View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by DigitalCommons@CalPoly

A HYBRID APPROACH TO GENERAL INFORMATION EXTRACTION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Marie Grap

September 2015

© 2015
Marie Grap
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE:	A Hybrid Approach to General Information Extraction
AUTHOR:	Marie Grap
DATE SUBMITTED:	September 2015
COMMITTEE CHAIR:	Foaad Khosmood, Ph.D. Assistant Professor of Computer Science Department of Computer Science
COMMITTEE MEMBER:	Alexander Dekhtyar, Ph.D. Professor of Computer Science Department of Computer Science
COMMITTEE MEMBER:	Franz Kurfess, Ph.D. Professor of Computer Science Department of Computer Science

ABSTRACT

A Hybrid Approach to General Information Extraction

Marie Grap

Information Extraction (IE) is the process of analyzing documents and identifying desired pieces of information within them. Many IE systems have been developed over the last couple of decades, but there is still room for improvement as IE remains an open problem for researchers. This work discusses the development of a hybrid IE system that attempts to combine the strengths of rule-based and statistical IE systems while avoiding their unique pitfalls in order to achieve high performance for any type of information on any type of document. Test results show that this system operates competitively in cases where target information belongs to a highly-structured data type and when critical contextual information is in close proximity to the target.

ACKNOWLEDGMENTS

Thanks to:

- Andrew Guenther, for uploading this template

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Today, the amount and variety of information available is not only immense, but also ever-expanding. From news articles to order confirmations to social media posts, it has never been so easy to create data and send it out into the world. However, despite this great abundance of information, much of it remains virtually useless to machines because it is written in human language. Information Extraction (IