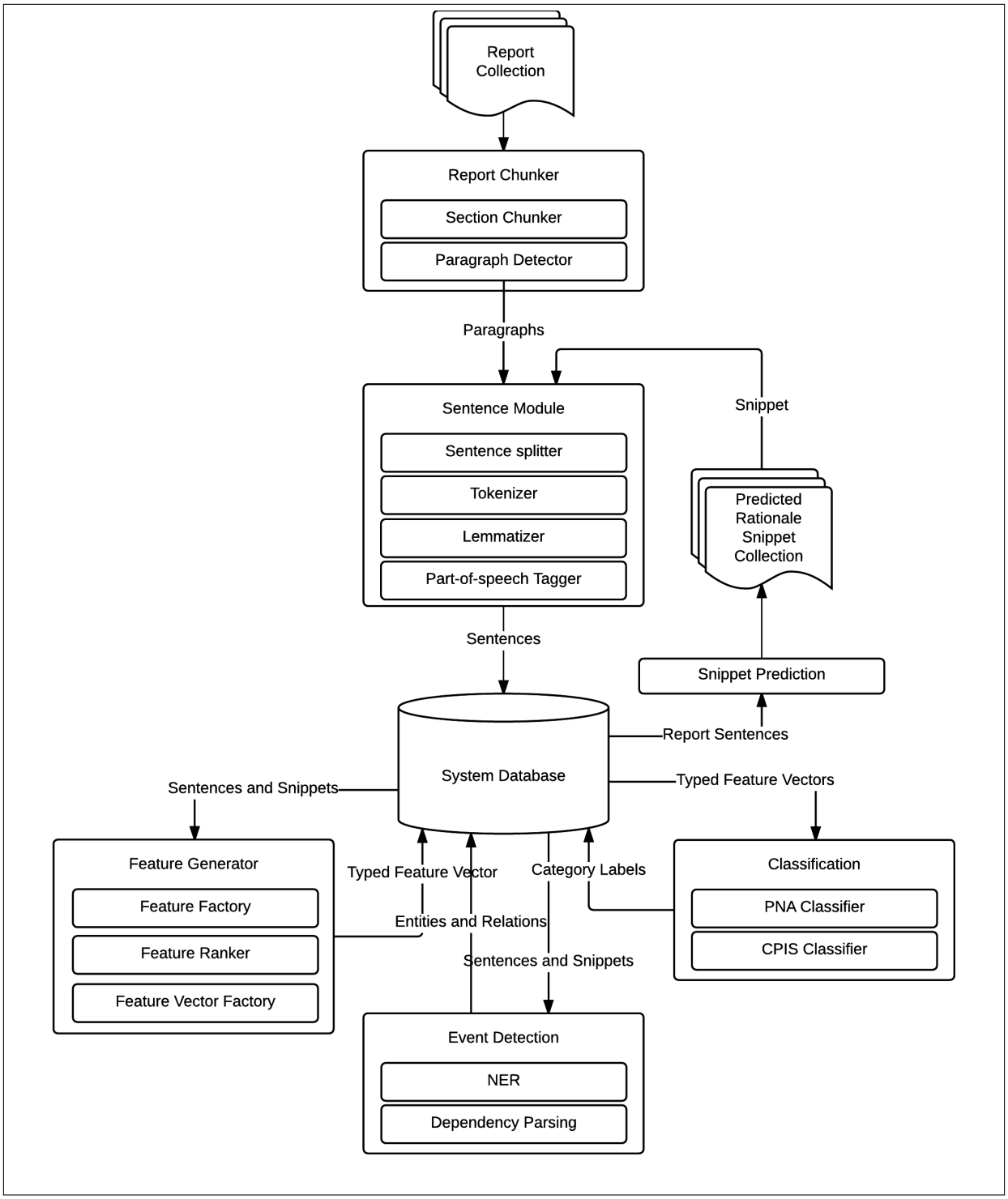

Figure 5.2: ALI system diagram

Feature	Baseline	Event-only	All
Word unigram and bigram	X		X
Entity attribute name & value		X	X
Entity name & value		X	X
Event tuple		X	X
Change of state		X	X
Diagnosis		X	X
Observation		X	X

Table 5.5: ALI report classification feature sets

I integrated the extracted event-based features into feature sets for new ALI report classification experiments and measured the performance of the classifier with the evaluation measures precision, recall, F-score, and accuracy.

5.5 Results

In this section, I review results for the final ALI report classification experiments for whole document, CPIS and PNA predicted snippets. I also discuss additional feature selection threshold experiments I conducted to explore the impact of feature selection on the performance of the report classifier.

5.5.1 ALI report classification results

Table 7.5 lists the results from the original study in Yetisgen-Yildiz et al. (2013a). The study emphasized feature selection as the approach with the most impact of the different approaches they explored in their study. An important aspect of their classification task that differs from mine is that they pivoted their classification task on the patient. The

classification feature vector is a roll-up of all features in all the reports associated with the same patient id. Because categorical information for each report, positive or negative for ALI, was reverse engineered from the patient classifications in the database, and my snippet prediction module was trained on reports, I chose to use this information to change the task to report classification instead of patient classification.

Table 5.7 lists the results of my baseline experiments with default features (unigram plus bigram plus trigram), Table 5.8 lists experiments with event-based features only, and Table 5.9 with all features.

Feature Type	θ	TP	TN	FP	FN	P	R	F1	Acc
Uni+Bi+Tri	0	470	783	220	275	68.1	63.1	65.5	71.7
Uni+Bi+Tri	800	494	833	170	251	74.4	66.3	70.1	75.9

Table 5.6: Baseline n-gram results from replicated system (evaluated by report)

Textual Unit	θ	TP	TN	FP	FN	P	R	F1	Acc
Whole	0	470	783	220	275	68.1	63.1	65.5	71.7
Document	800	494	833	170	251	74.4	66.3	70.1	75.9
CPIS	0	463	796	207	282	69.1	62.1	65.4	72.0
Predicted Snippets	800	455	833	170	290	72.8	61.1	66.4	73.7
PNA	0	480	807	196	265	71.0	64.4	67.5	73.6
Predicted Snippets	800	469	836	167	276	73.7	63.0	67.9	74.6

Table 5.7: Final ALI report classification results with **baseline** features.

Textual Unit	θ	TP	TN	FP	FN	P	R	F1	Acc
Whole	0	469	770	233	276	66.8	63.0	64.8	70.9
Document	800	496	845	158	249	75.8	66.6	**70.9**	76.7
CPIS	0	465	796	207	280	69.2	62.4	65.6	72.1
Predicted Snippets	800	465	832	171	280	73.1	62.4	67.3	74.2
PNA	0	459	792	211	286	68.5	61.6	64.9	71.2
Predicted Snippets	800	446	830	173	299	72.1	59.9	65.4	73.0

Table 5.8: Final ALI report classification results with **event-only** features.

Textual Unit	θ	TP	TN	FP	FN	P	R	F1	Acc
Whole	0	471	781	222	274	68.0	63.2	65.5	71.6
Document	800	490	839	164	255	74.9	65.8	70.1	76.0
CPIS	0	467	788	215	278	68.5	62.7	65.4	71.8
Predicted Snippets	800	469	836	167	276	73.7	63.0	67.9	74.7
PNA	0	480	805	198	265	70.8	64.4	67.4	73.5
Predicted Snippets	800	471	841	162	274	74.4	63.2	68.4	75.1

Table 5.9: Final ALI report classification results with **all** features.

The impact of event-based features on the results of the ALI report classification experiments are mixed and inconclusive. Experiments with all baseline and event features extracted from oracle snippets and whole documents benefit from feature selection, with whole document extraction demonstrating the most improvement in F-score. Extracting features from predicted snippets does not benefit ALI classification when snippets are predicted using a CPIS-snippet trained model, but do benefit when snippets are predicted using a PNA-snippet trained model. Again, as mentioned in the discussion in Chapter 4, this may be influenced by the distribution of reports across PNA and CPIS label in the original corpus on which the snippet prediction module was trained. The best performing ALI classification

system, the only system to marginally beat the baseline of F-score 70.1 with an F-score of 70.9, is with event features-only, in whole document configuration, with a feature threshold of 800.

5.5.2 ALI feature selection threshold results

In this section, I explore the effect of feature selection thresholds on the performance of the ALI report classifier. A set of experiments for whole documents were run and are listed and discussed in this section. No additional feature selection threshold experiments were run for predicted CPIS and PNA snippets. The whole document experiments were grouped into sets for baseline features, event-only features, and all features. Each set based on feature type included one or more experiments for a specific feature threshold. The same increments of feature threshold were applied across all experiment sets. The standard feature threshold increments for whole documents were: 50, 100, 150, 200, 250, 500, 750, 1000, 1500, 2500, 5000, 7500, 10000, and 15000. Three other factors influenced the number of feature threshold experiments run for each experiment set: (1) if the maximum number of χ^2 significant features was less than one of the standard feature threshold increments, experiments were not run for thresholds greater than the maximum, (2) if the maximum number of average features per fold in the MaxEnt model with no feature selection was less than a feature threshold, no experiments were run for feature threshold with a greater value, and (3) the F-scores for the larger feature threshold increments demonstrated a pattern of diminishing performance. The row at the top of each table with a hyphen as its value for θ represents an experiment with no feature selection.

The series of Tables 5.10 - 5.12 list the results of ALI report classification experiments for on whole documents for baseline, event-only, and all features. The Table 5.13 lists the highest performing experiment across the three feature sets.

θ	**S**	**G**	**TP**	**TN**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1748	1748	470	783	220	275	68.1	63.1	65.5	71.7
10000	1748	1748	488	873	130	257	79.0	65.5	71.6	77.9
7500	1748	1748	504	869	134	241	79.0	67.7	72.9	78.5
5000	1748	1748	516	871	132	229	79.6	69.3	74.1	79.3
2500	1748	1748	527	868	135	218	79.6	70.7	74.9	79.8
1500	1748	1748	516	849	154	229	77.0	69.3	72.9	78.1
1000	1748	1748	504	852	151	241	76.9	67.7	72.0	77.6
750	1748	1748	509	839	164	236	75.6	68.3	71.8	77.1
500	1748	1748	493	837	166	252	74.8	66.2	70.2	76.1
250	1748	1748	484	839	164	261	74.7	65.0	69.5	75.7
150	1748	1748	490	849	154	255	76.1	65.8	70.6	76.6
100	1748	1748	469	839	164	276	74.1	63.0	68.1	74.8
50	1748	1748	456	836	167	289	73.2	61.2	66.7	73.9

Table 5.10: **Baseline** features (unigram, bigram, and trigram) on whole document (Average number of features across folds in model with no feature selection = 23,483/Number of significant χ^2 ranked features = 8500

θ	**S**	**G**	**TP**	**TN**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1748	1748	469	770	233	276	66.8	63.0	64.8	70.9
15000	1748	1748	469	770	233	276	66.8	63.0	64.8	70.9
10000	1748	1748	475	853	150	270	76.0	63.8	69.3	76.0
7500	1748	1748	472	862	141	273	77.0	63.4	69.5	76.3
5000	1748	1748	478	862	141	267	77.2	64.2	70.1	76.7
2500	1748	1748	505	851	152	240	76.9	67.8	72.0	77.6
1500	1748	1748	513	850	153	232	77.0	68.9	72.7	78.0
1000	1748	1748	506	850	153	239	76.8	67.9	72.1	77.6
750	1748	1748	495	845	158	250	75.8	66.4	70.8	76.7
500	1748	1748	484	841	162	261	74.9	65.0	69.6	75.8
250	1748	1748	475	846	157	270	75.2	63.8	69.0	75.6
150	1748	1748	453	831	172	292	72.5	60.8	66.1	73.5
100	1748	1748	437	851	152	308	74.2	58.7	65.5	73.7
50	1748	1748	425	834	169	320	71.5	57.0	63.5	72.0

Table 5.11: **Event-based** features only on whole document (Average number of features across folds in model with no feature selection = 13,297/Number of significant χ^2 ranked features = 14,100)

θ	**S**	**G**	**TP**	**TN**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1748	1748	471	781	222	274	68.0	63.2	65.5	71.6
15000	1748	1748	492	878	125	253	79.7	66.0	72.2	78.4
10000	1748	1748	502	877	126	243	79.9	67.4	73.1	78.9
7500	1748	1748	511	868	135	234	79.1	68.6	73.5	78.9
5000	1748	1748	516	875	128	229	80.1	69.3	74.3	79.6
2500	1748	1748	508	845	158	237	76.3	68.2	72.0	77.4
1500	1748	1748	499	837	166	246	75.0	67.0	70.8	76.4
1000	1748	1748	492	840	163	253	75.1	66.0	70.3	76.2
750	1748	1748	492	841	162	253	75.2	66.0	70.3	76.3
500	1748	1748	488	837	166	257	74.6	65.5	69.8	75.8
250	1748	1748	471	837	166	274	73.9	63.2	68.2	74.8
150	1748	1748	454	845	158	291	74.2	60.9	66.9	74.3
100	1748	1748	455	847	156	290	74.5	61.1	67.1	74.5
50	1748	1748	433	831	172	312	71.6	58.1	64.1	72.3

Table 5.12: **All** features on whole document (Average number of features across folds in model with no feature selection = 36,780/Number of significant χ^2 ranked features = 15,000)

Features	θ	**S**	**G**	**TP**	**TN**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
baseline	2500	1748	1748	527	868	135	218	79.6	70.7	74.9	79.8
event-only	1500	1748	1748	513	850	153	232	77.0	68.9	72.7	78.0
all	5000	1748	1748	516	875	128	229	80.1	69.3	74.3	79.6

Table 5.13: The highest performing feature threshold experiments for whole document

5.6 Summary

In this study, I focused on adapting the components I developed for pneumonia report classification in Chapter 4 to the task of ALI report classification. The language of the radiology reports for ALI and VAP are close in regards to the anatomical locations, clinical

attributes, diagnosis, and change-of-state terms and phrases that are used by radiologists to describe imaging. I applied the pneumonia report classification snippet prediction modules for both CPIS and PNA to the ALI corpus, and ran the pneumonia report classification NER module and dependency parsing RE module on the snippets and whole document. I extracted five of the six event-based features defined in the pneumonia report classification study and applied those features to the ALI report classification task. I compared my results to the n-gram-based results published in Yetisgen-Yildiz et al. (2013a) with and without feature selection.

Although the language of ALI diagnosis is similar to pneumonia, there are a number of terms unique to ALI. Using the current pneumonia report-trained snippet prediction module to extract candidate ALI rationale snippets, and then vetting those candidate rationale snippets for ALI diagnosis could provide additional positive examples to retrain the pneumonia report classification snippet prediction module. Given the improvement in performance of pneumonia report classification based on the snippet prediction module in the pneumonia study, an ALI-trained snippet prediction module would likely produce better results than a pneumonia-trained model.

Integrating event-based features into the ALI report classification task matched but did not improve the performance of the report classifier trained on baseline n-gram features. The best performing configuration for optimal feature selection threshold was with baseline features and θ=2500 (see Table 5.13). Additional experiments, exploring feature selection threshold values improved the performance of alls classifier models, but did not demonstrate an increase of performance when event-based features were added to the feature set.

Although the results of the ALI experiments with event-based features did not demonstrate dramatic improvements in performance, they were able to identify snippets and extract change-of-state and diagnosis event trees from the ALI corpus. Given the simplicity of the change-of-state and diagnosis event annotation scheme, selecting a collection of candidate snippets from the ALI corpus and annotating them with change-of-state and diagnosis events annotations, adding these annotations to the pneumonia report classification corpus used to

train the original models, and then training new models, could demonstrate how to adapt the NER and RE extraction modules to the ALI report classification task.

Chapter 6

CRITICAL RECOMMENDATIONS

In Chapter 4, I developed a dependency tree-based event model for change-of-state and diagnosis events as an extension of the n-tuple representation proposed in Vanderwende et al. (2013). In this chapter, I explore an alternate structure for representing clinical events by modeling a critical follow-up recommendation event as a template of properties rather than an n-tuple or a dependency tree. Yetisgen-Yildiz et al. (2013b) defined a critical follow-up recommendation as a sentence in a radiology report that communicates an important and critical follow-up recommendation to clinicians or other ordering providers. In this study, I define a critical recommendation event as a critical recommendation sentence associated with a number of additional properties including:

- the text of the critical follow-up recommendation sentence
- `Reason`, `Test`, and `Time` entities found in the recommendation sentence (See Table 6.5 for definitions)
- the `index exam` described in the *exam description* metadata field of its radiology report
- the `diagnosis` in the *diagnosis summary* metadata field of its radiology report
- a *default recommended follow-up test* and *default follow-up timeframe* computed from information extracted from report text and metadata and based on rules prescribed by the radiologist

A critical follow-up recommendation sentence can co-occur with other critical follow-up recommendation sentences within the same radiology report. It does not necessarily share the same level of criticality and importance as the other critical follow-up recommendation sentences in the same report. Criticality and importance is a four label category I devel-

oped in collaboration with domain experts—an internal medicine specialist, radiologist, and attending neurologist—during the initial phase of corpus development (see Table 6.4 for definitions of the labels in the criticality and importance category). We defined and established annotation guidelines for identifying and labeling critical follow-up recommendation sentences with the four labels of criticality and importance. I developed a model for critical follow-up recommendation events through this process and created a classification system that automatically detects, extracts, and labels critical follow-up recommendation events from the text of radiology reports.

01 CT ABDOMEN AND PELVIS WITH INTRAVENOUS CONTRAST
02 HISTORY:
03 Prostate CA-Prostate CA Surveillance
04 COMPARISON: None
05 CONTRAST: iv contrast was used. Positive oral contrast was administrated
06 TECHNIQUE:
07 Region of interest: Abdomen-Pelvis
08 Superior Extent: Diaphragm. Inferior Extent: Symphysis Pubis
09 .....
10 FINDINGS:
11 Lung bases: A 6-mm nodule is noted in the peripheral left lung base (image 9, series 2). There is a focal area of
12 atelectasis in the anterior right lung base.
13 Pleura: No pleural effusions or thickening.
14 Included heart: No gross abnormality..
15 Liver: Normal
16 Portal veins: Normal.
17 Gallbladder and bile ducts: Normal
18 Spleen: Normal
19 Aorta and IVC: There is atherosclerotic calcification of the aorta.
20 There is a small focus of ulcerated plaque in the infrarenal aorta (image 49, series 2).
21 Stomach, duodenum and small bowel: Normal
22 .....
23 IMPRESSION:
24 1. Incidental 6-mm left lung nodule. Follow-up chest CT is recommended in 6 months.
25 2. A few prostatic calcifications are noted. No CT evidence of metastatic prostate cancer.
26 3. Small ulcerated atheromatous plaque in the infrarenal aorta.

Figure 6.1: A radiology report containing a critical follow-up recommendation sentence

Figure 6.1 is an example of a report that contains a critical follow-up recommendation sentence. The reason for the imaging test, prostate surveillance, is listed on line 03. Lines 11 and 12 record a radiologist's incidental finding of a nodule in the lung. Line 24 features the critical follow-up recommendation sentence: *Follow-up chest CT is recommended in 6 months.*

In Section 6.1, I provide an overview of the critical follow-up recommendation sentence identification task and the annotation of collections of candidate critical follow-up recommendation sentences with labels of criticality and importance. In Section 6.2, I review previous research—a radiology critical follow-up recommendation study that introduced a new corpus and a preliminary approach to critical follow-up recommendation sentence identification (Yetisgen-Yildiz et al., 2013a). In Section 6.3, I describe how I adapt my general research methodology to the Yetisgen-Yildiz et al. (2013a) critical follow-up recommendation study and, in Section 6.4, I detail my implementation of a critical follow-up recommendation annotation process, event analysis, sentence identification and event classification system. In Section 6.5, I review the results of my experiments and component evaluations and, in Section 6.6, I discuss the outcomes of my experiments and component evaluations. I present the conclusions of my study in Section 6.7.

6.1 Task overview

In the last ten years, radiologists and clinicians have benefited greatly from advances in imaging technologies and their increased availability, giving rise to previously unavailable diagnostic and screening capabilities (Hendee et al., 2010). However, the growth in the amount of reports and images that require review and comment demands that radiologists optimize their use of clinical information without adding unnecessary noise to the signal between radiologist and ordering provider. They are called on not only to answer questions posed by ordering providers, but also called on to identify unexpected incidental findings that might pose a significant health risk to patients in the short or medium term (Berland et al., 2010). Automated systems that integrate with the electronic health record (EHR) are necessary to support the radiologist in tracking and identifying both critical test results and unexpected findings.

Yetisgen-Yildiz et al. (2013b) defined a critical follow-up recommendation sentence as a statement made by a radiologist in a given radiology report to advise the referring clinician to further evaluate an imaging finding by either other tests or further imaging. A critical follow-

up recommendation sentence may contain the same or similar text as other critical follow-up recommendation sentences, but it might differ from the other sentences in its importance and criticality. The difference between the two sentences, one being an important and critical recommendation and the other not, lies in the context of surrounding information in the report, its metadata, and subtle differences in wording.

Yetisgen-Yildiz et al. (2013b) stated their long term goal was to identify clinically important and critical recommendations so that the reports can be flagged visually and electronically. Separate workflow processes can then be initiated to reduce the chance that important test and follow-up exams suggested in the report are not missed by clinicians. To accomplish this goal, they designed a text processing approach based on NLP and supervised machine learning to identify sentences that contain clinically important recommendation information. In their study, they introduced a corpus of 800 randomly-sampled radiology reports, annotated by two medical experts for critical follow-up recommendation sentences.

In this study, I define a critical follow-up recommendation event model and develop a system to extract and label critical follow-up recommendation events by their importance and criticality. I build upon the previous research in Yetisgen-Yildiz et al. (2013b), and work with medical expert annotators to annotate critical follow-up recommendation events in randomly selected collections of imaging reports from a new multi-institutional corpus.

6.2 Previous research

In order to develop and evaluate a system to identify critical recommendations, Yetisgen-Yildiz et al. (2013b) created a corpus of radiology reports composed of 800 randomly selected and de-identified radiology reports, across 12 imaging modalities (See Table 6.1), extracted from the Harborview Medical Center radiology information system (Yetisgen-Yildiz et al., 2013b). The reports were generated by a reporting system that offers radiologists multiple modes in one user interface for recording their observations:

- a microphone-enabled voice recognition feature for speech-to-text translation
- a mouse-based, menu-driven controlled vocabulary and structured text template entry feature
- a keyboard-based feature for adding freely typed text and editing previously translated speech-to-text and menu-selected entries

Imaging modality	Frequency
Computer tomography (CT)	486
Radiograph	259
Magnetic resonance imaging (MRI)	45
Ultrasound	10

Table 6.1: Distribution of corpus radiology reports across imaging modalities from Yetisgen et. al., (2013)

Two annotators, a radiologist and a clinician, evaluated each sentence in the corpus, and identified critical follow-up recommendation sentences in two rounds of annotation. The annotations were compared by inter-annotator agreement (IAA) measures and differences in annotation were resolved via consensus. See Table 6.2 for the IAA comparison of the two annotators. The high Kappa scores for the second round of annotations suggest the definition of a follow-up recommendation was easy to understand and apply for the medical expert annotators. At the end of the process, the annotators identified 113 critical follow-up recommendations as gold standard and finalized the annotation guidelines (Yetisgen-Yildiz et al., 2013b).

Round	Annotator 1	Annotator 2	Agreed	P	R	F1	Kappa
1st	110	109	83	75.5	76.1	75.8	75.7
2nd	114	118	113	99.1	95.8	97.4	97.4

Table 6.2: IAA measures for corpus annotation from Yetisgen et. al., (2013)

The 113 annotated recommendation sentences in the gold standard were used to train a maximum entropy (MaxEnt) model with MALLET[1] (McCallum, 2002) incorporating four types of features:

1. word *n*-gram
2. syntactic and temporal
3. knowledge-based
4. structural

Features were ranked by χ^2 value and experiments were run with different feature count thresholds. Due to the number of positive sentences (113) being much smaller than negative sentences (18,364), data balancing was used to identify an optimal ratio of positive to negative sentences (Yetisgen-Yildiz et al., 2013b). They experimented with feature selection, feature types, and data balancing over 165 partitions, and achieved the highest F-score of 75.8 at k = 9 (precision = 66.2, recall = 88.5) where the ratio of positive to negative sentences in the training data was 1 to 9 (k is the number of partitions of negative sentences that are used in the data-balanced fold configurations).

6.3 Methodology

I organize my study into an adapted form of the four-phase research framework I described in Chapter 3. The four phases are:

[1]http://mallet.cs.umass.edu/

1) **Event analysis**—analyze the task and related previous research to define events
2) **Corpus development**—develop corpora by annotating events
3) **Event detection**—build systems for automatic event detection
4) **Event-based feature extraction**—extract event-based features for classification

Figure 6.2 provides a visual representation of the flow of research-related tasks in the adapted critical follow-up recommendation methodology framework. In this study, the critical follow-up recommendation event classification task is new and not an extension of the previous study's approach. Critical follow-up recommendation sentence identification, as described in Yetisgen-Yildiz et al. (2013b), is used as a preliminary step in critical follow-up recommendation event classification, but no error analysis contributes to the critical follow-up recommendation event model. That step is removed from the general framework in this study. Another difference between the general research framework and the research methodology for this study, is the removal of the corpus gold standard publishing step. The corpus used in this study is not available for release.

6.3.1 Event analysis

In the first phase of research, the general framework was adapted for two basic tasks:

1. design a critical follow-up recommendation event model in consultation with domain experts
2. replicate the system from Yetisgen-Yildiz et al. (2013b) and repeat critical follow-up recommendation sentence identification data-balancing experiments

The event model for a critical follow-up recommendation is a template of properties that includes the original critical follow-up recommendation sentence but also other information, such as relevant report metadata, `Time`, `Test`, and `Reason` entities, etc. In this phase, I defined the properties of the event model (see Section 6.4.2 for details of the implementation).

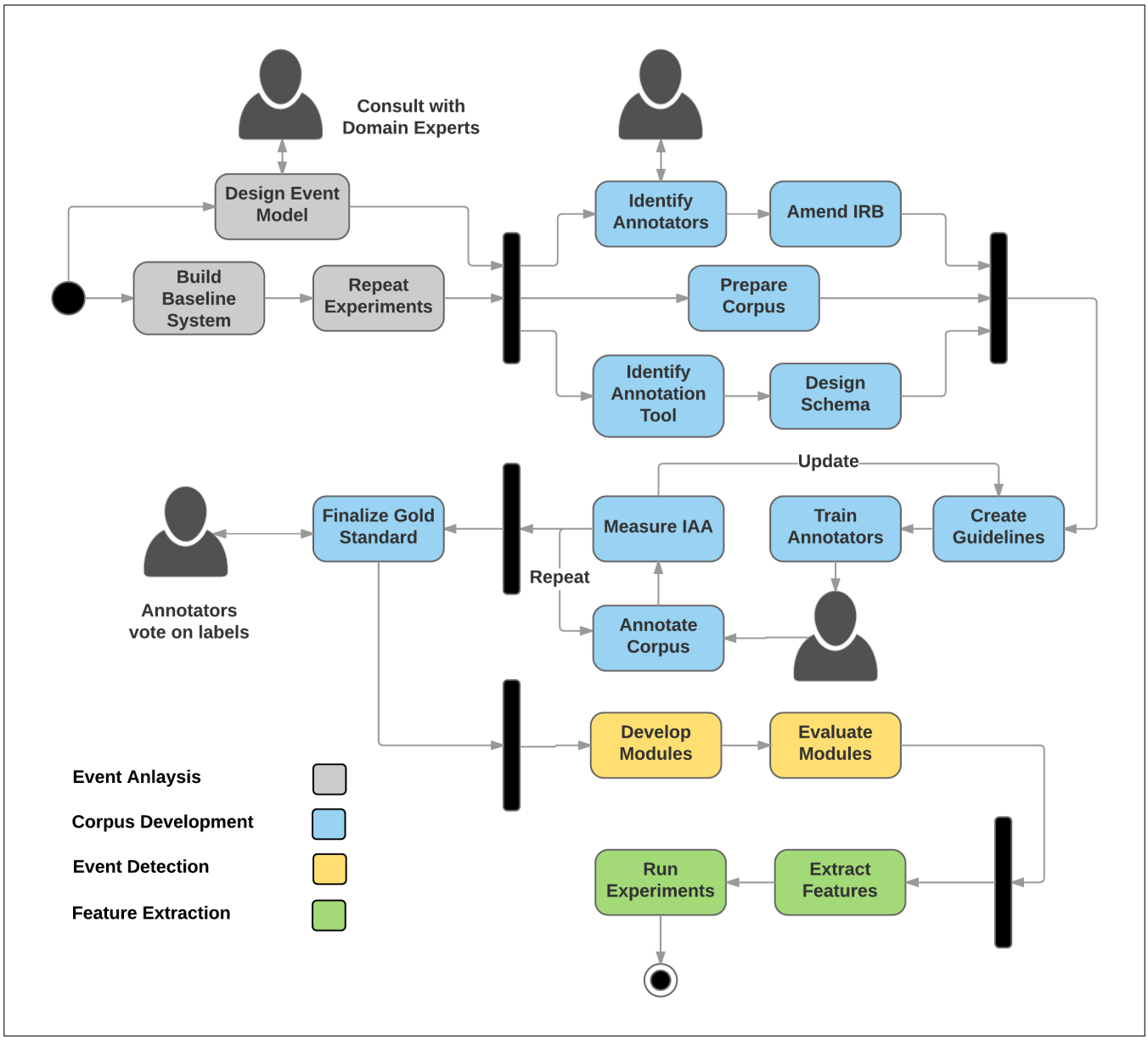

Figure 6.2: Critical follow-up recommendation methodology overview

To replicate the data-balancing experiments from Yetisgen-Yildiz et al. (2013b), I created a series of high-recall/low-precision critical follow-up recommendation sentence identification models and selected the most appropriate for the task of identifying candidate sentences

in the multi-institutional corpus. Once the model was trained, candidate sentences were identified and added to a new sub-corpus. The sub-corpus was annotated by medical expert annotators for labels of criticality and importance and by medical students for the entities `Reason`, `Test`, and `Time`.

6.3.2 Corpus development

There were three stages in the corpus development phase:

Corpus preparation To prepare the corpus, I exported a collection of multi-institutional radiology reports from the EHR into the system database. Only reports that met the modality requirements of the study, contained the required metadata fields, and had report text, were selected and organized into randomly sampled collections of 1000. The critical follow-up recommendation sentence identifier was run on eight collections of 1000 reports, and the same medical expert annotators from the Yetisgen-Yildiz et al. (2013b) study participated in the validation of system-generated candidate critical follow-up recommendation sentences. Three additional annotators joined the study and were added to the institutional review board (IRB) agreement. A new annotation tool, BRAT[2] (Stenetorp et al., 2012), was selected, and a BRAT schema for annotation was designed with the help of medical expert annotators. See Section 6.4 for implementation details. Tasks adapted from the general framework include:

1. identify annotators
2. amend IRB
3. prepare corpus
4. identify annotation tool
5. design schema

[2]http://brat.nlplab.org/index.html

Corpus annotation The corpus annotation phase occurred at three levels: (1) the binary classification level, where medical expert annotators validated system-generated critical follow-up recommendation sentences, (2) the sentence labeling level, where medical expert annotators applied a label of criticality and importance to the valid sentences, and (3) at the entity level, where medical student annotators identified `Reason`, `Test`, and `Time` entities in the sentence text. The tasks of validating system-generated critical follow-up recommendation sentences and applying labels of criticality and importance to the identified sentences required the expertise of medical experts, while entity level annotation was concerned only with `Reason`, `Test`, and `Time` entities, and therefore a task appropriate for medical student annotators. Sentence level classification was a difficult task for the medical expert annotators, with little consensus throughout the project. It was much easier to get consensus on label definitions at the entity level. The following tasks were adapted from the general framework for this phase:

1. create annotation guidelines
2. train annotators
3. annotate corpus
4. measure IAA

Corpus finalization For the sentence level annotation tasks, finalizing the gold standard required additional meetings due to the difficulty of establishing final guidelines concerning the labels of importance and criticality.

6.3.3 Event detection and feature extraction

Event detection for critical follow-up recommendations is the process of extracting features from template properties. Entities are one of the properties of a critical follow-up recommendation event and require the design and evaluation of a named entity recognition (NER) module to extract `Reason`, `Test`, and `Time` entities. See Section 6.4.5 for implementation details and Section 6.5.2 for NER module evaluation results.

The final versions of the binary critical follow-up recommendation sentence identifier, the NER module, the template-filling aggregation module for critical follow-up recommendation events, and the critical follow-up recommendation event criticality and importance classification system, were run on the remaining unlabeled collections of the multi-institutional corpus to calculate statistics for a future study. See Section 6.5 for results of running these tools on the entire multi-institutional corpus.

The baseline critical follow-up recommendation event classification experiments in this study were based on combinations of n-gram features (unigram, bigram, and trigram) extracted from the text of critical follow-up recommendation sentences and their report metadata. See Section 6.4.8 for implementation details and Section 6.5.4 for classification results.

6.4 Implementation

Table 6.3 breaks down the overall system development effort into individual components that were implemented during the four research phases of the study. The table also provides descriptions of each component and evaluation measures used to measure their performance. Figure 6.3 presents a diagrammatic view of the overall system and how information flows between major components.

Impl. Step	Research Phase	Description	Evaluation
Event Analysis	Baseline system	Replicate the data-balanced critical follow-up recommendation sentence identification system described in Yetisgen-Yildiz et al. (2013b)	Achieve high-recall/low-precision data-balanced results to select a model for critical follow-up recommendation identification
	Event model	Model the critical follow-up recommendation event, defining all properties for the template	Ensure all properties can be extracted for any critical follow-up recommendation
Corpus annotation	Annotation	Annotate reports by vetting system-generated critical follow-up recommendation, add labels for importance and criticality, and entities `Reason`, `Test`,and `Time`	IAA measures by label for sentence and entity level
Event Detection	NER module	Train and test an NER module to identify and label named entities in critical follow-up recommendation	P,R, and F1 measures of NER labeling by entity and token
Feature Extraction	Template Extractor	Module to extract all properties of a critical follow-up recommendation event for classification and feature extraction	Manual ad-hoc testing
	Critical follow-up recommendation identification classification system	Classify critical follow-up recommendation events by importance and criticality Identify candidate sentences over entire corpus	P, R, and F1 measured and evaluated against gold standard labels in multi-institutional corpus for 753 identified sentences in sub-corpus. Evaluation of 1000 randomly selected reports from the overall corpus.

Table 6.3: A breakdown of the overall critical follow-up recommendation task into steps and evaluation metrics

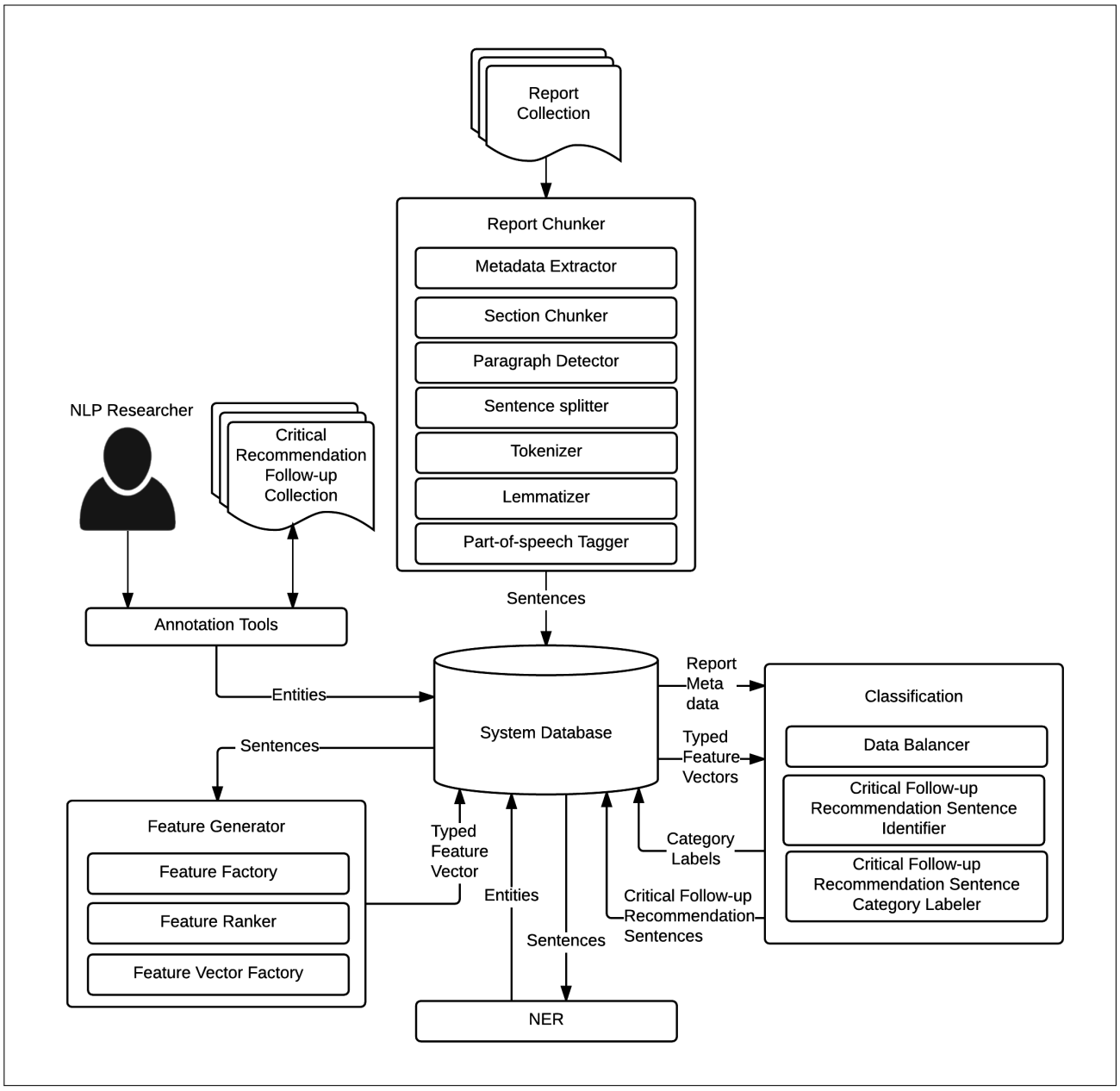

Figure 6.3: Critical follow-up recommendation system

6.4.1 Replicated baseline system and experiments

To prepare a corpus of candidate recommendation sentences for the first round of annotation I created a high-recall/low-precision model for binary recommendation sentence classification by training a classifier on the annotated corpus of 800 randomly sampled radiology reports from the Harborview Medical Center radiology information system described in Yetisgen-Yildiz et al. (2013b). I imported the data into the system database and used the same feature set, unigrams and syntactic features, in k-partition data-balancing experiments to select an appropriate high-recall/low-precision model for sentence identification. See Table 6.18 for a sampling of the k-partition table used to select a high-recall (90.2)/low-precision (35.1) model.

The high-recall/low-precision binary sentence classifier identified many false positives. Even though annotators had to vet many false positives, the filtering of reports using the high-recall/low-precision model reduced the number of overall reports they needed to review, which expedited the annotation process.

6.4.2 Event model

The medical expert annotators who annotated recommendation sentences and participated in creating the annotation guidelines in the previous study proposed a four-label category to differentiate between types of critical follow-up recommendations and to indicate their level of importance and criticality. The final list of four recommendation types were: (1) Non-contingent clinically important recommendation—an advisory statement that could result in mortality or significant morbidity if appropriate clinical assessment, diagnostic or therapeutic follow-up steps are not followed, (2) Contingent clinically important recommendation—similar to (1), but the statement is contingent on the presence of a clinical condition, (3) Clinically important recommendation likely reported—similar to (1) and (2), but considered to be unlikely not to be reported in communication between radiologist and clinician, and, (4) Clinically unimportant recommendation—an advisory statement that is unlikely to

result in mortality or significant morbidity if appropriate clinical assessment, diagnostic or therapeutic follow-up steps are not followed. See Table 6.4 for detailed description of each label and both case and recommendation sentence examples.

Name	Description	Case example	Recommendation example
Non-contingent clinically important recommendation	An advisory statement that could result in mortality or significant morbidity if appropriate clinical assessment, diagnostic or therapeutic follow-up steps are not followed.	Incidental lung mass suspicious for malignancy on a trauma CT of the abdomen.	CT chest is recommended to further evaluate the lung mass.
Contingent clinically important recommendation	Similar to non-contingent clinically important recommendation, but the statement is contingent on the presence of a clinical condition.	Adrenal mass identified on a CT of the abdomen and pelvis for appendicitis.	If the patient has a history of malignancy, consider bio-chemical testing and an adrenal mass protocol CT for further evaluation.
Clinically important recommendation likely reported (miss unlikely)	Similar to non-contingent and contingent clinically important recommendation, but considered to be unlikely not to be reported in communication between radiologist and clinician.	A distal radius fracture was identified on a previous week's x-ray of patients hand. A follow-up x-ray of the hand is requested to rule out possible additional scaphoid fracture.	L distal radius fracture x 1 week, please also follow-up to rule out scaphoid fracture compared with last week's x-rays.
Clinically unimportant recommendation	An advisory statement that is unlikely to result in mortality or significant morbidity if appropriate clinical assessment, diagnostic or therapeutic follow-up steps are not followed, and/or a low probability that the recommendation would be overlooked.	Following trauma, a radiograph demonstrates a probable non-displaced fracture of the mid ulna.	Consider an MRI of the forearm if diagnostic certainty is desired

Table 6.4: Labels of criticality and importance for critical follow-up recommendation sentences and events

The medical expert annotators also defined three optional and repeatable entities, `Reason`, `Test`, and `Time`, which are found within a critical follow-up recommendation sentence.

Entity	Description
`Reason`	Reason for the critical follow-up recommendation, for example an observation, finding, or diagnosis in the text of the sentence
`Test`	The imaging test or clinical exam that is recommended for follow-up
`Time`	The recommended timeframe for the recommended follow-up test or exam

Table 6.5: Critical follow-up recommendation entities: `Reason`, `Test`, and `Time`

I defined an event model for critical follow-up recommendations in radiology reports as a template of properties made up of:

- the text of the critical follow-up recommendation sentence
- `Reason`, `Test`, and `Time` entities found in the recommendation sentence (See Table 6.5 for definitions)
- the `index exam` described in the *exam description* metadata field of its radiology report
- the `diagnosis` in the *diagnosis summary* metadata field of its radiology report
- a *default recommended follow-up test* and *default follow-up timeframe* computed from information extracted from report text and metadata and based on rules prescribed by the radiologist

The values for *default recommended follow-up test* and *default recommended follow-up timeframe* were based on a set of rules prescribed by the radiologist annotator in this study. They represent a set of heuristics used in the radiology department to determine a default test or timeframe and are used when no explicit test or timeframe is recommended in the radiology

report. These values are calculated with the expectation they will be part of the metadata provided to the end user of a future critical follow-up recommendation identification system integrated with an EHR.

I developed a template extractor for critical follow-up recommendation events including a rules engine for calculating default test and timeframe. See Section 6.4.6 for details on the implementation.

6.4.3 Annotation

The goal of the first round of annotation was to have medical expert annotators vet system-generated candidate critical follow-up recommendation sentences from a sub-corpus of 8,000 reports randomly selected from the much larger multi-institutional corpus. The sub-corpus was created because manual annotation is a time-consuming and labor-intensive process. The annotators assigned to the project were limited in the time they had available for annotation.

The first step was to import the multi-institutional corpus of X-ray reports from the EHR systems of three different institutions including a university medical center, a trauma center, and a cancer care hospital into the system database. The corpus was made up of 745,077 radiology reports representing all imaging modalities with the exception of Mammography (MG), of which another 37,754 reports were stored in the database but were excluded from the study because a specific follow-up and alert system already exists to manage recommendations for that modality. See Table 6.6 for a distribution of reports across modalities.

Code	Imaging modality	# of reports
CR	Computed Radiography	413,889
CT	Computed Tomography	146,181
DF	Digital Fluoroscopy	12
DX	Digital Radiography	1,626
MR	Magnetic Resonance Imaging	52,127
NM	Nuclear Medicine	12,895
OT	Portable Radiography	6,166
PT	Portable Radiography	4,121
RF	Fluoroscopy	27,239
US	Ultrasound	68,999
XA	Angio-Interventional	11,803
-	No Category	19
	Total	745,077

Table 6.6: Distribution of radiology reports by standard modality code in multi-institutional radiology corpus.

A high-recall/low-precision model for the binary sentence classifier described in Section 6.4.1 was used to identify 1086 candidate critical follow-up recommendation sentences from the 140,044 sentences that make up the 8,000 report multi-institutional sub-corpus. The percentage of candidate sentences identified was approximately .8%. The number of reports with system-identified recommendation sentences was 818 out of a total of 8,000, or approximately 10.2%.

Figure 6.4 provides a diagrammatic view of the flow of data through the entire system and how annotations created during the corpus development phase factor into the eventual identification and labeling of critical follow-up recommendation sentences and events.

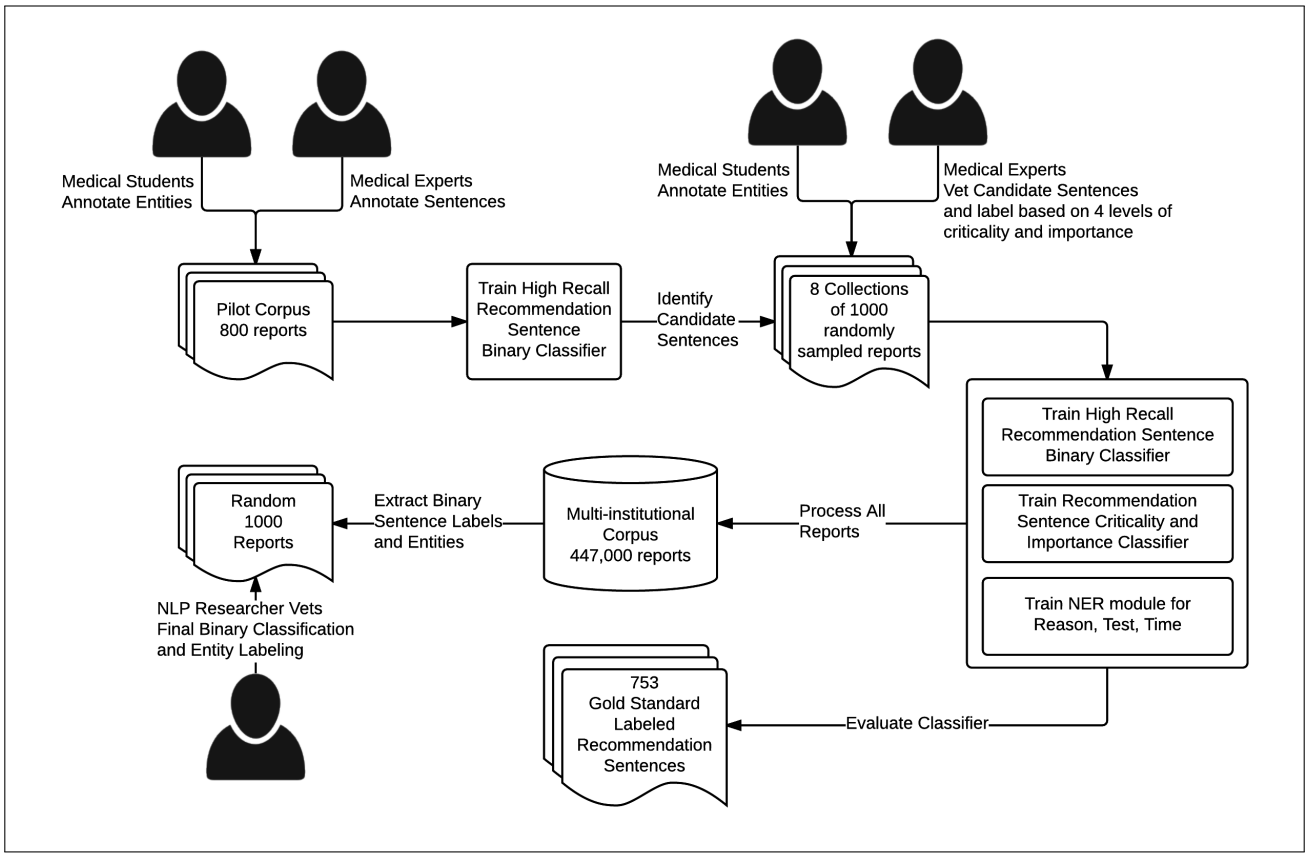

Figure 6.4: Overall system data flow diagram for the corpus development process

131 reports were identified as having system-identified candidate recommendation sentences in the first collection of 1000 randomly selected reports that made up the 8,000 report sub-corpus. To measure IAA, the first 50 of these reports were annotated by three medical expert annotators. After measuring IAA and discussing the results, the annotation guidelines were updated and two of the three annotators completed another round of annotation, reviewing an additional 81 reports from the same collection of 131 reports with system-identified recommendations. A second IAA measure was calculated and discussed by the annotators, leading to the finalization of the annotation guidelines. The two annotators completed the annotation of the 8 collections of 1000 reports, vetting the 1086 candidate sentences identified by the high-recall/low-precision classifier, and creating a final annotated

corpus of 753 critical follow-up recommendation sentences in 567 reports, each labeled with one of the four labels of criticality and importance.

The annotation interface displayed the full text of each report to the annotators and highlighted by color-coded labeling the candidate recommendation sentences. The annotators could then label the sentence as *Incorrect* if they believed the system had wrongly identified a recommendation sentence. They could also label a new recommendation sentence if they believed it had not been identified by the system. If they introduced a new recommendation and the other annotators did not also identify the same sentence, the label for the annotation was considered to be *None* for the other annotators for IAA measures. Although each annotator added one or two additional sentences per collection, there was no agreement and therefore no annotator-labeled sentences were added to the final gold standard collection. See Table 6.7 for sentence statistics of the annotated sub-corpus of 8,000 reports, Table 6.8 for report statistics, and Section 6.4.4 for IAA results at the sentence level.

Collection	**All Sent.**	**Sys. Rec. Sent.**	**1**	**2**	**3**	**4**	**Final Rec Sent.**
1	17,146	188	9	38	29	46	122
2	17,463	154	8	30	23	18	79
3	17,714	163	14	36	30	23	103
4	18,136	173	8	35	31	30	104
5	17,900	95	19	21	27	12	79
6	17,513	110	12	30	23	20	85
7	17,420	96	13	42	17	13	85
8	16,752	107	7	35	42	12	96
Totals	140044	1086	90	267	222	174	753

Table 6.7: Critical follow-up recommendation sentence sub-corpus statistics (categories: (1) IMP_COND, (2) IMP, (3) IMP_M_UN, and (4) UN_IMP)

Collection	**Reports w/Sys. Rec.**	**Reports w/Rec.**
1	132	87
2	116	56
3	122	82
4	125	84
5	75	60
6	86	65
7	80	64
8	82	69
Totals	818	567

Table 6.8: Critical follow-up recommendation sub-corpus statistics for recommendations in reports

The vetting, categorization, and annotation of system-identified critical follow-up recommendation sentences by medical expert annotators took place at the same time medical student annotators were being trained to annotate `Reason`, `Test`, and `Time` entities. Because the medical expert annotators were annotating the multi-institutional corpus, and their annotations needed to be final before entities could be added, entity annotation training, guideline development, and IAA was applied to the smaller 800 report corpus from the Yetisgen-Yildiz et al. (2013b) study. One medical expert annotator and one medical school student annotator annotated 112 critical follow-up recommendation sentences in 89 reports that contained a critical follow-up recommendation sentence, in two rounds of annotation. Due to differences in sentence normalization between the data in the previous study, and its imported representation in the system database, the number of recommendation sentences was 112 in this study, rather than the 113 reported in Yetisgen-Yildiz et al. (2013b). Once the annotators completed two rounds of IAA on the corpus from the Yetisgen-Yildiz et al.

(2013b) study, they completed the annotation of the 753 critical follow-up recommendation sentences in the multi-institutional sub-corpus. See Section 6.4.4 for IAA measures at the entity level. Table 6.9 gives the final entity statistics of the multi-institutional sub-corpus.

Collection	**Reason**	**Test**	**Time**	**Totals**
1	92	129	36	341
2	72	88	16	235
3	87	103	15	283
4	101	116	29	337
5	81	81	14	239
6	76	90	24	241
7	66	93	28	236
8	93	106	16	292
Totals	668	806	178	2204

Table 6.9: Critical recommendation corpus statistics for entities `Reason`, `Test`, and `Time`

6.4.4 IAA measures

Table 6.10 summarizes the first round of IAA for critical follow-up recommendation sentence category labeling, pairwise for three medical expert annotators: an internal medicine specialist, radiologist, and attending neurology clinician. The results of the comparison of annotators reveals the difficult discussions and challenge to achieve consensus on labels for critical follow-up recommendation sentences. Given the poor agreement between all three annotators, the pair with the highest agreement continued to the next round of annotation. Agreement was still difficult to achieve, and the results for the second round in Table 6.11 reflect that. The IAA scores for sentence classification based on the labels of criticality and importance are much lower than the binary sentence identification IAA scores from the previous study (Yetisgen-Yildiz et al., 2013b), suggesting that identifying a general critical follow-up recommendation sentence is much easier and more straightforward than deciding

on its level of importance and criticality.

Annotator pairwise	P	R	F1	Acc	Kappa
A/B	35.2	24.0	28.5	35.5	20.4
A/C	51.7	69.1	59.1	63.6	49.5
B/C	42.9	36.8	39.3	46.2	33.1
AVG	43.3	43.1	42.3	48.4	34.3

Table 6.10: First round of critical follow-up recommendation annotation (first 50 reports with system-identified critical follow-up recommendation sentences)

Annotator pairwise	P	R	F1	Acc
A/C	59.8	58.7	59.2	54.5

Table 6.11: Second round of critical follow-up recommendation annotation (remaining 81 reports with system-identified critical follow-up recommendation sentences in collection of 131 reports for the first 1000 reports)

Table 6.12 and Table 6.13 list the results for the first and second round of named entity annotation on the corpus of 89 reports with 112 gold standard critical follow-up recommendation sentences from Yetisgen-Yildiz et al. (2013b). The results of the two rounds of entity annotation are better than the critical follow-up recommendation sentence classification IAA results. The task of identifying `Reason`, `Test`, and `Time` is easier than applying a criticality and importance label to a critical follow-up recommendation sentence.

	TP	FP	FN	P	R	F1
Reason	516	298	433	63.4	54.4	58.5
Time	43	8	10	84.3	81.1	82.7
Test	248	38	86	86.7	74.3	80.0
F-micro	807	344	529	70.1	60.4	64.9
F-macro						73.7
Kappa						17.4

Table 6.12: Round one IAA per token results for critical follow-up recommendation named entities

	TP	FP	FN	P	R	F1
Reason	694	169	218	80.4	76.1	78.2
Time	46	15	3	75.4	93.9	83.6
Test	302	24	57	92.6	84.1	88.2
F-micro	1042	208	278	83.4	78.9	81.1
F-macro						83.3
Kappa						46.3

Table 6.13: Round two IAA per token results for critical follow-up recommendation named entities

6.4.5 NER module

The 2204 annotations of `Reason`, `Test`, and `Time` entities in the multi-institutional sub-corpus of 753 critical follow-up recommendation sentences were used to train and test an NER module. The module integrated version 3.5.2 of the conditional random fields (CRF)-based Stanford Named Entity Recognizer[3] (Finkel et al., 2005) for training and decoding NE se-

[3]http://nlp.stanford.edu/software/CRF-NER.shtml

quences in critical follow-up recommendation sentences. The NER module was evaluated using 5-fold cross-validation in a number of configurations, including:

- all entities
- each entity on its own
- phrasal versus word-based entities (`Reason` vs `Test` and `Time`)

The results of the evaluation of the NER module are presented in Section 6.5.2. The module was used to extract `Reason`, `Test`, and `Time` entities from the entire multi-institutional corpus. See Section 6.5.3 NER results on the entire corpus.

6.4.6 Template extractor module

The template extractor module extracts all of the properties of a critical follow-up recommendation event and converts the properties into a typed feature vector for testing and training the critical follow-up recommendation event classification module using the MALLET machine learning toolkit. Table 6.16 describes how critical follow-up recommendation event properties are converted into features. Tables 6.14 and 6.15 list the rules that are used to compute the *default recommended follow-up test* and *default recommended follow-up timeframe*.

The *default recommended follow-up test* and *default recommended follow-up timeframe* are also available in a report form for each critical follow-up recommendation sentence, for example:

```
RECOMMENDATION(IMPT_CONTINGENT_REC):
The examination should be ordered as a low dose nodule follow-up chest CT.
Date: 2011-03-31 Test/Exam:
RECOMMENDED FOLLOW-UP: low dose nodule follow-up chest CT + CT, CT-PET, or MRI
RECOMMENDED TIMEFRAME: TARGET: 2011-04-30 RANGE: 2011-04-20 - 2011-05-10
```

The report form is generated in anticipation of integrating with a future EHR reporting system for critical follow-up recommendations.

Location	Index Exam	Body Region	Test	Rec Test
Emergency	x	Chest	-	x + CT
Emergency	x	Abdomen	-	x + CT
Emergency	-/x	Chest	y	y + CT
Emergency	-/x	Abdomen	y	y + CT
All	x	Chest	-	x + CT
All	-/x	Chest	y	y + CT
All	x	Abdomen	-	x + CT, CT-PET, or MRI
All	-/x	Abdomen	y	y + CT, CT-PET, or MRI
All	x	All	-	x
All	-/x	All	y	y
All	-	Chest	-	CT
All	-	Abdomen	-	CT
All	-	Pelvis	-	CT

Table 6.14: Rules for *default recommended follow-up test*. The symbol `x` represents the value of an index exam and `y`, the value of the recommended `Test` entity present in the sentence. The symbol `-` indicates the value is missing. Rules are listed by precedence

Location	Body Region	Time	Rec Time	Rec Time Start	Rec Time End
All	All	x	x	x - 1/3	x + 1/3
Emergency	Spine	-	0-3 days	0 days	4 days
Emergency	Bone	-	0-10 days	0 days	10 days
Emergency	Trauma Series	-	0-1 day	0 days	1 day
Emergency	Chest	-	0-2 months	0 days	2 months
Emergency	Abdominal	-	0-2 months	0 days	2 months
In/Out	Lung Lesions	-	3 months	2 months	4 months
All	Thyroid	-	6 months	4 months	8 months
All	All	-	30 days	20 days	40 days

Table 6.15: Rules for *default recommended follow-up timeframe*. The symbol `x` represents the value of a `Time` entity present in the sentence. The rules are listed by precedence.

Feature	Feature Description	Feature Example
Report and sentence text	Unigrams of tokenized text from report or sentence	`unigram=FOLLOW-UP` `unigram=EXAM`
Named entities	`Reason`, `Test`, and `Time` entities from within the boundaries of the critical follow-up recommendation sentence	`test=FOLLOW-UP_MRI` `time=3_WEEKS`
Index exam	Unigram sequence of tokenized index exam description from report metadata. Root type is first token in array	`index_test=MRI unigram=MRI` `unigram=CHEST`
Diagnosis	Unigram sequence of tokenized diagnosis summary from report metadata	`unigram=TRAUMATIC` `unigram=LUNG unigram=INJURY`
Default recommended follow-up test	a default recommended follow-up test or exam, derived from rules and extracted information in the report and its metadata	default_test=MRI
Default recommended follow-up timeframe	a default follow-up timeframe for the test or exam, derived from rules and extracted information in the report and its metadata	default_timeframe=3_WEEKS

Table 6.16: Features derived from the properties of the critical follow-up recommendation event

6.4.7 Critical follow-up recommendation sentence identification system

The critical follow-up recommendation sentence and `Reason`, `Test`, and `Time` entity annotations created for the multi-institutional sub-corpus were used to create a new binary sentence classifier and NER module to run against the entire multi-institutional corpus. See Section 6.5.3 for the results of running these modules against the 737,077 reports in the multi-institutional corpus (minus the 8,000 reports in the sub-corpus).

6.4.8 Critical follow-up recommendation classification system

In the event analysis phase, I defined a critical follow-up recommendation event as a template of properties and in Section 6.4.6, I described how I created a template properties extractor to extract them from critical follow-up recommendation sentences, report text, and metadata. See Table 6.16 for a description of each of the template properties and how they are extracted in the template extractor. In this section I describe how the feature vectors from the extractor are used in a classification system for labeling critical follow-up recommendation sentences.

I used the MALLET machine learning toolkit and its MaxEnt classifier to train and test, in 5-fold cross-validation, a critical follow-up recommendation sentence classification system using the 753 gold standard critical follow-up recommendation sentences from 567 reports in the multi-institutional sub-corpus. To determine the impact on performance of using critical follow-up recommendation event-based features, I ran baseline word unigram, unigram plus bigram, and unigram plus bigram and trigram experiments. I then ran experiments using the critical follow-up recommendation event-based features alone and in combination with the baseline. See Section 6.5.4 for results of the critical follow-up recommendation sentence classification experiments.

6.5 Results

Section	Description
Data-balancing	K partition data-balancing experiments trained and tested on annotated previous study data
NER evaluation	NER experiments on the multi-institutional sub-corpus for `Reason`, `Test`, and `Time`
Sentence identification & NER statistics (entire corpus)	Results of binary critical follow-up recommendation sentence identification and NER module to entire multi-institutional corpus.
Event classification	Critical follow-up recommendation event classification experiments integrating features extracted from event templates

Table 6.17: Results section overview

All experiments in this study are limited by the amount of data available for training and testing. In all studies, unless otherwise noted, 5-fold cross-validation is used. Each of the five folds is broken down into a 80%/20% split of training and testing data. There is no data overlap between testing and training splits, and each fold's test split does not overlap with any other fold's test split. See Table 1 for a list of the abbreviations and terms used in the result tables headers.

6.5.1 Data-balancing experiments

Table 6.18 lists a select number of the rows of k-partition data-balancing experiments replicated with data from the corpus of 800 radiology reports described in Yetisgen-Yildiz et al. (2013b). The number of sentences, both negative and positive, differed from the original study, due to line ending interpretation and sentence normalization after import into the system database. In the original study, the k-partition data-balancing experiments were run on

113 positive sentences and 18,634 negative sentences, for a ratio of 1:165 (165 k partitions). For this study, 112 positive and 20,887 negative sentences with a ratio of 1:186 (186 k partitions) were used to run data-balancing experiments and select a high-recall/low-precision model. The k partition 5 was selected because it closest to a threshold of 90.0 recall and had a higher precision and F-score than k partition 4.

K	TP	FP	TN	FN	P	R	F1	TOTAL
1	106	461	20314	6	18.7	94.6	31.2	20887
2	103	310	20465	9	24.9	92.0	39.2	20887
3	102	259	20516	10	28.3	91.1	43.1	20887
4	101	209	20566	11	32.6	90.2	47.9	20887
5	101	187	20588	11	35.1	**90.2**	50.5	20887
6	100	174	20601	12	36.5	89.3	51.8	20887
7	98	126	20649	14	43.8	87.5	58.3	20887
8	98	104	20671	14	48.5	87.5	62.4	20887
9	96	95	20680	16	50.3	85.7	63.4	20887
10	94	84	20691	18	52.8	83.9	64.8	20887
186	59	15	20760	53	79.7	52.7	63.4	20887

Table 6.18: Replicated data-balancing experiments on the imported previous study corpus, highlighting the row, partition k = 5, which came closest to the target threshold of a 90.0% recall

6.5.2 NER evaluation results

Table 6.19 and Table 6.20 list the results of the critical follow-up recommendation NER module, trained and tested with the NE annotations created on the 753 critical follow-up recommendation sentences identified in the multi-institutional sub-corpus of 8,000 randomly selected reports. In the first two tables in this section, all entities were trained and tested in the same pass. Table 6.19 measures the exact entity match and Table 6.20 exact token

match. Tables 6.21 and 6.21 list the exact entity and token match result for only `Test` and `Time` decoded in the same pass, and Table 6.23 and 6.24 list the results of `Reason`, an entity that is more phrasal in nature than a traditional entity, decoded alone. The results for training and testing word-based entities, `Time` and `Test`, and phrasal entities, `Reason`, separately, did not significantly improve performance.

Entity	TP	FP	FN	P	R	F1
Reason	460	155	158	74.8	74.4	74.6
Test	622	140	184	81.6	77.2	79.3
Time	147	11	31	93.0	82.6	87.5
Totals	1229	306	373	80.1	76.7	78.4

Table 6.19: Performance of NER (all entities) with exact match of **entities** including multi-word

Entity	TP	FP	FN	P	R	F1
Reason	2900	516	471	84.9	86.0	85.5
Test	2334	227	328	91.1	87.7	89.4
Time	298	8	65	97.4	82.1	89.1
Totals	5532	751	864	88.1	86.5	87.3

Table 6.20: Performance of NER (all entities) wth exact match of **tokens** in all entities

Entity	TP	FP	FN	P	R	F1
Test	609	129	197	82.5	75.6	78.9
Time	146	9	32	94.2	82.0	87.7
Totals	755	138	229	84.6	76.7	80.5

Table 6.21: Performance of NER (Time and Test entities only) **entity** exact match

Entity	TP	FP	FN	P	R	F1
Test	2275	158	387	93.5	85.5	89.3
Time	292	5	71	98.3	80.4	88.5
Totals	2567	163	458	94.0	84.9	89.2

Table 6.22: Performance of NER (Time and Test entities only) **token** exact match

Entity	TP	FP	FN	P	R	F1
Reason	461	129	157	78.1	74.6	76.3

Table 6.23: Performance of NER (Reason entity only) **entity** exact match

Entity	TP	FP	FN	P	R	F1
Reason	2807	328	564	89.5	83.3	86.3

Table 6.24: Performance of NER (Reason entity only) **token** exact match

6.5.3 Critical follow-up recommendation sentence identification

Table 6.25 lists the results of k partition data-balancing experiments trained on the sub-corpus of 8,000 annotated reports, and applied to the remaining 737,077 reports of the multi-institutional corpus. There are 753 positive sentences to 140,046 negative sentences, which is a 1:184 ratio, resulting in 184 k partitions. Three results are highlighted, the model with high-recall/low-precision (with at least a 70.0 threshold precision) at K=15, a balanced model (where the delta between precision and recall is lowest: .1) K = 67, and the complete model, where K=184. In the tables that follow, results for all three models are shown for critical follow-up recommendation sentences identified in the overall multi-institutional corpus.

K	TP	TN	FP	Total	FN	Prec	Rec	Delta	F1	Acc
1	748	136971	2322	5	140046	24.4	99.3	74.9	39.2	98.3
2	743	137958	1335	10	140046	35.8	98.7	62.9	52.5	99.0
3	736	138308	985	17	140046	42.8	97.7	54.9	59.5	99.3
4	735	138506	787	18	140046	48.4	97.6	49.2	64.7	99.4
5	730	138619	674	23	140046	52.1	96.9	44.9	67.7	99.5
6	727	138706	587	26	140046	55.3	96.5	41.2	70.3	99.6
7	725	138762	531	28	140046	57.7	96.3	38.5	72.2	99.6
8	722	138822	471	31	140046	60.6	95.9	35.3	74.2	99.6
9	719	138852	441	34	140046	62.0	95.5	33.5	75.2	99.7
10	719	138888	405	34	140046	64.0	95.5	31.5	76.6	99.7
11	718	138913	380	35	140046	65.4	95.4	29.9	77.6	99.7
12	717	138936	357	36	140046	66.8	95.2	28.4	78.5	99.7
13	717	138955	338	36	140046	68.0	95.2	27.3	79.3	99.7
14	715	138975	318	38	140046	69.2	95.0	25.7	80.1	99.7
15	713	138993	300	40	140046	70.4	94.7	24.3	80.8	99.8
16	709	139004	289	44	140046	71.1	94.2	23.1	81.0	99.8
17	708	139019	274	45	140046	72.1	94.0	21.9	81.6	99.8
18	707	139026	267	46	140046	72.6	93.9	21.2	81.9	99.8
19	705	139038	255	48	140046	73.5	93.6	20.1	82.3	99.8
20	704	139052	241	49	140046	74.6	93.5	18.9	82.9	99.8
...										
65	654	139186	107	99	140046	86.0	86.9	0.9	86.4	99.9
66	655	139190	103	98	140046	86.5	87.0	0.5	86.7	99.9
67	655	139194	99	98	140046	86.9	87.0	0.1	86.9	99.9
68	653	139194	99	100	140046	86.9	86.7	0.2	86.8	99.9
69	653	139196	97	100	140046	87.1	86.7	0.4	86.9	99.9
...										
184	617	139246	47	136	140046	92.9	81.9	11.0	87.1	99.9

Table 6.25: K partition data-balancing experiments using n-gram features and the annotated multi-institutional sub-corpus (Accuracy = TP/N)

Table 6.26 lists the number of critical follow-up recommendation sentences identified in the overall multi-institutional corpus by data-balanced k partition model.

Total Reports	Total Sentences	K=15	K=67	K=184
737,077	12,732,812	87,179	63,167	53,943
		0.685%	0.496%	0.424%

Table 6.26: Total critical follow-up recommendation sentences identified by model in overall multi-institutional corpus

Table 6.27 lists the number of reports by modality for reports with 1 or more critical follow-up recommendation sentences in the multi-institutional corpus.

	Total Reports	K=15 High-Recall	K=67 Balanced	K=184 Complete
No category	18	3	2	2
CR	409,411	15,658	11,460	9,943
CT	144,626	22,096	17,615	15,563
DF	11	6	-	-
DX	1,603	50	15	12
MR	51,573	6,655	5,203	4,535
NM	12,736	1,532	1,032	859
OT	6,110	94	30	26
PT	4,086	1,681	1,410	1,272
RF	26,936	963	567	469
US	68,280	12,249	8,879	7,954
XA	11,687	1,765	969	639
Total	737,077	62,752	47,182	41,274
		8.514%	6.401%	5.600%

Table 6.27: Modality by Report with 1 or more Recommendations

Table 6.28, Table 6.29, and Table 6.30 list the number of entities per modality for the multi-institutional corpus.

Modality	Abbrev.	Time	Test	Reason	Total
No Category	-	-	2	2	4
Computed Radiography	CR	695	16,653	17,035	34,383
Computed Tomography	CT	4,085	29,830	26,382	60,297
Digital Fluoroscopy	DF	-	-	12	12
Digital Radiography	DX	1	17	52	70
Magnetic Resonance Imaging	MR	4,526	12,621	7,028	24,175
Nuclear Medicine	NM	100	1,852	1,608	3,560
Portable Radiography	OT	10	54	91	155
Portable Radiography	PT	83	3,247	2,360	5,690
Fluoroscopy	RF	31	941	1,007	1,979
Ultrasound	US	4,851	15,328	9,452	29,631
Angio-Interventional	XA	912	1,479	1,707	4,098
Totals		15,294	82,024	66,736	164,054

Table 6.28: Number of Entities per Modality: **High-Recall K = 15**

Modality	Abbrev.	Time	Test	Reason	Total
No Category	4	-	2	1	3
Computed Radiography	CR	578	12,623	13,814	27,015
Computed Tomography	CT	3,724	24,398	21,705	49,827
Digital Fluoroscopy	DF	-	-	-	-
Digital Radiography	DX	1	13	21	35
Magnetic Resonance Imaging	MR	3,714	9,761	5,373	18,848
Nuclear Medicine	NM	55	1,353	1,217	2,625
Portable Radiography	OT	10	38	39	87
Portable Radiography	PT	72	2,726	2,034	4,832
Fluoroscopy	RF	23	595	655	1,273
Ultrasound	US	3,112	11,364	7,044	21,520
Angio-Interventional	XA	542	924	919	2,385
Totals		11,831	63,797	52,822	128,450

Table 6.29: Number of Entities per Modality: **Balanced K = 67**

Modality	Abbrev.	Time	Test	Reason	Total
No Category	-	2	1	3	
Computed Radiography	CR	539	11,105	12,625	24,269
Computed Tomography	CT	3,484	21,393	19,655	44,532
Digital Fluoroscopy	DF	-	-	-	-
Digital Radiography	DX	-	11	18	29
Magnetic Resonance Imaging	MR	3,496	8,926	4,944	17,366
Nuclear Medicine	NM	46	1,131	1,071	2,248
Portable Radiography	OT	10	34	35	79
Portable Radiography	PT	71	2,367	1,878	4,316
Fluoroscopy	RF	21	501	561	1,083
Ultrasound	US	2,879	10,286	6,365	19,530
Angio-Interventional	XA	404	643	703	1,750
Totals		10,950	56,399	47,856	115,205

Table 6.30: Number of Entities per Modality: **Complete K= 184**

Table 6.31 and Table 6.32 list the final evaluation results for 100 randomly selected reports from the multi-institutional corpus. Critical follow-up recommendation sentences and named entities were evaluated by a NLP researcher after the critical follow-up recommendation sentence identification and NER module were run against the entirety of the multi-institutional corpus. An NLP researcher completed the final review of categories and entity labels due to the original medical expert annotators being no longer available to the project.

S	G	TP	FP	FN	P	R	F1
183	172	169	14	3	92.3	98.3	95.2

Table 6.31: Final evaluation of 100 randomly sampled reports from entire multi-institutional corpus

Entity	S	G	TP	FP	FN	P	R	F1
Reason	115	121	114	1	7	99.1	94.2	96.6
Test	162	168	157	5	11	96.9	93.5	95.2
Time	50	54	50	0	4	100.0	92.6	96.2
Total	327	343	321	6	22	98.2	93.6	95.8

Table 6.32: Final evaluation of named entities automatically identified from 100 randomly sampled reports from entire multi-institutional corpus

6.5.4 Critical follow-up recommendation event classification experiments

Table 6.33 lists the results of preliminary baseline *n*-gram experiments for critical follow-up recommendation event classification with no feature selection. The feature set for these experiments were *n*-grams created from the text of the 753 critical follow-up recommendation sentences in the gold standard.

Feature Type	TP	FP	FN	P	R	F1
Unigram	458	295	295	58.1	57.1	57.5
Unigram + Bigram	457	296	296	59.2	57.6	**58.2**
Unigram + Bigram + Trigram	458	295	295	59.3	56.9	57.7

Table 6.33: Baseline *n*-gram experiments for critical follow-up recommendation classification

6.5.4.1 Feature selection threshold experiments

In this section, I explore the effect of feature selection thresholds on the performance of the critical follow-up recommendation event classifier. Five sets of experiments based on different combinations of feature types were run. Each set based on feature type included one or more experiments for a specific feature threshold. The same increments of feature threshold were applied across all experiment sets. The standard feature threshold increments for CPIS and

PNA labels on whole documents were: 50, 100, 150, 200, 250, 500, 750, 1000, 1500, and 2500. Three other factors influence the number of feature threshold experiments run for each experiment set: (1) if the maximum number of χ^2 significant features is less than one of the standard feature threshold increments, experiments are not run for thresholds greater than the maximum, (2) if the maximum number of average features per fold in the MaxEnt model with no feature selection is less than a feature threshold, no experiments are run for feature threshold with a greater value, and (3) the F-scores for the larger feature threshold increments demonstrate a pattern of diminishing performance. The row at the top of each table with a hyphen as its value for θ represents an experiment with no feature selection.

Tables 6.34-6.38 list a series of critical follow-up recommendation classification experiments using different combinations of baseline and event-based feature types and feature selection thresholds. The best performing result is listed in Table 6.38, which used a reduced feature set of unigram plus bigram plus metadata location and diagnosis with a feature threshold of 500. The ambiguous boundaries of the four label category for criticality and importance make it difficult to create a well-categorized corpus, especially given the small number of sentences per category in the gold standard. However, the use of metadata from the event properties template and feature selection improved the performance of the classifier.

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	753	753	469	284	284	63.0	58.1	59.8	62.3
1500	753	753	485	268	268	64.1	61.0	62.2	64.4
1000	753	753	489	264	264	64.8	61.6	62.8	64.9
750	753	753	463	290	290	62.6	58.7	60.1	61.5
500	753	753	459	294	294	62.2	58.7	60.0	61.0
250	753	753	450	303	303	61.3	58.3	59.4	59.8
150	753	753	403	350	350	56.9	50.7	52.6	53.5
100	753	753	397	356	356	57.1	48.9	49.7	52.7
50	753	753	397	356	356	56.6	49.0	49.7	52.7

Table 6.34: **Baseline** feature experiments with **unigram, bigram, and trigram** features extracted from sentence and report metadata text (Average number of features across folds in model with no feature selection = 15,698/Number of significant χ^2 ranked features = 1202)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	753	753	423	330	330	55.7	50.1	51.2	56.2
500	753	753	435	318	318	60.9	51.8	53.3	57.8
250	753	753	436	317	317	60.4	51.6	53.0	57.9
150	753	753	434	319	319	62.1	51.4	53.0	57.6
100	753	753	425	328	328	58.0	49.9	51.2	56.4
50	753	753	386	367	367	50.8	44.5	45.1	51.3

Table 6.35: **Event-only** features experiments with **entities, computed values, and structured metadata** (Average number of features across folds in model with no feature selection = 3,510/Number of significant χ^2 ranked features = 361)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	753	753	465	288	288	63.1	57.6	59.4	61.8
1500	753	753	484	269	269	63.8	60.7	61.9	64.3
1000	753	753	477	276	276	63.9	60.8	62.0	63.3
750	753	753	460	293	293	62.0	58.8	60.0	61.1
500	753	753	453	300	300	60.6	57.8	58.9	60.2
250	753	753	446	307	307	61.1	57.4	58.7	59.2
150	753	753	395	358	358	55.7	49.0	49.6	52.5
100	753	753	391	362	362	54.7	48.0	48.6	51.9
50	753	753	388	365	365	54.2	48.0	48.5	51.5

Table 6.36: **All** features experiments with **unigram, bigram, and trigram** features extracted from sentence and report metadata text and **entities, computed values, and structured metadata** (Average number of features across folds in model with no feature selection = 19,208/Number of significant χ^2 ranked features = 1563)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	753	753	475	278	278	62.7	59.0	60.3	63.1
750	753	753	497	256	256	65.4	62.2	63.4	66.0
500	753	753	490	263	263	65.0	61.0	62.4	65.1
250	753	753	449	304	304	61.4	58.2	59.5	59.6
150	753	753	401	352	352	56.8	50.0	51.6	53.3
100	753	753	390	363	363	56.2	47.9	47.7	51.8
50	753	753	393	360	360	56.2	48.1	47.9	52.2

Table 6.37: **Reduced all** features experiments with **unigrams, bigrams, and trigrams** extracted from sentence text and **entities, computed values, and structured metadata** (Average number of features across folds in model with no feature selection = 10,246/Number of significant χ^2 ranked features = 727)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	753	753	483	270	270	62.9	59.8	60.9	64.1
750	753	753	506	247	247	66.2	63.6	64.6	67.2
500	753	753	508	245	245	66.4	63.8	64.8	67.5
250	753	753	471	282	282	65.1	59.9	61.7	62.5
150	753	753	429	324	324	59.4	54.4	56.0	57.0
100	753	753	393	360	360	57.0	49.2	50.7	52.2
50	753	753	397	356	356	57.6	48.5	48.1	52.7

Table 6.38: **Reduced all** features experiments with **unigram and bigrams** extracted from sentence text and **two fields of structured metadata: (1) diagnosis, and (2) location** (Average number of features across folds in model with no feature selection = 6,422/Number of significant χ^2 ranked features = 509)

6.6 Discussion

In this section I discuss results and observations of the different phases of my research framework and follow the same section heading and organization structure for discussion as my methodology and results sections.

1) **Event analysis**—analyze the task and related previous research to define events

2) **Corpus development**—develop corpora by annotating events

3) **Event detection**—build systems for automatic event detection

4) **Event-based feature extraction**—extract event-based features for classification

6.6.1 Event analysis

The original data from Yetisgen-Yildiz et al. (2013b) was imported from a MySQL batch script into a Microsoft SQL Server database. Meta-characters and line-endings were incorrectly mapped from one encoding to another and the result was a difference in how sentences

were normalized in the system database. The mapping from sentences to categories was close enough that the final set of gold standard sentences numbered 112 rather than 113. Manual review of the gold standard sentences in the previous study data and the table in the database verified that the all of the text in the original 113 sentences was represented in the 112 sentences in the database. The data error did not impact the development of the high-recall/low-precision model developed for sentence identification.

6.6.2 Corpus development

Creating the four label criticality and importance category for critical follow-up recommendation sentences proved to be the biggest challenge in the study. In initial discussions with the two medical expert annotators, both of whom participated in the original (Yetisgen-Yildiz et al., 2013b) study, a seven-label, hierarchical category for critical follow-up recommendation was suggested. The seven-label category was difficult to agree on and apply to examples of critical follow-up recommendation sentences identified by the binary classifier. The task was further complicated by the difference in interpretation of recommendation language by a clinician and a radiologist. The addition of a third medical expert voice, an attending physician in neurology, helped to clarify some of the subtle differences in interpretation and adjudicate decisions regarding what constitutes a critical follow-up recommendation. Eventually, we were able to constrain the original seven label category for critical follow-up recommendation to the current four label category.

Due to the subtle differences between category labels in our four label critical follow-up recommendation classification scheme, several follow-up meetings and measurements of agreement were made after the guidelines were finalized and annotation of the corpus proper had begun. This is not typical of a traditional annotation process (Xia and Yetisgen-Yildiz, 2012). We continued to use double annotation for the first three sets of 1000 reports, comparing annotations and reaching consensus on not only the final definitions of the four category labels, but on a set of heuristics for applying the labels based on actual report content. Another reason for the follow-up IAA measures and discussions was the length of time between

annotation sessions. All of the medical experts in our study were fully employed and working the long hours of the healthcare professional. Finding time in their schedule to dedicate to manual annotation was difficult.

The named entities in the multi-institutional corpus were annotated by two annotators. Initial NER IAA results and a review of annotations uncovered a number of annotation errors and inconsistencies.

Errors included:

- inclusion of punctuation in non-sentence text spans
- errors generated in the annotation environment, such as starting a text span with a space
- overlapping text spans where they are not allowed by the schema
- additional words or spaces added to the text span

Inconsistencies included:

- repeated phrases or expressions within a recommendation sentence were not annotated consistently across reports
- words were not included in a repeated phrasal text span, such as `Reason`, whereas the same words were repeated in other reports
- the annotator overlooked recommendation sentences that should have been annotated for entities but were not

All of these errors and inconsistencies were resolved in follow-up meetings and addressed explicitly in the annotation guidelines.

6.6.3 Event detection and event feature extraction

In the event detection phase of the study, I created a three-stage process for detecting critical follow-up recommendation events:

1) Binary critical follow-up recommendation classification Binary classification models were trained on sentences in the annotated corpus. A high-recall/low-precision classifier was selected and run against unseen radiology reports and candidate critical follow-up recommendation sentences were extracted.

2) Named Entity Recognition A named entity recognition module, trained on critical follow-up recommendation sentences annotated with `Reason`, `Test`, and `Time` entities in the annotated corpus, was run against the candidate critical follow-up recommendation sentences output from the previous stage.

3) Critical follow-up recommendation event classification A critical follow-up recommendation event MaxEnt classifier was trained on vetted, labeled candidate sentences and named entities from stage (1) and (2), as well as other extracted properties, and run on candidate critical follow-up recommendation sentences from the previous two stages.

6.6.4 Critical follow-up recommendation event classification error analysis

The distribution of critical follow-up recommendation sentences across the importance and criticality category labels was fairly balanced (See Table 6.39 for a summary of sentences to labels) with the exception of the *contingent clinically important* class, which was represented by only 90 sentences.

Number	Abbrev	Label	# of sentences
1	IMP	Non-contingent clinically important recommendation	267
2	IMP_COND	Contingent clinically important recommendation	90
3	IMP_M_UN	Clinically important recommendation likely reported (miss unlikely)	222
4	UN_IMP	Clinically unimportant recommendation	174

Table 6.39: Number of sentences per critical follow-up recommendation important and criticality labels

The performance of the classifier varied per label. Tables 6.40 and 6.41 provide examples of detailed results by class. Table 6.42 lists the label per class distribution of sentences for the best performing experimental settings.

Label	TP	FP	FN	P	R	F1
1 IMP	198	113	69	63.7	74.2	68.5
2 IMP_COND	34	28	56	54.8	37.8	44.7
3 IMP_M_UN	126	119	96	51.4	56.8	54.0
4 UN_IMP	111	24	63	82.2	63.8	71.8
Total	469	284	284	63.0	58.1	59.8

Table 6.40: Detailed per label results for **baseline** feature experiments with **no feature selection threshold**

Label	TP	FP	FN	P	R	F1
1 IMP	208	98	59	68.0	77.9	72.6
2 IMP_COND	42	28	48	60.0	46.7	52.5
3 IMP_M_UN	126	102	96	55.3	56.8	56.0
4 UN_IMP	113	36	61	75.8	64.9	70.0
Total	489	264	264	64.8	61.6	62.8

Table 6.41: Detailed per label results for **baseline** feature experiments where **feature selection threshold=1000**

Label	TP	FP	FN	P	R	F1
1 IMP	211	85	56	71.3	79	75
2 IMP_COND	42	33	48	56	46.7	50.9
3 IMP_M_UN	137	97	85	58.5	61.7	60.1
4 UN_IMP	118	30	56	79.7	67.8	73.3
Total	508	245	245	66.4	63.8	64.8

Table 6.42: Detailed per label results for **reduced all feature** experiments with **unigram and bigram** features extracted from the sentence and **structured metadata for diagnosis and location** and where feature selection threshold=500

The *2 IMP_COND* and *3 IMP_M_UN* labels consistently underperformed in classification when compared to the *1 IMP* and *4 UN_IMP* labels. They were also the most difficult to get annotators to agree on during the annotation phase. With a limited amount of training data, common place conditional phrases may not be classified correctly. For example, in the baseline experiment, the sentences: *In a high risk patient, follow-up CT is suggested in 6-12 months for further evaluation* and *Correlation with MRI is recommended for further evaluation as clinically warranted* which are *IMP_COND* sentences were mislabeled as *IMP*. Label 3 is particularly difficult to pattern. The same version of the classifier labeled the

following *IMP_M_UN* sentence: *BI-RADS category 4. Given the low probability of accurately localizing this finding with ultrasound, an MR guided biopsy is recommended at this time.* and *IMP.*

The confusion matrices in Tables 6.43 - 6.41 provide an insight into class labeling errors in the corpus.

	1 IMP	**2 IMP_COND**	**3 IMP_M_UN**	**4 UN_IMP**
1 IMP	198	9	50	10
2 IMP_COND	22	34	33	1
3 IMP_M_UN	66	17	126	13
4 UN_IMP	25	2	36	111

Table 6.43: Confusion matrix for **baseline** experiments with **no feature selection**

	1 IMP	**2 IMP_COND**	**3 IMP_M_UN**	**4 UN_IMP**
1 IMP	208	11	39	9
2 IMP_COND	17	42	26	5
3 IMP_M_UN	59	15	126	22
4 UN_IMP	22	2	37	113

Table 6.44: Confusion matrix for **baseline** experiments with **optimal feature selection of 1000**

	1 IMP	2 IMP_COND	3 IMP_M_UN	4 UN_IMP
1 IMP	211	11	37	8
2 IMP_COND	15	42	28	5
3 IMP_M_UN	51	17	137	17
4 UN_IMP	19	5	32	118

Table 6.45: Confusion matrix for **reduced all** feature experiments with **unigram and bigram** features extracted from the sentence and **structured metadata for diagnosis and location** and where **feature selection threshold=500**

6.7 Summary

In previous Chapters, I introduced a new model for clinical events as dependency trees and developed a pipeline of modules to identify and extract them. My goal was to measure the impact of event-based features on two applied clinical NLP tasks: pneumonia and ALI report classification. In this study, I expanded my previous definition of clinical events and introduced a different type of event, one based on a template of properties, with a sentence at its core. I created an annotation schema, guidelines, and a process for annotating a multi-institutional sub-corpus of 8,000 radiology reports across 12 imaging modalities with category labels for critical follow-up recommendation sentences and named entities for `Reason`, `Test`, and `Time` entities. I used the annotations to train and test a binary sentence classifier to identify critical follow-up recommendation sentences, an NER module for critical follow-up recommendation named entities, and an event classification module to classify critical follow-up recommendation events. I applied the binary sentence classifier and NER module against the remaining unlabeled 737,077 reports in the greater multi-institutional corpus to generate statistics for a future study.

Conf.	θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
baseline	1000	753	753	489	264	264	64.8	61.6	62.8	64.9
event-only	500	753	753	435	318	318	60.9	51.8	53.3	57.8
all	1000	753	753	477	276	276	63.9	60.8	62.0	63.3
reduced all I	750	753	753	497	256	256	65.4	62.2	63.4	66.0
reduced all II	500	753	753	508	245	245	66.4	63.8	**64.8**	67.5

Table 6.46: Comparison of optimal feature selection experiments for critical follow-up recommendation event classification

In conclusion, for critical follow-up recommendation event classification, a selection of event-based (diagnosis and location metadata) and baseline n-gram (unigram and bigram) features combined had the highest F-score, 64.8, in feature selection threshold experiments, beating a baseline n-gram (unigram, bigram, and trigram) F-score of 62.8 (see Table 6.46). The binary sentence identification and NER module performed reasonably well. The identification of 62,752 reports with candidate critical follow-up recommendation sentences and 87,179 candidate critical follow-up recommendation sentences, establishes a potential source of new annotations and additions to the existing sub-corpus of 753 identified recommendation sentences.

Chapter 7
DISCUSSION

In this chapter, I summarize the results of my experiments in Section 7.1, and discuss the outcomes of my research into three applied NLP classification tasks within the context of the four phases of my research: event analysis in Section 7.2, corpus development in Section 7.3, event detection in Section 7.4, and event-based feature extraction for classification tasks in Section 7.5.

7.1 Summary of results

In my dissertation I explored applying event-based features in three clinical NLP classification tasks:

1. Pneumonia report classification
2. ALI report classification
3. Critical follow-up recommendation event classification

7.1.1 Pneumonia report classification

Ventilator-associated pneumonia (VAP) is a pneumonia acquired in a hospital setting by patients who are on mechanical ventilation. It is a difficult disease to diagnosis early due to patients with VAP also demonstrating symptoms of either an established critical illness or a traumatic injury that has led them to be admitted to the hospital. In previous research, Xia and Yetisgen-Yildiz (2012), Tepper et al. (2013), and Vanderwende et al. (2013) introduced an annotated corpus, clinical NLP classification system, rationale snippet prediction module, and change-of-state event n-tuple to predict the presence of VAP by classifying reports by Clinical Pulmonary Infection Score (CPIS) and suspicion for pneumonia (PNA) labels. I

replicated their systems and baseline experiments, implemented a new dependency tree-based model for change-of-state and diagnosis events, developed event detection modules, and extracted event-based features to extend the experiments in the original study, and demonstrated a small improvement in performance when compared to the n-gram baseline results I replicated from the descriptions of experiments in their studies.

Table 7.1 lists the best performing final experiments on whole documents with baseline, event-only, and all features. The threshold was set to 250 for all experiments with feature selection. In this scenario, for both CPIS and PNA, all features with feature selection performed best.

Text Boundary	Type	Features	θ	P	R	F1	Acc
Whole Document	CPIS	baseline	250	77.0	71.1	73.4	86.0
		event-only	250	79.6	70.4	73.0	86.7
		all	250	79.7	75.5	**77.2**	87.6
	PNA	baseline	250	74.1	71.7	72.5	80.8
		event-only	250	73.2	69.2	70.2	80.5
		all	250	75.1	72.7	**73.5**	81.4

Table 7.1: Comparison of the best performing final systems for baseline, event-only, and all features on whole documents

Table 7.2 lists the best performing final predicted snippet experiments which compared baseline, event-only, and all features. The threshold was set to 250 for all experiments with feature selection. In this scenario, for CPIS, all features, combining baseline and event-based, with feature selection performed best. For PNA, baseline features with feature selection performed best.

Text Boundary	Type	Features	θ	P	R	F1	Acc
Predicted Snippet	CPIS	baseline	0	76.9	72.4	74.2	86.1
		event-only	0	77.6	70.4	72.7	85.9
		all	250	80.9	73.3	**76.1**	87.4
	PNA	baseline	250	77.5	74.4	**75.1**	83.3
		event-only	250	75.2	71.0	72.2	86.1
		all	250	75.6	73.2	73.8	83.3

Table 7.2: Comparison of the performance of baseline systems with feature selection and without and final system with event-only or all feature sets over the predicted snippets

After conducting feature selection threshold experiments to explore the optimal number of features to select based on feature type for whole documents, the margin of performance gain over the baseline systems narrowed for both CPIS and PNA. Tables 7.3 and 7.4 below list the final classification results using optimal feature selection thresholds to maximize performance.

Features	θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
baseline	150	1341	1341	1171	170	170	79.9	72.6	75.2	87.3
event-only	150	1341	1341	1163	178	178	78.5	72.4	74.3	86.7
all	250	1341	1341	1175	166	166	79.7	75.5	77.2	87.6

Table 7.3: The highest performing feature threshold experiments for the CPIS category on whole documents

Features	θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
baseline	150	1343	1343	1100	243	243	75.6	73.1	73.8	81.9
event-only	1000	1343	1343	1097	246	246	76.0	72.5	73.6	81.7
all	200	1343	1343	1100	243	243	76.0	73.4	**74.3**	81.9

Table 7.4: The highest performing feature threshold experiments for the PNA category on whole documents

7.1.2 ALI report classification

ALI is a critical illness of the lung. Yetisgen-Yildiz et al. (2013a) introduced an annotated corpus and clinical NLP classification system to detect ALI in patient's radiology reports. I replicated their system and baseline experiments, in order to apply the snippet prediction, NER, and dependency parser-based RE modules I trained in the previous pneumonia report classification study in order to extract event-based features and integrate them with replicated n-gram baseline ALI experiments. My goal was to demonstrate the adaptability of the tools trained on the annotated pneumonia report classification corpus, to other similar genres of disease.

Feature Type	θ	TP	TN	FP	FN	P	R	F1	Acc
Uni+Bi+Tri	800	494	833	170	251	74.4	66.3	70.1	75.9
Event-only (whole document)	800	496	845	158	249	75.8	66.6	**70.9**	76.7

Table 7.5: Baseline n-gram results from replicated system (evaluated by report) on whole document feature extraction

I conducted additional feature selection threshold experiments, similar to the previous pneumonia report classification study and the updated performance results were close to the

same value but no longer beat the experimental results with baseline features. Table 7.6 lists the performance results from the feature selection threshold experiments.

Features	θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**	
baseline	2500	1748	1748	527	868	135	218	79.6	70.7	**74.9**	79.8
event-only	1500	1748	1748	513	850	153	232	77.0	68.9	72.7	78.0
all	5000	1748	1748	516	875	128	229	80.1	69.3	74.3	79.6

Table 7.6: The highest performing feature threshold experiments for whole document

7.1.3 Critical follow-up recommendation event classification

A critical follow-up recommendation is a statement made by a radiologist in a given radiology report to advise the referring clinician to further evaluate an imaging finding by either other tests or further imaging (Yetisgen-Yildiz et al., 2013b). In my case study, I introduced a new model for a critical follow-up recommendation event, composed of the recommendation sentence itself and a template of properties that include computed values for default test and timeframe, `Reason`, `Test`, and `Time` entities, and metadata. I created a three-stage critical follow-up recommendation event detection process:

1) Binary critical follow-up recommendation classification Binary classification models were trained on sentences in the annotated corpus. A high-recall/low-precision classifier was selected and run against unseen radiology reports and candidate critical follow-up recommendation sentences were extracted.

2) Named Entity Recognition A named entity recognition module, trained on critical follow-up recommendation sentences annotated with `Reason`, `Test`, and `Time` entities were run against the candidate critical follow-up recommendation sentences output from the previous stage.

3) Critical follow-up recommendation event classification A critical follow-up recommendation event MaxEnt classifier was trained on vetted, labeled candidate sen-

tences and named entities from stage (1) and (2), as well as other extracted properties, and run on candidate critical follow-up recommendation sentences from the previous two stages.

I conducted a series of experiments evaluating the final stage of the event detection process. I demonstrated that the integration of event-based metadata properties and feature selection can improve the performance of the event classification task. Table 7.7 lists the baseline of unigram, bigram, and trigram and the best performing system, unigram plus bigram, plus metadata location and diagnosis features.

Feature Types	θ	TP	FP	FN	P	R	F1
Baseline: Unigram + Bigram + Trigram	1000	489	264	264	64.8	61.6	62.8
uni +bi + Loc + Diag	500	508	245	245	66.4	63.8	**64.8**

Table 7.7: Critical follow-up recommendation event classification experiments with combinations of feature types and feature selection thresholds

7.1.4 Summary of results

Results across the three applied tasks were mixed, however, there were a few general conclusions that could be made:

Feature selection In most cases, feature selection improved the performance of the classifier. In the rare cases, such as PNA predicted snippet classification for pneumonia report classification, feature selection caused performance to decrease. The pneumonia report classification experiments were run with the same fixed feature selection threshold of 250 in order to compare the results of all feature type sets (baseline, event-only, and all features). Later, feature selection threshold experiments were run to determine the optimal feature selection threshold for all feature types sets. The comparison of the lists of features ranked by χ^2 and features ranked by the MaxEnt classifier in

the pneumonia report classification study, reveal that for combined PNA baseline and event experiments, there were more event features ranked in the χ^2 feature list. A more detailed comparison of the entire list of ranked features for both the MaxEnt model internal to the classifier and the external χ^2 ranking might reveal an mismatch that impacted feature selection experiments for PNA.

Snippet prediction Snippet prediction worked well for pneumonia report classification baseline feature sets, but did not translate into the ALI genre. Although the list of common terms for pneumonia and ALI radiology reports was similar when analyzed for top 10 or 25 terms, the ALI experiment results suggest the language of observations and diagnosis for ALI did not translate at the rationale snippet level.

Event-based features Event features, with the exception of a minute improvement of .8 hundreds of a percent in the ALI whole document experiments, did not perform well when used alone. Event-features worked best when integrated with the baseline n-gram features. This pattern repeated across all three studies. The lists of ranked features in the pneumonia report classification study reveal that many of the highest ranked event features are entity name and value pairs and simple COS and OBS constructs that contain a surface text string. Because of this, they cluster with unigram features when ranked in combined experiments and serve to *boost* the weight of the unigram representation in the models.

7.2 Event analysis

The replication of baseline systems and experiments was a component of all three applied tasks. Each replicated system was successful in reproducing the experiments described in previous research, but none were able to replicate performance numbers exactly. Given the goal of comparing the performance of event-based features to baseline n-gram feature sets, matching the exact results of the previous study was less important than reproducing similar patterns in the results, such as the continuum of improvement from whole, to predicted snippet, to oracle snippet in the pneumonia report classification task experiments.

The translation of the change-of-state n-tuple into a dependency tree structure benefited both the annotation process and the event extraction process. Annotators were able to use labeled arcs to connect entities in hierarchical relationships within the BRAT annotation tools. The process was much easier for them than filling out slots in an external tuple structure, trying to normalize shared slots between multiple connected change-of-state events on their own. Representing connected change-of-state and diagnosis events as graphs makes it much easier to extract algorithmically. By constraining the graph to a dependency tree, extraction is simplified even further—any off-the-shelf dependency parser can parse the representation.

7.3 Corpus development

The corpus development phase in the pneumonia report classification and critical follow-up recommendation studies represented challenges. While the pneumonia report classification unique snippet corpus was annotated by a single NLP researcher, which made the task easier because it did not rely on anyone else's availability, it did not benefit from the collaboration and sharing of opinions about annotation choices that was so much a part of annotating the critical follow-up recommendation corpus. The annotation of the critical follow-up recommendation corpus benefited from many different opinions and feedback on both the criticality and importance category of labels as well as the entities `Reason`, `Test`, and `Time`. However, scheduling meetings, annotators' availability, and long periods of inactivity impacted the quality and consistency of the actual annotations.

7.3.1 Annotation process and tools

The annotation process, adapted for the clinical domain based on recommendations in Xia and Yetisgen-Yildiz (2012), worked well as a repeatable process across the pneumonia report classification and critical follow-up recommendation applied tasks. The BRAT[1] (Stenetorp

[1]http://brat.nlplab.org/index.html

et al., 2012) annotation tool and file format was easy to install, configure, and customize. The annotators for both the pneumonia report classification and critical follow-up recommendation applied tasks were able to quickly get started and be productive because of the tool's simple interface. The only drawback to using BRAT was its inability to integrate with a database system without extensive customization and because of that, maintaining a separate identity protocol for entities and labeled arcs in the file-based BRAT environment and other identities for the same entities and labeled arcs in the system database proved a difficult platform management issue.

7.3.2 Inter-annotator agreement

The IAA scores reported in the pneumonia report classification and critical follow-up recommendation studies were predictive of issues and challenges with the annotation process. In the first rounds of IAA for entities, in both studies, annotators did not pay close attention to text span boundaries, words or punctuation being accidentally included in an entity, or whether to create multiple entities for a phrase or capture the entire multi-word expression in a single label. Lower IAA scores helped bring attention to differences and encouraged discussion amongst annotators.

The four-label category for criticality and importance used to label critical follow-up recommendation types was problematic throughout the project. Many conversations and efforts to create exemplars and better guidelines did not improve IAA. Post-guidelines and well into annotation, additional rounds of double annotation were conducted to review the consistency of annotation, and the IAA scores consistently predicted lack of consensus between the annotators.

7.4 Event detection

The two-stage process for event extraction developed in the pneumonia report classification study, was straightforward to implement and demonstrated the benefit of off-the-shelf state-of-the-art Open Source NLP software. The combination of BRAT, Stanford CRF-NER, and

the Malt dependency parser, facilitated a simple pipeline with straightforward format and representation conversions.

7.5 Event-based feature extraction

Extracting change-of-state and diagnosis events n-tuple instances from the change-of-state and diagnosis events dependency tree representations involved the creation of a simple graph traversal. The six types of event-based features I developed for the pneumonia report classification and ALI report classification studies were basic manipulations of the n-tuple slots, as well as simple name and value pairs (entities). Linking change-of-state and diagnosis events events across sentences in reports or across reports themselves could be a way of enriching the representation and capturing more explicit information about a patient's changes in state as well as a clinician's evolving diagnosis and observations.

Chapter 8

CONCLUSION

In the introduction to this dissertation I posed two fundamental research questions:

1. What are the components and constraints of a semantic representation that can describe speaker-meaning in a task-specific analysis of events in the clinical domain?

2. How does the analysis contribute to applied NLP tasks in the clinical domain such as classifying radiology reports to detect disease or critical follow-up recommendation identification?

In this study, entities, relations, and schema rules are examples of the *components and constraints of a semantic representation* I used to describe clinical event structures. By *speaker meaning*, I am describing the communication between a radiologist and a clinician or ordering provider concerning the radiologist's observations and interpretations of imaging data. By *task specific*, I am referring to an analysis of events constrained to a specific clinical task or domain, for example, pneumonia report classification.

To address my research questions I pursued four goals:

1. **Define a set of events for the clinical domain**—I analyzed corpora and NLP systems for three clinical NLP tasks and defined and marked events in clinical text that captured semantic information unique and valuable to the task.
2. **Develop a clinical corpus of event annotations**—I developed, in collaboration with NLP research and medical expert annotators, corpora of event annotations and category labels for event classification that I used to train and test event extraction modules.

3. **Extract event representations from clinical records**—I developed, trained, tested, and evaluated statistical NLP system modules that automatically extract event representations, based on task-specific definitions, from clinical records.
4. **Apply event representations to multiple NLP tasks in the clinical domain**—I used the extracted representations of events as a source of features for three applied tasks: (1) pneumonia report classification, (2) acute lung injury (ALI) report classification, and (3) critical follow-up recommendation identification.

In summary, I analyzed corpora and the experimental results of systems for three clinical NLP tasks and defined three events: (1) change-of-state, (2) diagnosis, and (3) critical follow-up recommendation. Two of the three types, change-of-state and diagnosis events, were designed to describe patient state and radiologists's observations and diagnostic statements in rationale snippets for a pneumonia report classification task. The third event type, critical follow-up recommendations, were designed to describe a critical follow-up recommendation sentence and its associated report properties, such as metadata fields and named entities. Collectively, the sentence and its report properties, are aggregated as an event in order to classify critical follow-up recommendation events by labels of criticality and importance.

I introduced a preliminary directed acyclic graph (DAG) model for the change-of-state event in order to connect labeled entities in an overarching event structure and link descriptions of change of state with observations that precede or follow it in a sentence. The new event structure addressed the challenges of applying the n-tuple model described by Tepper et al. (2013) and Vanderwende et al. (2013) to the annotation task. By using a graph structure instead of a tuple structure, I simplified event annotation and extraction.

I extended the change-of-state event model by decomposing the `Diagnosis` entity into an event tree which was made up of the same entities as the change-of-state event with the exception of its head entity, `Dhead`. I added entities for negation and coordination to constrain the change-of-state and diagnosis event DAG to a dependency tree representation. Realizing the event as a dependency tree made it into a well-known and well-formed structure

that could be easily represented and extracted by off-the-shelf Open Source NLP software.

In the critical follow-up recommendation study, a recommendation and non-recommendation sentence can be very similar. In order to determine which sentence is truly a critical follow-up recommendation, I developed an event structure of related properties that could be mined for features in classification experiments to help differentiate types of critical follow-up recommendation sentences based on their importance and criticality. Unlike the change-of-state and diagnosis event I defined in the pneumonia report classification study, the critical follow-up recommendation event is a collection of properties defined within the context of the metadata of a single EHR.

All three event types try to represent a clinical event as more than just the text of a snippet or a sentence. The change-of-state and diagnosis events consist of a layer of semantic labels attached to words in a rationale snippet and connected by labeled arcs, which form relations understood by the human reader but are not necessarily expressed by the words in the snippet by themselves. The critical follow-up recommendation is associated with properties in metadata and labeled entities that help disambiguate its criticality and importance from other, non-critical follow-up recommendation sentences with similar wording.

I developed, in collaboration with NLP researcher, medical expert, and medical student annotators, two clinical corpora that include event-based annotations: (1) The pneumonia report classification unique rationale snippet corpus, and (2) The critical follow-up recommendation corpus of multi-institutional radiologist reports across 12 modalities. The two annotated event corpora were used to train and evaluate event detection modules for change-of-state, diagnosis, and critical follow-up recommendation events.

The event detection modules were used in a pipeline to extract event-based features for radiology report classification tasks. I conducted a series of experiments for pneumonia and ALI report classification, using change-of-state and diagnosis event event detection modules to compare the performance of baseline n-grams alone, event-based features alone, and a combined set of all features in the classification task. I performed similar experiments for the task of critical follow-up recommendation sentence identification. The pneumonia report

classification (CPIS F-score +2.0 and PNA F-score +.5) and critical follow-up recommendation (F-score +2.0) experimental results indicated an improvement in the performance of report classification when event-based features were included with baseline features. In the ALI report classification task, adding event-based features to the baseline n-gram features for the classification task did not improve the performance of the classifier (F-score -.6), however, these experiments were based on the application of a pipeline of snippet and event detection modules that were trained on snippet and change-of-state and diagnosis event event annotations from the pneumonia report classification unique snippet corpus. An event-based annotation of ALI reports could potentially adapt and improve the pneumonia report classification modules for the specific vocabulary of the ALI task.

8.1 Contribution

In this dissertation, I have demonstrated that event-based features, when combined with other types of features, such as n-grams, can improve the performance of classification experiments on radiology reports. The simple models for events and the tools and processes I introduce to implement them are extensible and adaptable to other applied NLP tasks. Specifically, the introduction of a dependency-tree structure for change-of-state and diagnosis events and a template of report properties for critical follow-up recommendation events contributes concrete examples of clinical events applied to NLP tasks in the clinical domain. Other researchers can leverage the existing event structures for their own experiments on clinical text or adapt and extend the event schemas for different applied NLP tasks.

8.1.1 Event-based annotated corpora

The final version of the pneumonia report classification unique rationale snippet corpus annotated for change-of-state and diagnosis events is a collection of 1008 rationale snippets in the BRAT[1] (Stenetorp et al., 2012) annotation format. I will be releasing the annotated

[1] http://brat.nlplab.org/index.html

corpus, along with annotation guidelines, event-based research framework documentation, and BRAT schema, to the Web[2] after the publication of my dissertation.

The goal of releasing these resources to the Web is to enable other researchers in clinical NLP to extend the change-of-state and diagnosis event structures and apply them to other types of clinical records. The corpus and annotation guidelines can be used to apply change-of-state and diagnosis event events to other clinical text corpora and the schema and research framework documentation can be used to create new event structures based on the analysis of other types of clinical reports.

8.1.2 Event detection tools

The change-of-state and diagnosis event detection tools were written in Java and integrated NLP features of five open-source tools: (1) MALLET version 2.0.7[3] (McCallum, 2002) for maximum entropy (MaxEnt) classification, (2) Stanford CoreNLP version 3.5.2[4] (Manning et al., 2014) for tokenization, lemmatization, and part-of-speech tagging, (3) Stanford conditional random fields (CRF)-based named entity (NE) Recognizer[5] (Finkel et al., 2005) for named entity recognition, (4) MALT dependency parser version 2.7.2[6] (Nivre et al., 2007) for dependency parsing, and (5) MaltEval[7] (Nilsson and Nivre, 2008) for evaluating dependency parses. I plan to release a Java-based package of my NER and dependency parsing change-of-state and diagnosis event detection and extraction tools bundled with the above listed dependencies under an open-source license for other researchers to use, adapt, and extend. I will include models trained on the existing change-of-state and diagnosis event unique rationale snippet corpus for each tool. Using the corpus and annotation guidelines

[2]See the University of Washington BioNLP Lab's website at http://depts.washington.edu/bionlp for download instructions.

[3]http://mallet.cs.umass.edu/

[4]http://stanfordnlp.github.io/CoreNLP/

[5]http://nlp.stanford.edu/software/CRF-NER.shtml

[6]http://www.maltparser.org/

[7]http://www.maltparser.org/malteval.html

for change-of-state and diagnosis event, researchers can annotate their own clinical text and train and test the event-detection tools for their own research tasks.

8.1.3 Research framework for events

In Chapter 3, I provided an overview of a general research framework for integrating events and event-based features into applied NLP classification tasks for the clinical domain. The methodology was broken down into four phases. Each phase was further decomposed into a number of steps that detailed the tasks required to complete the phase. A visual diagram organized the phases and steps into a canonical work flow that can be adapted to a specific task and event analysis. Chapters 4, 5, and 6 each contained a methodology section that described how the general framework was adapted to the individual tasks of each study.

I believe the general framework can be used by other researchers as a simple guide and organizing tool for their own event-based NLP research. I will bundle a simple version of the framework with the release of the change-of-state and diagnosis event annotated corpus, providing instruction for other researchers to use the framework, annotation schema, and suggested tools to adapt the change-of-state and diagnosis event approach to their own studies.

8.2 Future work

There are both specific and general next steps that I would like to pursue for future research. In the following paragraphs, I address specific next steps relating to one or more of the three case studies in my dissertation or a general theme of all three.

The ALI pneumonia classification experiments showed that using the snippet prediction and event detection modules for the ALI task, which were trained on labeled reports from the pneumonia report classification unique rationale snippet corpus, did not generate features that improved performance. Annotating ALI reports for rationale snippets and change-of-state and diagnosis events and retraining the modules in the event detection pipeline could potentially improve performance. This approach to adaptation of the tools could also apply

to many other types of phenotype or disease detection tasks. An analysis of the vocabulary of change-of-state and diagnosis events across genres or domains could also determine the common vocabulary of change-of-state and diagnosis events and which words and phrases are domain or genre specific.

Bejan et al. (2013a) explored using timeline parameters to define units of classification for pneumonia detection for a cohort of patients, as well as integrating semantically-motivated assertion classification features into their experiments. Change-of-state and diagnosis events express patient state over time. Integrating change-of-state and diagnosis events into patient-centered, timeline-structured experiments for pneumonia report classification, similar to the configuration described in Bejan et al. (2013a) and in combination with assertion classification features, could help evaluate the impact of event-based features on pneumonia detection over time as suggested in (Vanderwende et al., 2013).

The multi-institutional corpus annotated for critical follow-up recommendation sentences, named entities, and criticality and importance is made up of a small number of reports (8,000) when compared to the overall multi-institutional corpus (745,000). The process of using the binary critical follow-up recommendation sentence identifier to select candidate sentences, having medical expert validate and label the sentences for criticality and importance, and medical student annotators annotate the candidate sentences as for entities, as described in Chapter 6, should be repeated for additional collections in the larger overall multi-institutional corpus. Each new collection of reports will contribute to a better understanding of what constitutes a critical follow-up recommendation and the distribution of reports across the four categories of criticality and importance.

The three studies in this dissertation demonstrated that task-specific events can contribute to the description of patient state in clinical reports and improve the performance of applied NLP disease surveillance tasks, such as report classification and critical recommendation sentence identification. The contribution of clinical events to the performance of NLP applications underscores the importance of creating annotated resources that encode the knowledge of radiologists and the meaning of their communication with clinicians. A

layer of semantic annotation that captures this important information can help systems better analyze and understand the condition of the patient over time. Exploring other types of reports in the EHR that contribute to disease surveillance, and defining new semantic structures that contribute information concerning patient state are next steps that I expect to extend my research presented in this dissertation.

BIBLIOGRAPHY

[Albright et al.2013] Daniel Albright, Arrick Lanfranchi, Anwen Fredriksen, William F. Styler IV, Colin Warner, Jena D. Hwang, Jinho D. Choi, Dmitry Dligach, Rodney D. Nielsen, James Martin, Wayne Ward, Martha Palmer, and Guergana K. Savova. 2013. Towards comprehensive syntactic and semantic annotations of the clinical narrative. *Journal of American Medical Informatics Association (JAMIA)*, 20:922–930.

[Basile et al.2012] Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, volume 12, pages 3196–3200. European Language Resources Association (ELRA).

[Bejan et al.2012] Cosmin Adrian Bejan, Fei Xia, Lucy Vanderwende, Mark M Wurfel, and Meliha Yetisgen-Yildiz. 2012. Pneumonia identification using statistical feature selection. *Journal of the American Medical Informatics Association (JAMIA)*, 19(5):817–823.

[Bejan et al.2013a] Cosmin A Bejan, Lucy Vanderwende, Heather L Evans, Mark M Wurfel, and Meliha Yetisgen-Yildiz. 2013a. On-time clinical phenotype prediction based on narrative reports. In *AMIA Annual Symposium Proceedings*, volume 2013, page 103. American Medical Informatics Association.

[Bejan et al.2013b] Cosmin A. Bejan, Lucy Vanderwende, Fei Xia, and Meliha Yetisgen-Yildiz. 2013b. Assertion modeling and its role in clinical phenotype identification. *Journal of American Medical Informatics Association (JAMIA)*, 46(1):68–74.

[Bejček et al.2013] Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague dependency treebank 3.0.

[Bender et al.2015] Emily M Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015)*, pages 239–249.

[Berland et al.2010] Lincoln L Berland, Stuart G Silverman, Richard M Gore, William W Mayo-Smith, Alec J Megibow, Judy Yee, James A Brink, Mark E Baker, Michael P

Federle, W Dennis Foley, et al. 2010. Managing incidental findings on abdominal ct: white paper of the acr incidental findings committee. *Journal of the American College of Radiology*, 7(10):754–773.

[Björne and Salakoski2011] Jari Björne and Tapio Salakoski. 2011. Generalizing biomedical event extraction. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 183–191. Association for Computational Linguistics.

[Björne et al.2009] Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting Complex Biological Events with Rich Graph-based Feature Sets. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, BioNLP '09, pages 10–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Björne et al.2010] Jari Björne, Filip Ginter, Sampo Pyysalo, Jun'ichi Tsujii, and Tapio Salakoski. 2010. Complex event extraction at PubMed scale. *Bioinformatics*, 26(12):i382–i390.

[Cai et al.2016] Tianrun Cai, Andreas A Giannopoulos, Sheng Yu, Tatiana Kelil, Beth Ripley, Kanako K Kumamaru, Frank J Rybicki, and Dimitrios Mitsouras. 2016. Natural language processing technologies in radiology research and clinical applications. *Radio-Graphics*, 36(1):176–191.

[Chang and Lin2011] Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

[Chapman and Cohen2009] Wendy W Chapman and K Bretonnel Cohen. 2009. Current issues in biomedical text mining and natural language processing. *Journal of biomedical informatics*, 42(5):757–759.

[Chapman et al.2011] Brian E Chapman, Sean Lee, Hyunseok Peter Kang, and Wendy W Chapman. 2011. Document-level classification of ct pulmonary angiography reports based on an extension of the context algorithm. *Journal of biomedical informatics*, 44(5):728–737.

[Cunningham et al.2011] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. *Text Processing with GATE (Version 6).*

[Cunningham et al.2013] Hamish Cunningham, Valentin Tablan, Angus Roberts, and Kalina Bontcheva. 2013. Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLoS computational biology*, 9(2):e1002854.

[Demner-Fushman et al.2009] Dina Demner-Fushman, Wendy W Chapman, and Clement J McDonald. 2009. What can natural language processing do for clinical decision support? *Journal of biomedical informatics*, 42(5):760–772.

[Finkel et al.2005] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*, pages 363–370, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Franzén et al.2002] Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Lidén, and Joakim Cöster. 2002. Protein names and how to find them. *International journal of medical informatics*, 67(1):49–61.

[Glavan et al.2011] Bradford J Glavan, Tarah D Holden, Christopher H Goss, R Anthony Black, Margaret J Neff, Avery B Nathens, Thomas R Martin, and Mark M Wurfel. 2011. Genetic variation in the fas gene and associations with acute lung injury. *American journal of respiratory and critical care medicine*, 183(3):356–363.

[Hendee et al.2010] William R Hendee, Gary J Becker, James P Borgstede, Jennifer Bosma, William J Casarella, Beth A Erickson, C Douglas Maynard, James H Thrall, and Paul E Wallner. 2010. Addressing overutilization in medical imaging. *Radiology*, 257(1):240–245.

[Jonnalagadda and Gonzalez2010] Siddhartha Jonnalagadda and Graciela Gonzalez. 2010. Can distributional statistics aid clinical concept extraction. In *Proceedings of the 2010 i2b2/VA workshop on challenges in natural language processing for clinical data.*, Boston, Massachusetts, USA. i2b2.

[Kim et al.2003] Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

[Kim et al.2008] Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10.

[Kim et al.2009] Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 Shared Task on Event Extraction. In

Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task, BioNLP '09, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Kulick et al.2004] Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Pete White. 2004. Integrated annotation for biomedical information extraction. In *HLT-NAACL 2004 Workshop: BioLINK 2004, Linking Biological Literature, Ontologies and Databases*, pages 61–68, Boston, Massachusetts, USA, May 6. Association for Computational Linguistics.

[Luo et al.2016] Yuan Luo, Özlem Uzuner, and Peter Szolovits. 2016. Bridging semantics and syntax with graph algorithms—state-of-the-art of extracting biomedical relations. *Briefings in bioinformatics*, page bbw001.

[Maamouri and Bies2004] Mohamed Maamouri and Ann Bies. 2004. Developing an arabic treebank: Methods, guidelines, procedures, and tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based languages*, pages 2–9. Association for Computational Linguistics.

[Manning et al.2014] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.

[Marcus et al.1993] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

[McCallum2002] Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

[McClosky et al.2011] David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1626–1635. Association for Computational Linguistics.

[Miltsakaki et al.2004] Eleni Miltsakaki, Rashmi Prasad, Aravind K Joshi, and Bonnie L Webber. 2004. The penn discourse treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. European Language Resources Association (ELRA).

[Miwa et al.2010] Makoto Miwa, Rune S\a etre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010. Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology*, 8(01):131–146.

[Murtola et al.2013] Sanna Salanterä Murtola, Hanna Suominen, David Martinez, Noemie Elhadad, Sameer Pradhan, Guergana Savova, and Wendy W Chapman. 2013. Task 2: ShARe/CLEF eHealth Evaluation Lab 2013.

[Nilsson and Nivre2008] Jens Nilsson and Joakim Nivre. 2008. MaltEval: an Evaluation and Visualization Tool for Dependency Parsing. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA).

[Nivre et al.2007] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

[Nivre2006] Joakim Nivre. 2006. Constraints on non-projective dependency parsing. In *11th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

[Palmer et al.2005] Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

[Poon and Vanderwende2010] Hoifung Poon and Lucy Vanderwende. 2010. Joint Inference for Knowledge Extraction from Biomedical Literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 813–821, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Pradhan et al.2013] Sameer Pradhan, Noemie Elhadad, Brett R South, David Martinez, Lee Christensen, Amy Vogel, Hanna Suominen, Wendy Chapman, and Guergana Savova. 2013. Task 1: ShARe/CLEF eHealth evaluation lab 2013. *Online Working Notes of CLEF, CLEF*, 230.

[Riedel et al.2009] Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A Markov Logic Approach to Bio-molecular Event Extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, BioNLP '09, pages 41–49, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Roberts et al.2009] Angus Roberts, Robert Gaizauskas, Mark Hepple, George Demetriou, Yikun Guo, Ian Roberts, and Andrea Setzer. 2009. Building a semantically annotated corpus of clinical texts. *Journal of biomedical informatics*, 42(5):950–966.

[Rubenfeld et al.2005] Gordon D Rubenfeld, Ellen Caldwell, Eve Peabody, Jim Weaver, Diane P Martin, Margaret Neff, Eric J Stern, and Leonard D Hudson. 2005. Incidence and outcomes of acute lung injury. *New England Journal of Medicine*, 353(16):1685–1693.

[Saurı et al.2006] Roser Saurı, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. TimeML annotation guidelines, version 1.2.1, January.

[Shortliffe and Blois2006] Edward H Shortliffe and Marsden S Blois. 2006. The computer meets medicine and biology: emergence of a discipline. In *Biomedical Informatics*, pages 3–45. Springer.

[Stenetorp et al.2012] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April. Association for Computational Linguistics.

[Stubbs et al.2015] Amber Stubbs, Christopher Kotfila, Hua Xu, and Özlem Uzuner. 2015. Identifying risk factors for heart disease over time: overview of 2014 i2b2/uthealth shared task track 2. *Journal of biomedical informatics*, 58:S67–S77.

[Sun et al.2012] Weiyi Sun, Anna Rumshisky, Ozlem Uzuner, Peter Szolovits, and James Pustejovsky. 2012. The 2012 i2b2 temporal relations challenge annotation guidelines. Manuscript available at https://www.i2b2.org/NLP/TemporalRelations/Call.php.

[Sun et al.2013] Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association (JAMIA)*, 20(5).

[Tanabe et al.2005] Lorraine Tanabe, Natalie Xie, Lynne H Thom, Wayne Matten, and W John Wilbur. 2005. Genetag: a tagged corpus for gene/protein named entity recognition. *BMC bioinformatics*, 6(1):1.

[Tepper et al.2013] Michael Tepper, Heather L. Evans, Fei Xia, and Meliha Yetisgen-Yildiz. 2013. Modeling annotator rationales with application to pneumonia classification. In *Expanding the Boundaries of Health Informations Using AI Workshop of AAAI 2013*.

[Uzuner et al.2008] Özlem Uzuner, Ira Goldstein, Yuan Luo, and Isaac Kohane. 2008. Identifying patient smoking status from medical discharge records. *Journal of the American Medical Informatics Association (JAMIA)*, 15(1):14–24.

[Uzuner et al.2010] Özlem Uzuner, Imre Solti, Fei Xia, and Eithon Cadag. 2010. Community annotation experiment for ground truth generation for the i2b2 medication challenge. *Journal of American Medical Informatics Association (JAMIA)*, 17(5):519–23.

[Uzuner et al.2011] Özlem Uzuner, Brent R. South, Shuying Shen, and Scott L. DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of American Medical Informatics Association (JAMIA)*, 18(5):552–6.

[Uzuner et al.2012] Ozlem Uzuner, Andreea Bodnari, Shuying Shen, Tyler Forbush, John Pestian, and Brett R South. 2012. Evaluating the state of the art in coreference resolution for electronic medical records. *Journal of the American Medical Informatics Association (JAMIA)*, 19(5):786–791.

[Uzuner2009] Özlem Uzuner. 2009. Recognizing obesity and comorbidities in sparse data. *Journal of the American Medical Informatics Association (JAMIA)*, 16(4):561–570.

[Vanderwende et al.2013] Lucy Vanderwende, Fei Xia, and Meliha Yetisgen-Yildiz. 2013. Annotating change of state for clinical events. In *Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 47–51, Atlanta, Georgia, June. Association for Computational Linguistics.

[Ware and Matthay2000] Lorraine B Ware and Michael A Matthay. 2000. The acute respiratory distress syndrome. *New England Journal of Medicine*, 342(18):1334–1349.

[Xia and Yetisgen-Yildiz2012] Fei Xia and Meliha Yetisgen-Yildiz. 2012. Clinical corpus annotation: challenges and strategies. In *Proceedings of the Third Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM'2012) in conjunction with the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

[Xia et al.2000] Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Fu-Dong Chiou, Shizhe Huang, Tony Kroch, and Mitchell P Marcus. 2000. Developing guidelines and ensuring consistency for chinese text annotation. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*. European Language Resources Association (ELRA).

[Xu et al.2013] Yan Xu, Yining Wang, Tianren Liu, Junichi Tsujii, I Eric, and Chao Chang. 2013. An end-to-end system to identify temporal relation in discharge summaries: 2012

i2b2 challenge. *Journal of the American Medical Informatics Association (JAMIA)*, 20(5):849–858.

[Yetisgen-Yildiz et al.2010] Meliha Yetisgen-Yildiz, Imre Solti, Fei Xia, and Scott Russell Halgrim. 2010. Preliminary experience with amazon's mechanical turk for annotating medical named entities. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 180–183. Association for Computational Linguistics.

[Yetisgen-Yildiz et al.2011] Meliha Yetisgen-Yildiz, Martin L Gunn, Fei Xia, and Thomas H Payne. 2011. Automatic identification of critical follow-up recommendation sentences in radiology reports. In *AMIA Annual Symposium Proceedings*, volume 2011, page 1593. American Medical Informatics Association.

[Yetisgen-Yildiz et al.2013a] Meliha Yetisgen-Yildiz, Cosmin Bejan, and Mark Wurfel. 2013a. Identification of patients with acute lung injury from free-text chest x-ray reports. In *Proceedings of the 2013 Workshop on Biomedical Natural Language Processing*, pages 10–17, Sofia, Bulgaria, August. Association for Computational Linguistics.

[Yetisgen-Yildiz et al.2013b] Meliha Yetisgen-Yildiz, Martin L Gunn, Fei Xia, and Thomas H Payne. 2013b. A text processing pipeline to extract recommendations from radiology reports. *Journal of biomedical informatics*, 46(2):354–362.

[Yu et al.2011] Shipeng Yu, Faisal Farooq, Balaji Krishnapuram, and Bharat Rao. 2011. Leveraging rich annotations to improve learning of medical concepts from clinical free text. In *AMIA Annual Symposium Proceedings*, volume 2011, page 1603. American Medical Informatics Association.

[Yu et al.2014] Sheng Yu, Kanako K Kumamaru, Elizabeth George, Ruth M Dunne, Arash Bedayat, Matey Neykov, Andetta R Hunsaker, Karin E Dill, Tianxi Cai, and Frank J Rybicki. 2014. Classification of CT pulmonary angiography reports by presence, chronicity, and location of pulmonary embolism with natural language processing. *Journal of biomedical informatics*, 52:386–393.

[Zaidan and Eisner2008] Omar Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 31–40, Honolulu, Hawaii, October. Association for Computational Linguistics.

[Zaidan et al.2007] Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Associ-*

ation for Computational Linguistics; Proceedings of the Main Conference, pages 260–267, Rochester, New York, April. Association for Computational Linguistics.

[Zilberberg and Shorr2010] Marya D Zilberberg and Andrew F Shorr. 2010. Ventilator-associated pneumonia: the clinical pulmonary infection score as a surrogate for diagnostics and outcome. *Clinical Infectious Diseases*, 51(Supplement 1):S131–S135.

Appendix A

CHANGE-OF-STATE, CLINICAL ATTRIBUTE, AND DIAGNOSIS EVENTS FOR CLINICAL RECORDS

In this appendix, I include a modified final version of the annotation guidelines provided to annotators annotating change-of-state and diagnosis events for the pneumonia report classification rationale snippet corpus. The guidelines provide a comprehensive definition of change-of-state and diagnosis events discussed in Section 2.4, Section 4.4.2, and Section 4.4.3.

A.1 Introduction

These guidelines describe a tree-based model, implemented in the BRAT annotation environment, consisting of entities (labeled text spans), attributes (properties of entities), and relations (labeled, directed arcs between entities) for change-of-state, clinical attribute, and diagnosis events in narrative X-ray reports.

The model consists of two types of clinical events: change-of-state and diagnosis. Both are composed of clinical attributes. A clinical attribute is an aspect or characteristic of a patient's condition or disease that has a measurable value and an anatomical location. It can also be the condition or disease itself when referred to within a diagnosis event. A clinical attribute can also stand alone in a snippet and not be contained within the context of a change-of-state or diagnosis event. When a clinical attribute is not contained within a change-of-state or diagnosis event, it is still annotated but its relation to a change-of-state or a diagnosis is unknown and must be inferred based on context.

A change-of-state event describes change in one or more clinical attributes, whereas a diagnosis event relates a diagnosis of a change-of-state event to one or more clinical attributes.

A.1.1 Clinical event tuples

An individual change-of-state or diagnosis event can be represented as a tuple composed of five named, ordered elements or fields, for example a change-of-state tuple consists of the fields `Cos`, `Attr`, `Val`, `Loc`, and `Ref`, and a diagnosis tuple consists of the fields `Dhead`, `Attr`, `Val`, `Loc`, and `Ref`. Every field is optional and can contain a null value, but all clinical event tuples should contain a value for either its `Cos` or `Dhead` field or its `Attr` field. There are rare exceptions when an abbreviated clinical attribute occurs in a sentence in a snippet without a `Cos`, `Dhead`, or `Attr` field. An example is the sentence, The lungs are clear. In this sentence, only a `Loc` entity, lungs, and its value, clear, make up the tuple.

The fields of a clinical event tuple are defined as follows:

`Cos` change-of-state compared to the state of a clinical attribute in previous reports for the same patient (e.g., increased, decreased, worsened, unchanged) or a previously mentioned clinical attribute in a preceding sentence within the snippet.

`Dhead` the head of a diagnosis statement typically indicating a possible or likely diagnosis of a previously mentioned change-of-state. (e.g. likely edema or atelectasis).

`Attr` something doctors are measuring or observing (e.g., volume, opacity) including symptoms or diseases (e.g., edema).

`Val` a possible value for an `Attr` (e.g., clear, low, patchy, diffuse).

`Loc` anatomical location (e.g., lung).

`Ref` a link to the mention of a previous report(s) or observation that the change-of-state is being compared to (e.g., prior examination)

A.1.2 Clinical event entities and relations

Clinical event tuples are automatically generated from a clinical event annotation tree. A clinical event annotation tree consists of event annotations, which are labeled text spans (entities) connected by directed labeled arcs (relations) and composed into a connected hierarchical annotation tree. The names of the labeled text spans (entities) in our clinical event

annotations correspond to the labels defined in our event tuple definitions: `Cos`, `Dhead`, `Loc`, `Attr`, `Val`, and `Ref`. Additional entities, `Conj`, `Versus`, and `Negation`, are used to coordinate multiple entities or negate individual entities when connecting entities in an overarching annotation tree.

A.1.3 Entity annotations

The change-of-state and diagnosis event annotation trees have an entity at their root: `Cos` for change-of-state and `Dhead` for diagnosis. The following sections provide definitions for `Cos` and `Dhead` entities.

Entity	Description
Cos	A text span of one or more words that describe a change-of-state event. Typical Cos entities are past tense forms of verbs like increased or decreased (e.g. consolidation increased), verbal nouns such as increase (e.g. an increase in), or verbal adjectives such as increased (e.g. increased consolidation of), and finally as adverbs such as the word increasingly (e.g. increasingly high consolidation). Cos values can also be temporal terms such as now (e.g. now demonstrates patchy edema) or words that relate to changes over time such as new (e.g. No new focal abnormalities). If Cos is modified, include the modifying word, for example, the word slight in the phrase slight increase in. When selecting text for the Cos text span, articles such as the, an, a (e.g. an increase), copula verbs such as be and existential there (e.g. there is an increase), are not included in the text span.
Dhead	A text span of one or more words that suggest a diagnosis of or interpretation of the previously mentioned change-of-state. Typical Dhead entities are adverbials of possibility, such as possibly, likely (e.g. likely edema or atelectasis). When selecting text for the Dhead text span, copula verbs such as be and existential there I (e.g. there is an increase), are not included in the text span. However, words that modify the Dhead should be included (e.g. possibly a combination of). The Dhead has an optional attribute in the annotation tool that indicates whether the diagnosis statement is a hedge. Include this optional attribute if the language of the dhead is ambiguous and/or the clinical attributes are connected by or conjunctions.

Table A.1: Description of the entities Cos and Dhead

A.1.3.1 Cos attributes

There are 8 attributes that can be applied to a Cos entity. Although some of the attributes are logically mutually exclusive (increased vs decreased), there are no constraints in the annotation tool to express this constraint. Annotators need to ensure they are applied the attributes correctly not to introduce contradictions. Some of the attributes may be used in combination and may imply an additional value. For example, a Cos entity may have

a changed attribute indicating that a Attr value has changed as well. This implies that the change may also warrant an attribute (e.g. edema improved both a changed and an improved attribute).

Attribute	Description
Increased	Implies Changed attribute
Decreased	Implies Changed attribute
Improved	Implies Changed attribute
Worsened	Implies Changed attribute
New	Implies Changed attribute
Stable	Implies No Changed attribute
Persistent	Implies No Changed attribute
Changed	Does not require any other additional attribute but typically accompanied by an additional attribute from list above describing the type of change

Table A.2: Descriptions of attributes for entities Cos and Dhead

A.1.3.2 Dhead attributes

The Dhead has one possible attribute, Hedge. It should be applied to a Dhead if the diagnosis event it heads is ambiguous.

Attribute	Description
Hedge	Not required. Usually applied if alternate or contrasting conjunctions connect diagnosis event entities.

Table A.3: Description of entity Dhead attributes

A.1.4 Clinical attribute

The clinical attribute is composed of three entities: Attr, Val, and Loc. The Attr entity is the root of a clinical attribute and includes one or more Val and Loc entities.

Attribute	Description
Attr	A clinical condition or disease (e.g. edema, atelectasis, pneumonia). Also includes more abstract or general terms referring to symptoms such as consolidation or opacities. Attr entities can demonstrate a change in state or value. They can also be the main subjects of a diagnosis.
Val	A value of a clinical attribute (Attr), typically terms that describe a change over time or a measure (e.g. low, high). They can also be more general terms that describe the state of a clinical attribute (e.g. patchy, diffuse, focal, clear).
Loc	An anatomical location where a clinical attribute is observed. Loc entities can be very general (e.g. lungs) or very specific (partial bilateral lower lobe).

Table A.4: Description of the clinical attribute entities, Attr, Val, and Loc

A.1.5 Reference entity

The Ref entity can be attached to any change-of-state, diagnosis, or clinical attribute entity, however, it is usually attached to the root of the event annotation tree, a Cos in the case of a change-of-state event, a Dhead for diagnosis, and an Attr for a standalone clinical attribute.

Attribute	Description
Ref	A reference to a previous patient report, incident, test, or mentioned condition (e.g. since the prior examination). It can also be a more general reference to a past observation such as again or as before.

Table A.5: Description of entity Ref

A.1.6 Connector entities

The coordinating connector entities that link entities in the event annotation tree are Conj, a conjugation entity that can have four different types of relations: Combine, Alternate, Contrast, and Exclude. These represent the basic types of coordination between similar

entities, and are typically expressed in natural language as and, or, but, and except. Another connector for linking entities is `Versus`, which links an entity that is preferred over another. The relations are `For` and `Against`. This connector is often used in diagnosis statements.

Attribute	Description
`Conj`	A conjunction in natural language that links one or more entities. The relations that are available to a `Conj` entity are: `Combine` (and), `Alternate` (inclusive or), `Contrast` (exclusive or), and `Exclude` (except).
`Versus`	A connector between two entities that prefers one over another. The relations available to this entity are `For` and `Against`.

Table A.6: Description of connector entities `Conj` and `Versus`

A.1.7 Negation

A `Negation` entity can attach to any other entity and represents a negation of the entities it has scope over.

Attribute	Description
`Negation`	An entity that indicates a negation over one or more entities in an annotation tree (e.g. no, without, cannot)

Table A.7: Description of a `Negation` entity

A.1.8 Global attributes

The `Slash_Delimited` attribute can be applied to all entities and it indicates that the entity contains two values that are not explicitly separate in the text and are instead contained within a single slash delimited string (e.g. edema/pneumonia, and/or). This attribute is for processing applications only.

Attribute	Description
Slash_Delimited	Indicates the entities values are in a single slash delimited string (e.g. edema/pneumonia, and/or).

Table A.8: Description of the Slash_Delimited attribute

A.2 Relations

In our annotation trees, entities are connected by relations, which are labeled directed arcs. The arcs between entities form a connected tree structure which describes a clinical attribute, change-of-state event, or diagnosis event.

A.2.1 Clinical attribute relations

The `Attr`, `Loc`, and `Val` entities which make up clinical attribute annotation trees, participate in two basic relations: `Value` and `Location`

`Attribute`	`Description`	`Argument One`	`Argument Two`
`Value` (`Attr`/`Loc`, `Val`)	A directed link between two entities where `Arg_1` represents the entity that has a value and `Arg_2` the `Val` entity that contains a string representing the value.	`Arg_1` in a fully formed clinical attribute is an `Attr`. In abbreviated form it can also be `Loc`. `Conj` and `Versus` can also be `Arg _1` if they coordinate one or more `Attr` and `Loc` entities.	`Arg_2` is a `Val` entity except when a `Conj` or `Versus` entity coordinates one or more `Val` entities.
`Location` (`Attr`, `Loc`)	A directed link between two entities where `Arg_1` represents the source entity and `Arg_2` its anatomical location.	`Arg_1` in a fully formed clinical attribute is an `Attr`. In an abbreviated form of a change of event, `Conj` and `Versus` can also be `Arg_1` if they coordinate one or more `Attr` or `Cos` entities	`Arg_2` is a `Loc` entity except when a `Conj` or `Versus` entity coordinates one or more `Loc` entities.

Table A.9: Description of clinical attribute attributes.

A.2.2 Change-of-state relations

In a complete change-of-state event, the Cos entity participates in one important relation, the State relation. In abbreviated forms, a Cos entity can also participate in the Loc relation if the clinical attribute is in abbreviated form and has no Attr.

Attribute	Description	Argument One	Argument Two
State (Cos, Attr/Loc)	A directed link between two entities where Arg_1 represents an entity that describes a state (Cos) and an Attr or Loc the state describes.	Arg_1 is always a Cos entity or a Conj or Versus entity if they coordinate one or more Cos entities.	Arg_2 in a fully formed change-of-state event is an Attr entity. When a change-of-state event includes an abbreviated clinical attribute, it can also be Loc. Conj or Versus can be in the role of Arg_2 when they coordinate one or more Attr or Loc entities.

Table A.10: Descriptions of State relation

A.2.3 Diagnosis relations

In a complete diagnosis event, the `Dhead` entity participates in one important relation, the `Diag` (diagnosis) relation.

Attribute	Description	Argument One	Argument Two
`Diag` (`Dhead`, `Attr`)	A directed link between two entities where `Arg_1` represents an entity that heads a diagnosis statement (`Dhead`) and one or more `Attr` entities.	`Arg_1` is always a `Dhead` entity or a `Conj` or `Versus` entity if they coordinate one or more `Dhead` entities.	`Arg_2` in a fully formed diagnosis event is an `Attr` entity. `Conj` and `Versus` can be in the role of `Arg_2` when they coordinate one or more `Attr` entities.

Table A.11: Description of `Diag` relation

A.2.4 Global relations

The `Referenced-by` relation can attach to any entity in a clinical attribute, change-of-state event or diagnosis event on one side and a `Ref` entity on the other.

Attribute	Description	Argument One	Argument Two
`Referenced-by` (`Cos`/`Dhead`/`Attr`/ `Loc`/`Val`, `Ref`)	A directed link between two entities where `Arg_1` represents an entity that participates in a clinical attribute or event and `Arg_2` represents a reference to a previous report or attribute.	`Arg_1` can be any entity that participates in a clinical attribute, change-of-state or diagnosis event.	`Arg_2` is a `Ref` entity.

Table A.12: Description of referenced-by relation

A.2.5 Connecting/coordinating relations

The `Conj` and `Versus` entities provide connecting and coordinating relations between entities. Connecting and coordinating relations generalize `Combine` (add), `Alternate` (inclusive or), and `Contrast` (exclusive or) between one or more entities and the `Conj` entity. `Versus` participates in a bi-directional relation of one entity preferred (`For`) over another entity (`Against` relation).

Attribute	Description	Argument One	Argument Two
Combine A and B Combine(and, A) Combine(and, B)	A directed link between two entities establishing a logical and connection.	Arg_1 is a Conj entity.	Arg_2 can be any entity in clinical event structures.
Alternate A or B Alternate(or, A) Alternate(or, B)	A directed link between two entities establishing a logical inclusive or connection.	Arg_1 is a Conj entity.	Arg_2 can be any entity in clinical event structures.
Contrast A but B Contrast(but, A) Contrast(but, B)	A directed link between two entities establishing a logical exclusive or connection.	Arg_1 is a Conj entity.	Arg_2 can be any entity in clinical event structures.
Exclude A except B Combine(except, A) Exclude(except, B)	A directed link between two entities establishing a logical except connection.	Arg_1 is a Conj entity.	Arg_2 can be any entity in clinical event structures.
For A vs. B For(vs., A) Against(vs. B)	A directed link between two entities establishing a preference of the target of the For connection.	Arg_1 is a Versus entity.	Arg_2 can be any entity in clinical event structures.
Against	A directed link between two entities establishing a disfavor for the target of the Against connection.	Arg_1 is a Versus entity.	Arg_2 can be any entity in clinical event structures.

Table A.13: Description of connecting relations

A.2.6 Negation relations

A `Negation` entity can participate in `Negate` relations with all entities in the clinical attribute, change-of-state, and diagnosis event annotation trees.

`Attribute`	`Description`	`Argument One`	`Argument Two`
`Negate` (negation, *)	A directed link between two entities where `Arg_1` represents an entity that negates any entity in `Arg_2`.	`Arg_1` is always a `Negation` entity.	`Arg_2` can be any entity in all events structures: clinical attributes, change-of-state and diagnosis events.

Table A.14: Description of `Negation` relation

A.2.7 Clinical event annotation trees

There are three basic types of clinical event annotation trees: a standalone clinical attribute, a change-of-state event, and a diagnosis event. A clinical attribute may appear in a snippet without being contained within a change-of-state or diagnosis event. In these cases, it can usually be interpreted as being part of an change-of-state event with an unknown `Cos` entity. A change-of-state or diagnosis event annotation tree usually contain one or more clinical attribute annotation trees, however, this can include abbreviated forms of clinical attributes that may only include a connected `Loc` and `Val` entity—the `Attr` entity is missing and must be inferred from the surrounding context of the snippet or the report overall.

A.2.8 Clinical attribute annotation tree

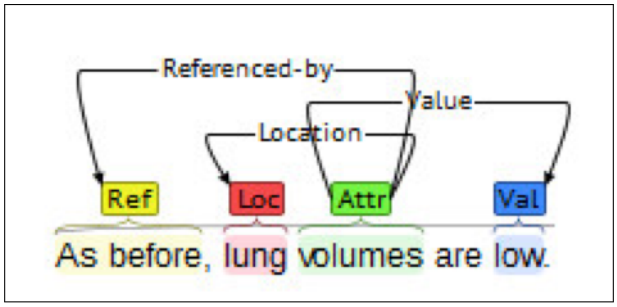

Figure A.1: An example of a complete standalone clinical attribute annotation tree without a top level change-of-state or diagnosis entity

In Figure A.1, the rationale snippet contains a single clinical attribute with no related change-of-state or diagnosis entity. It can be automatically transformed into the tuple below:

```
[Cos: -, Attr: volumes, Val: low, Loc: lung, Ref: As before]
```

A.2.9 Change-of-state event tree

A change-of-state event tree is a connected tree of one `Cos`, and zero or more `Attr`, `Loc`, `Val`, and `Ref` entities that describe a change-of-state in one or more `Attr` or `Loc` entities.

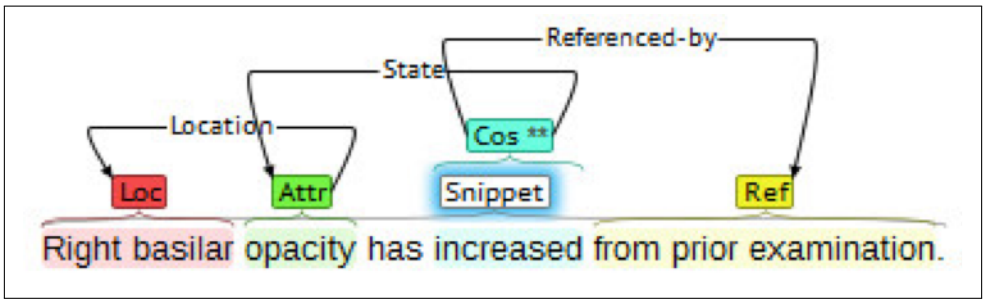

Figure A.2: A change-of-state event

In Figure A.2, the change-of-state event connects a root `Cos` entity to a clinical attribute headed by an `Attr` entity. A `Ref` entity is attached to the `Cos` entity. Unseen in the visualization is the fact that the `Cos` entity has two attributes: changed and increased. The tuple generated from this changed of state event is:

```
[Cos: increased (changed, increased), Attr: opacity, Val: -,
Loc: right basilar, Ref: from prior examination]
```

A.2.10 Diagnosis event

Very often a report will include a diagnosis statement made by physicians; they will often, but not always, give differential diagnoses. Such information is very useful for phenotype detection. The statement may include a hedge or other assertion types. In Stage 1, labels are marked text spans similar to change-of-state events, but headed by a diagnosis head (`Dhead`) instead of a `Cos` entity. Diagnosis event tuples can be generated in similar way to the change-of-state, but instead of a `Cos` field, a diagnosis tuple has a `Dhead` field.

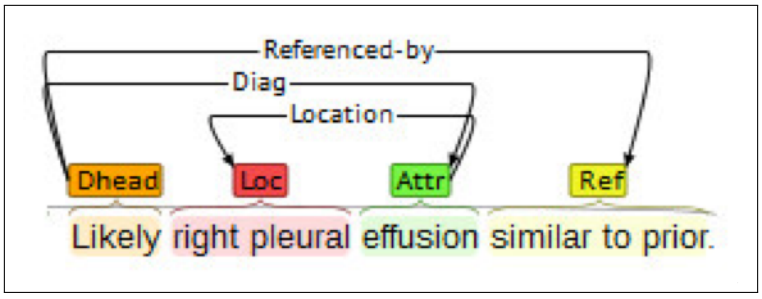

Figure A.3: A diagnosis event

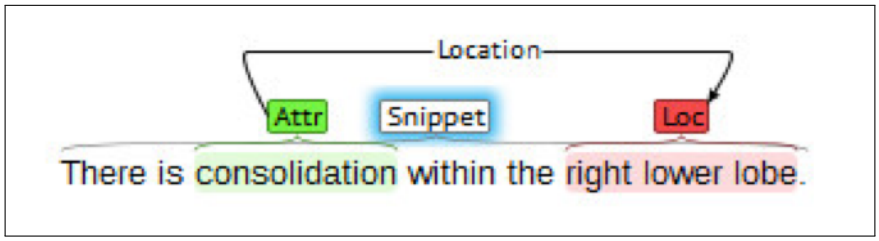

Figure A.4: a clinical attribute with no `Val` entity

In Figure A.3, the rationale snippet contains a diagnosis event only. The diagnosis event tuple automatically generated from the diagnosis event tree is:

```
[Dhead: likely, Attr: effusion, Val: -, Loc: right pleural, Ref: similar to prior]
```

A.3 Examples

Below are some examples of clinical attribute, change-of-state and diagnosis event annotation trees and their corresponding tuples.

A.3.1 Clinical attribute abbreviated forms

In Figure A.4, the clinical attribute is in an abbreviated form, it has no `Val` entity, only connected `Attr` and `Loc` entities. The tuple generated from this clinical attribute is:

```
[Cos: -, Attr: consolidation, Val: -, Loc: right lower lobe, Ref: -]
```

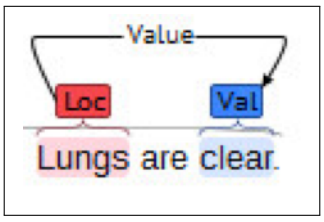

Figure A.5: A clinical attribute with no root level `Attr` entity

In Figure A.5, the clinical attribute is in abbreviated form, it has no root `Attr` value, only connected `Loc` and `Val` entities. The tuple generated from this clinical attribute is:

```
[Cos: -, Attr: -, Val: clear, Loc: Lungs, Ref: -]
```

A.3.2 Change of state abbreviated forms

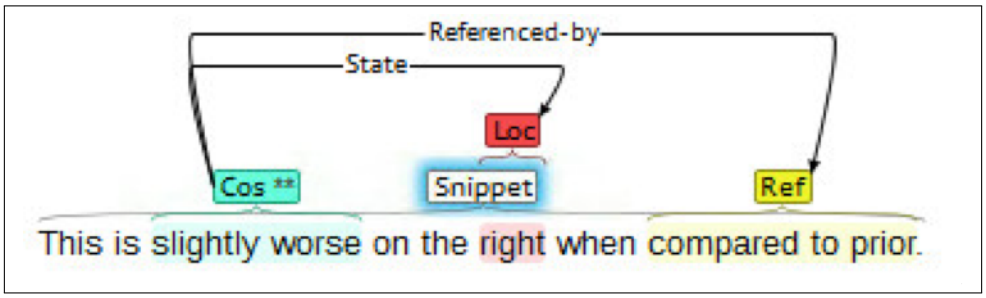

Figure A.6: A change-of-state in abbreviated form

In Figure A.6, the change state event is in abbreviated form. In only includes a `Loc` and `Ref` entity, No `Attr` or `Val` entities. The tuple generated from this change-of-state event is:

```
[Cos: slightly worse (changed, worsened), attr:-, Val: -, Loc: right,
ref: compared to prior]
```

A.3.3 Entities and relations

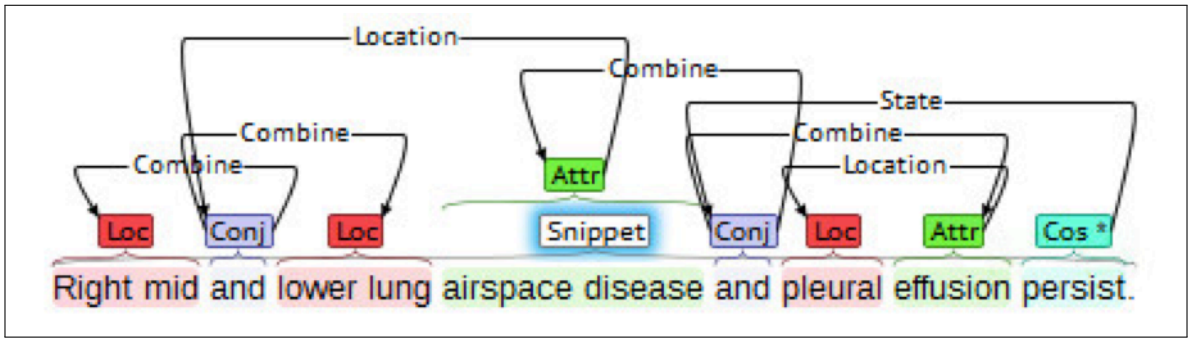

Figure A.7: `Conj Combine` relation

Figure A.7 is an example of the `Conj` entity and the `Combine` relation. In this example, the `Combine` relation between the `Loc` entities results in a logical _and_ connector being inserted into the `Loc` field. The higher level `Combine` relation between the `Attr` entities, results in two discrete tuples for how the `Cos` relates to each `Attr`. The tuples generated from this snippet are:

```
[Cos: persist, Attr: -, Val: airspace disease,
loc: right mid \_and\_ lower lung, Ref: -]
[Cos: persist, Attr: -, Val: effusion, Loc: pleural, Ref: -]
```

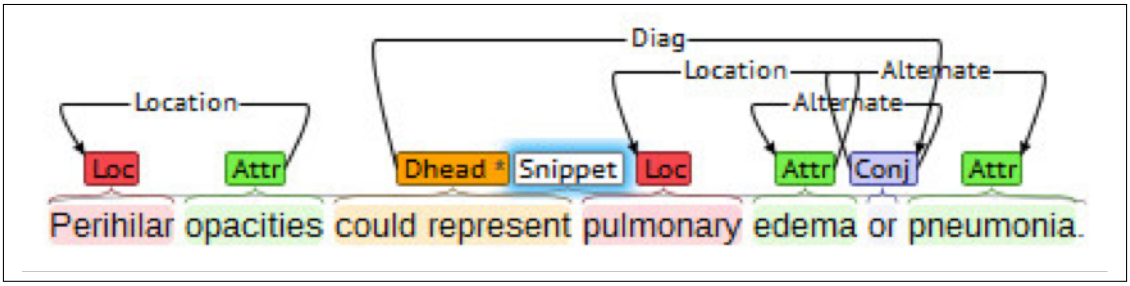

Figure A.8: `Conj Alternate` relation

Figure A.8 is an example of the `Conj` entity and the `Alternate` relation. Because the `Alternate` relation is between two `Attr` entities, the result is two discrete tuples for each relation between the `Dhead` and the `Attr` entities. The tuples generated from this snippet are:

```
[Cos: -, Attr: opacities, Val: -, Loc: perihilar, Ref: -]
[Dhead: could represent, Attr: edema, Val: -, Loc: pulmonary, Ref: -]
[Dhead: could represent, Attr: pneumonia, Val: -, loc:, Ref: -]
```

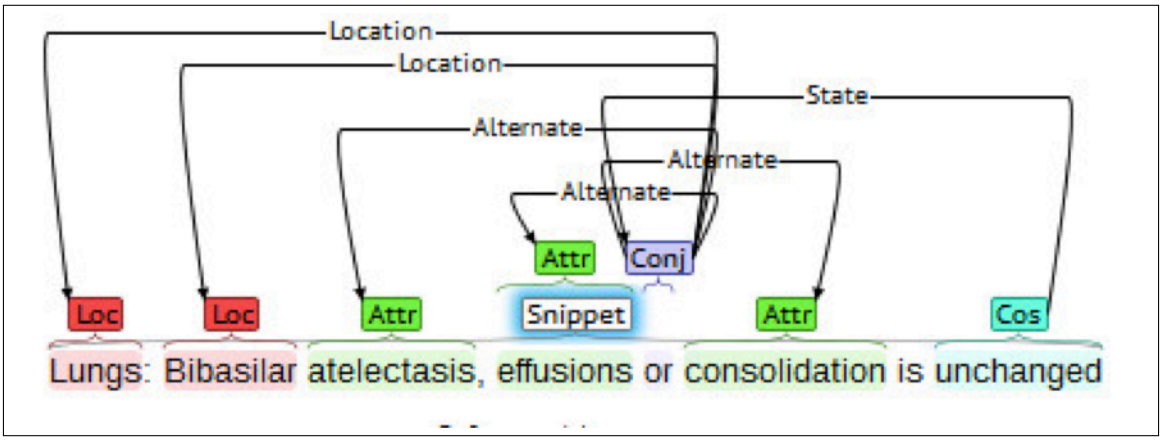

Figure A.9: `Conj Alternate` relation with more than two entities

Figure A.9 is an example of an `Alternate` relation with more than two participating entities. Six tuples are generated for this snippet due to the `Conj` entity aggregating three `Attr` entities and relating them to two locations. The tuples generated from this snippet are:

```
[Cos: unchanged, Attr: atelectasis, val:-, Loc: Lungs, Ref: -]
[Cos: unchanged, Attr: effusions, val:-, Loc: Lungs, Ref: -]
[Cos: unchanged, Attr: consolidation, val:-, Loc: Lungs, Ref: -]
[Cos: unchanged, Attr: atelectasis, val:-, Loc: Bibasilar, Ref: -]
```

[Cos: unchanged, Attr: effusions, val:-, Loc: Bibasilar ref: -]

[Cos: unchanged, Attr: consolidation, val:-, Loc: Bibasilar ref: -]

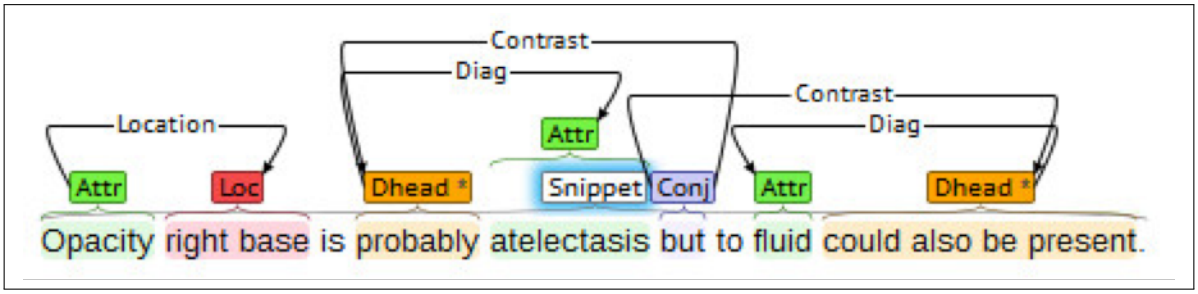

Figure A.10: `Conj Contrast` relation

Figure A.10 is an example of a `Contrast` relation between two diagnosis events. Although the `Contrast` relation adds information to the annotation tree, it results in discrete tuples and the contrastive aspect is lost in tuple generation. The tuples generated form this snippet are:

[Cos: -, Attr: opacity, Val: -, Loc: right base, Ref: -]

[Dhead: probably, Attr: atelectasis, Val: -, Loc: -, Ref: -]

[Dhead: could also be present, Attr: fluid, Val: -, Loc: -, Ref: -]

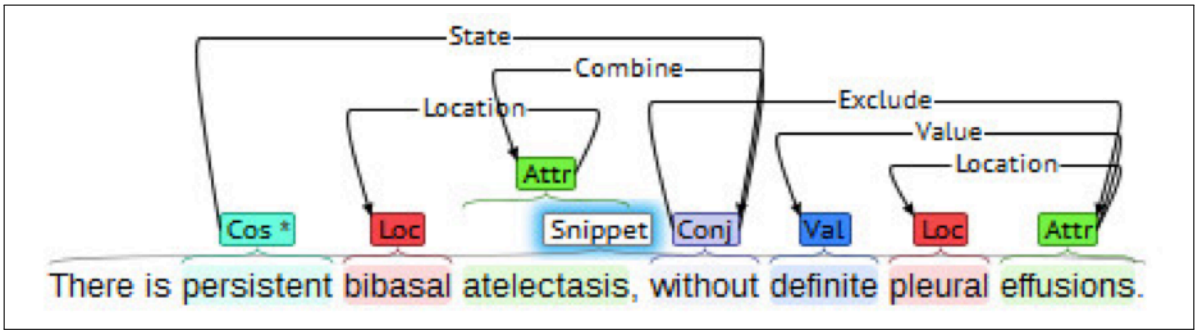

Figure A.11: `Conj Exclude` relation

Figure A.11 is an example of an `Exclude` relation. Although the `Exclude` relation adds information to the annotation tree, it only results in a discrete tuple in the generated tuple. The exception information is not part of the tuple. The tuples generated from this snippet are:

```
[Cos: persistent (persistent), Attr: atelectasis, Val: -, Loc: bibasil, Ref: -]
[Cos: persistent (persistent), Attr: effusions, Val: definite, Loc: pleural, Ref: -]
```

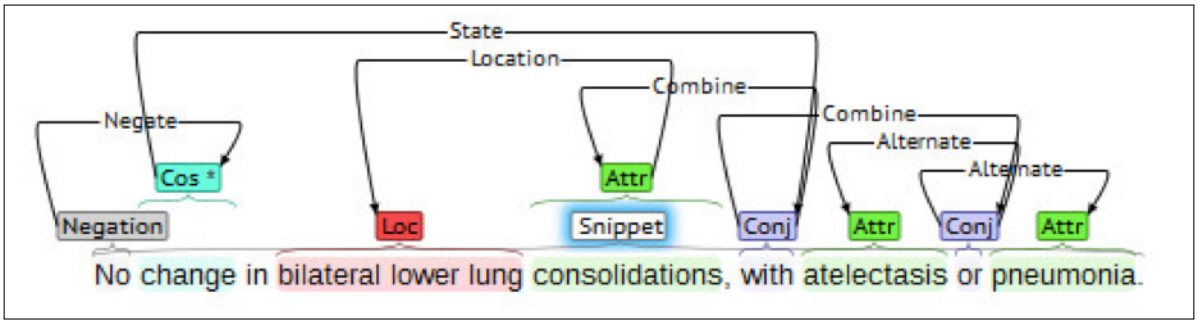

Figure A.12: `Complex Conj` relations

Figure A.12 is an example of a combination of `Conj` relations in a single snippet. In the translation to a tuple representation, the `Negate` relation adds a x_ prefix to the `Cos entity` in each tuple and the combination of `Conj` entities results in multiple discrete tuples. The Alternation aspect is not differentiated from the Combination aspect of coordination in tuple generated. Tuples generated by this snippet are:

```
[Cos: x_change, Attr: consolidations, Val: -, Loc: bilateral lower lung, Ref: -]
[Cos: x_change, Attr: atelectasis, Val: -, Loc: bilateral lower lung, Ref: -]
[Cos: x_change), Attr: pneumonia, Val: -, Loc: bilateral lower lung, Ref: -]
```

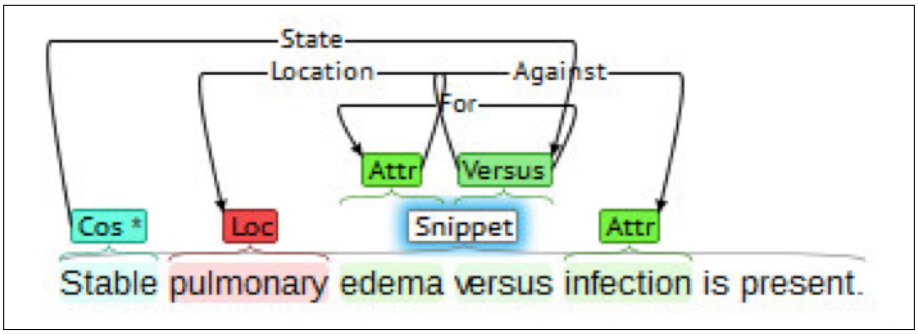

Figure A.13: `Versus` entity and `For` and `Against` relation

Figure A.13 is an example of a `Versus` relation. Although the versus aspect of the relation is not captured between the entities, two discrete tuples are generated:

```
[Cos: stable (stable), Attr: edema, Val: -, Loc: pulmonary, Ref: -]
[Cos: stable (stable), Attr: infection, Val: -, Loc: pulmonary, Ref: -]
```

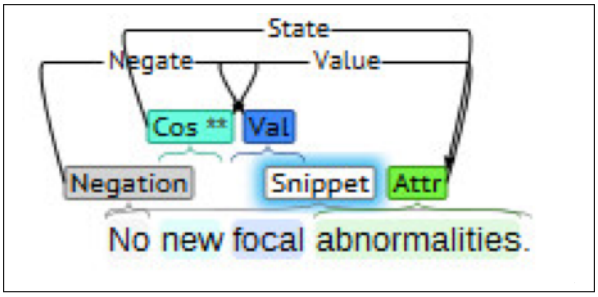

Figure A.14: Negation

Figure A.14 is an example of `Negation`. An x_ prefix is added to the `Cos` entity in tuple generation. The tuple generated from this snippet is:

```
[Cos: x_new (changed, new), Attr: abnormalities, Val: focal, Loc: -, Ref: -]
```

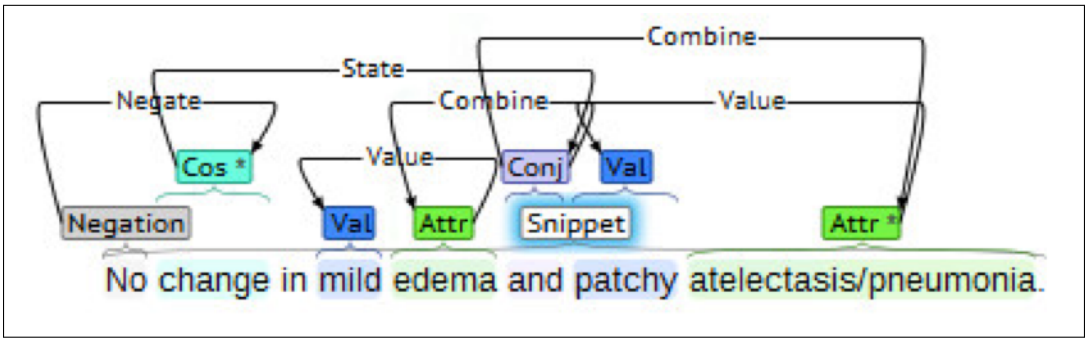

Figure A.15: A slash delimited entity

Figure A.15 is an example of a slash delimited entity. When tuples are generated they are generated for each value in the slash delimited entity. Tuples generated from this snippet are:

```
[Cos: x_change (changed), Attr: edema, Val: mild, Loc: -, Ref: -]
[Cos: x_change (changed), Attr: atelectasis, Val: patchy, Loc: -, Ref: -]
[Cos: x_change (changed), Attr: pneumonia, Val: patchy, Loc: -, Ref: -]
```

A.3.4 Hedging

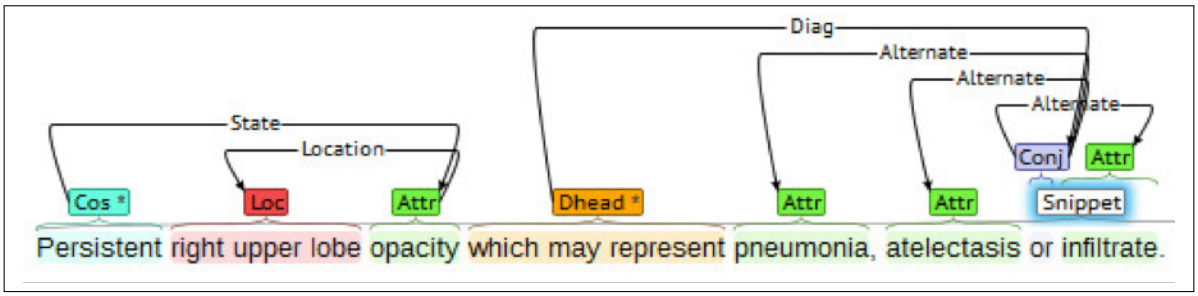

Figure A.16: A `Hedge`

Figure A.16 is an example of a `Hedge`. Each `Alternate` relation results in a discrete diagnosis tuple. Tuples generated from this snippet include:

```
[Cos: persistent, Attr: opacity, Val: -, Loc: right upper lobe, Ref: -]
[Dhead: which may represent, Attr: pneumonia, Val: -, Loc: -, Ref: -]
[Dhead: which may represent, Attr: atelectasis, Val: -, Loc: -, Ref: -]
[Dhead: which may represent, Attr: infiltrate, Val: -, Loc: -, Ref: -]
```

Appendix B

BRAT SCHEMAS

In this appendix, I include three listings of versions of the BRAT schema used to annotate the change-of-state and diagnosis events in the pneumonia report classification rationale snippet corpus. The listings are referred to in Section 3.4.1 and Section 4.4.3.

```
# August, 2014
# Original schema for PNA report classification unique rationale snippet
   corpus

[entities]

Loc
Attr
Val
Cos
Ref
Diagnosis

[relations]

Value   Arg1:Attr|Loc, Arg2:Val
Location    Arg1:Attr|Loc, Arg2:Loc
Referenced-by Arg1:Attr|Cos, Arg2:Ref
State   Arg1:Cos, Arg2:Attr|Loc|Val
```

Listing B.1: Original schema for PNA report classification unique rationale snippet corpus

```
# January, 2015
# First revision of schema for PNA report classification unique rationale
   snippet corpus

[entities]

Snippet
Loc
Attr
Val
Cos
Ref
Diagnosis
Duplicate
Dhead
Conj

[relations]

Value   Arg1:Attr|Loc, Arg2:Val|Conj
Location    Arg1:Attr|Loc|Val, Arg2:Loc|Conj
Referenced-by Arg1:Attr|Cos|Dhead, Arg2:Ref
State   Arg1:Cos, Arg2:Attr|Loc|Val|Conj
Diag    Arg1:Dhead, Arg2:Attr|Loc|Conj
Combine   Arg1:Conj, Arg2:Attr|Loc|Val|Dhead|Cos|Conj|Ref
Extend    Arg1:Val, Arg2:Attr

<OVERLAP> Arg1:<ANY>, Arg2:Snippet, <OVL-TYPE>:contain
<OVERLAP> Arg1:<ANY>, Arg2:Diagnosis, <OVL-TYPE>:contain
```

Listing B.2: First revision of schema for PNA report classification unique rationale snippet corpus

```
# March 2016
# Final version of schema for PNA report classification unique rationale
   snippet corpus

[entities]

# Sentences
Snippet
Duplicate

# clinical attributes <ATT>

Attr
Loc
Val

# clinical event heads <HEAD>

Cos
Dhead

# reference

Ref

# connectors <CON>

Conj
Versus

# negation

Negation
```

```
[relations]

<HEAD>=Cos|Dhead
<ATT>=Attr|Loc|Val
<CON>=Conj|Versus
<ALL>=<HEAD>|<ATT>|<CON>

Value   Arg1:Attr|Loc|<CON>, Arg2:Val|<CON>|Negation
Location    Arg1:<ATT>|<CON>, Arg2:Loc|<CON>|Negation
Referenced-by Arg1:<ALL>, Arg2:Ref
State   Arg1:Cos|<CON>, Arg2:<ATT>|<CON>|Negation
Diag    Arg1:Dhead, Arg2:<ATT>|<CON>|Negation
Combine   Arg1:Conj, Arg2:<ANY>
Contrast    Arg1:Conj, Arg2:<ANY>
Alternate Arg1:Conj, Arg2:<ANY>
Negate    Arg1:Negation, Arg2:<ANY>
Negated   Arg1:<ANY>, Arg2:Negation
For   Arg1:Versus, Arg2:<ANY>
Against   Arg1:Versus, Arg2:<ANY>
Exclude   Arg1:Conj, Arg2:<ANY>

<OVERLAP> Arg1:<ANY>, Arg2:Snippet, <OVL-TYPE>:contain
<OVERLAP> Arg1:<ANY>, Arg2:Diagnosis, <OVL-TYPE>:contain

[attributes]

Increased Arg:Cos
Decreased Arg:Cos
Improved  Arg:Cos
Worsened  Arg:Cos

Stable    Arg:Cos
```

```
New    Arg:Cos
Persistent   Arg:Cos

Changed    Arg:Cos
Slash\_Delimited   Arg:<ANY>
Hedge    Arg:Dhead
```

Listing B.3: Final version of schema for PNA report classification unique rationale snippet corpus

Appendix C

SYSTEM SETTINGS

In this appendix, I provide the default settings for the Stanford CRF-NE Recognizer as discussed in Section 4.4.4 and the configuration settings for the Malt evaluation tool as discussed in Section 4.4.5.

```
map word=0,answer=1,chunk=2,lemma=3,tag=4,docID=5,BEGIN\_POS=6,END\_POS=7,
   goldAnswer=8
useTypeSeqs2        true
maxLeft             1
usePrevSequences    true
useWord             true
wordShape           chris2useLC
usePrev             true
useDisjunctive      true
useTypeSeqs         true
useTypeySequences   true
noMidNGrams         true
useNext             true
useSequences        true
maxNGramLeng        6
useNGrams           true
useClassFeature     true
```

Listing C.1: Stanford CRF NER Settings

The recommended default settings for the Stanford CRF-NE Recognizer are listed in Listing C.1. See Section 4.4.4 for a description of the NER module developed for the VAP report classification study.

```
<evaluation>
    <parameter name="Metric">
        <value>LAS</value>
        <value>UAS</value>
        <value>LA</value>
    </parameter>
    <parameter name="GroupBy">
        <value>Token</value>
        <value format="all">Deprel</value>
    </parameter>
    <parameter name="ExcludeDeprels">
        <value>DEP|ROOT</value>
    </parameter>
    <formatting argument="pattern" format="0.00%"/>
    <formatting argument="micro-average" format="1" />
</evaluation>
```

Listing C.2: A MaltEval configuration file including DEPREL exclusions and micro-average setting

The configuration file for for MaltEval evaluations of VAP change-of-state and diagnosis events as dependency trees is listed in Listing C.2. See Section 4.4.5 for how the MaltEval evaluation tool is used to evaluate VAP change-of-state and diagnosis events in the VAP report classification study.

Appendix D

CONLL 2007 FORMAT DESCRIPTION

In this Appendix, I provide a brief description of the CoNLL 2007 Shared Task on Dependency Parsing format for representing dependency trees as discussed in Section 4.4.5.

The CoNLL 2007 format requires that:

1. Data files contain sentences separated by a blank line.
2. A sentence consists of one or tokens, each one starting on a new line.
3. A token consists of ten fields described in the table below. Fields are separated by a single tab character. Space/blank characters are not allowed in within fields
4. All data files will contains these ten fields, although only the ID, FORM, CPOSTAG, POSTAG, HEAD and DEPREL columns are guaranteed to contain non-dummy (i.e. non-underscore) values for all languages.
5. Data files are UTF-8 encoded (Unicode).

See Table D.1[1]

[1]http://nextens.uvt.nl/depparse-wiki/DataFormat

#	Name	Description
1	ID	Token counter, starting at 1 for each new sentence.
2	FORM	Word form or punctuation symbol.
3	LEMMA	Lemma or stem (depending on particular data set) of word form, or an underscore if not available.
4	CPOSTAG	Coarse-grained part-of-speech tag, where tagset depends on the language.
5	POSTAG	Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available.
6	FEATS	Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (—), or an underscore if not available.
7	HEAD	Head of the current token, which is either a value of ID or zero (0). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero.
8	DEPREL	Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningful or simply 'ROOT'.
9	PHEAD	Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. Note that depending on the original treebank annotation, there may be multiple tokens an with ID of zero. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all languages), whereas the structures resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).
10	PDEPREL	Dependency relation to the PHEAD, or an underscore if not available. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningful or simply 'ROOT'.

Table D.1: A description of CoNLL format

Appendix E

FEATURE THRESHOLD EXPERIMENTS

In this appendix, I list the experimental results of feature selection threshold experiments for pneumonia report classification on predicted and oracle snippets that were not included in the reporting of results in Section 4.4.6 and the discussion of results in Section 4.5.4.1.

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1217	124	124	85.6	81.0	83.0	90.8
200	1341	1341	1224	117	117	87.3	81.3	83.9	91.3
150	1341	1341	1224	117	117	87.3	81.3	83.9	91.3
100	1341	1341	1225	116	116	87.0	82.1	84.2	91.3
50	1341	1341	1224	117	117	87.0	82.2	**84.3**	91.3

Table E.1: CPIS feature threshold experiments, with baseline features, on oracle snippets (Average number of features across folds in model with no feature selection = 485/Number of significant χ^2 ranked features = 165)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1205	136	136	85.0	78.4	81.0	89.9
1500	1341	1341	1193	148	148	86.1	73.2	77.2	89.0
1000	1341	1341	1195	146	146	86.4	73.6	77.7	89.1
750	1341	1341	1195	146	146	86.0	74.0	78.0	89.1
500	1341	1341	1195	146	146	85.1	75.1	78.8	89.1
250	1341	1341	1203	138	138	85.4	77.8	81.0	89.7
200	1341	1341	1204	137	137	85.7	77.8	**81.1**	89.8
150	1341	1341	1198	143	143	85.0	76.3	79.8	89.3
100	1341	1341	1177	164	164	82.3	71.0	74.6	87.8
50	1341	1341	1170	171	171	81.0	68.9	72.2	87.2

Table E.2: CPIS feature threshold experiments, with event-only features, on oracle snippets (Average number of features across folds in model with no feature selection = 4330/Number of significant χ^2 ranked features = 1075)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1225	116	116	87.1	81.8	84.0	91.3
1500	1341	1341	1223	118	118	87.9	79.7	82.7	91.2
1000	1341	1341	1223	118	118	87.9	79.7	82.7	91.2
750	1341	1341	1223	118	118	87.6	80.1	83.0	91.2
500	1341	1341	1228	113	113	87.4	82.2	**84.4**	91.6
250	1341	1341	1225	116	116	87.0	82.0	84.2	91.3
200	1341	1341	1219	122	122	86.5	81.0	83.4	90.9
150	1341	1341	1208	133	133	85.2	79.0	81.7	90.1
100	1341	1341	1194	147	147	84.5	75.4	79.0	89.0
50	1341	1341	1197	144	144	84.9	75.8	79.3	89.3

Table E.3: CPIS feature threshold experiments, with all features, on oracle snippets (Average number of features across folds in model with no feature selection = 4800/Number of significant χ^2 ranked features = 1210)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1155	186	186	76.9	72.4	74.2	86.1
200	1341	1341	1154	187	187	77.6	71.1	73.4	86.1
150	1341	1341	1154	187	187	77.6	71.1	73.4	86.1
100	1341	1341	1154	187	187	77.6	71.1	73.4	86.1
50	1341	1341	1162	179	179	78.9	72.0	**74.4**	86.7

Table E.4: CPIS feature threshold experiments, with baseline features, on predicted snippets (Average number of features across folds in model with no feature selection = 475/Number of significant χ^2 ranked features = 160)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1152	189	189	77.6	70.4	72.7	85.9
1500	1341	1341	1157	184	184	83.3	66.3	70.1	86.3
1000	1341	1341	1159	182	182	83.3	67.2	71.3	86.4
750	1341	1341	1158	183	183	81.4	67.9	71.8	86.4
500	1341	1341	1159	182	182	80.0	69.9	**73.5**	86.4
250	1341	1341	1152	189	189	80.0	67.7	71.9	85.9
200	1341	1341	1152	189	189	79.9	67.9	71.9	85.9
150	1341	1341	1154	187	187	80.1	68.2	72.2	86.1
100	1341	1341	1154	187	187	79.8	68.5	72.2	86.1
50	1341	1341	1149	192	192	78.8	67.2	70.8	85.7

Table E.5: CPIS feature threshold experiments, with event-only features, on predicted snippets (Average number of features across folds in model with no feature selection = 4150/Number of significant χ^2 ranked features = 1190)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1341	1341	1159	182	182	78.2	71.8	74.1	86.4
1500	1341	1341	1163	178	178	83.0	68.4	71.9	86.7
1000	1341	1341	1163	178	178	81.9	69.3	72.8	86.7
750	1341	1341	1166	175	175	80.8	71.1	74.3	87.0
500	1341	1341	1174	167	167	81.3	73.5	76.4	87.5
250	1341	1341	1172	169	169	80.9	73.3	76.1	87.4
200	1341	1341	1174	167	167	80.9	73.7	76.3	87.5
150	1341	1341	1175	166	166	80.6	74.3	**76.6**	87.6
100	1341	1341	1170	171	171	79.8	73.1	75.2	87.2
50	1341	1341	1145	196	196	77.8	66.8	70.3	85.4

Table E.6: CPIS feature threshold experiments, with all features, on predicted snippets (Average number of features across folds in model with no feature selection = 4675/Number of significant χ^2 ranked features = 1350)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1139	204	204	78.6	77.1	77.6	84.8
100	1343	1343	1147	196	196	79.3	77.8	**78.3**	85.4
50	1343	1343	1120	223	223	76.2	73.8	74.7	83.4

Table E.7: PNA feature threshold experiments, with baseline features, on oracle snippets (Average number of features across folds in model with no feature selection = 480/Number of significant χ^2 ranked features = 90)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1096	247	247	74.5	72.2	73.2	81.6
500	1343	1343	1128	215	215	78.0	75.9	**76.8**	84.0
250	1343	1343	1124	219	219	77.3	75.5	76.3	83.7
200	1343	1343	1123	220	220	77.3	75.8	76.4	83.6
150	1343	1343	1108	235	235	76.0	73.8	74.8	82.5
100	1343	1343	1095	248	248	74.8	71.9	73.1	81.5
50	1343	1343	1082	261	261	73.2	68.5	70.0	80.6

Table E.8: PNA feature threshold experiments, with event-only features, on oracle snippets (Average number of features across folds in model with no feature selection = 3565/Number of significant χ^2 ranked features = 265)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1114	229	229	75.9	74.8	75.3	82.9
500	1343	1343	1139	204	204	78.6	77.3	**77.9**	84.8
250	1343	1343	1123	220	220	76.8	75.5	75.9	83.6
200	1343	1343	1117	226	226	75.7	74.8	75.1	83.2
150	1343	1343	1112	231	231	75.7	74.0	74.8	82.8
100	1343	1343	1125	218	218	77.0	74.9	75.8	83.8
50	1343	1343	1087	256	256	72.3	67.8	69.3	80.9

Table E.9: PNA feature threshold experiments, with all features, on oracle snippets (Average number of features across folds in model with no feature selection = 4010/Number of significant χ^2 ranked features = 380)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1093	250	250	74.4	72.2	72.4	81.4
150	1343	1343	1119	224	224	77.5	74.4	**75.1**	83.3
100	1343	1343	1116	227	227	77.1	74.3	74.9	83.1
50	1343	1343	1113	230	230	77.0	73.4	73.7	82.9

Table E.10: PNA feature threshold experiments, with baseline features, on predicted snippets (Average number of features across folds in model with no feature selection = 490/Number of significant χ^2 ranked features = 125)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1075	268	268	74.1	70.0	71.5	80.0
500	1343	1343	1108	235	235	76.7	72.5	**73.9**	82.5
250	1343	1343	1093	250	250	75.2	71.0	72.2	81.4
200	1343	1343	1098	245	245	76.0	71.4	72.8	81.8
150	1343	1343	1102	241	241	76.5	71.9	73.1	82.1
100	1343	1343	1090	253	253	75.6	70.0	71.6	81.2
50	1343	1343	1097	246	246	77.6	70.3	71.9	81.7

Table E.11: PNA feature threshold experiments, with event-only features, on predicted snippets (Average number of features across folds in model with no feature selection = 4200/Number of significant χ^2 ranked features = 460)

θ	**S**	**G**	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**Acc**
-	1343	1343	1077	266	266	73.6	70.9	71.8	80.2
750	1343	1343	1103	240	240	75.4	73.3	73.8	82.1
500	1343	1343	1105	238	238	75.8	73.5	74.2	82.3
250	1343	1343	1099	244	244	75.6	73.2	73.8	81.8
200	1343	1343	1111	232	232	76.8	74.1	**74.8**	82.7
150	1343	1343	1103	240	240	76.1	72.7	73.7	82.1
100	1343	1343	1104	239	239	76.3	72.3	73.5	82.2
50	1343	1343	1105	238	238	75.8	71.7	72.8	82.3

Table E.12: PNA feature threshold experiments, with all features, on predicted snippets (Average number of features across folds in model with no feature selection = 4690/Number of significant χ^2 ranked features = 585)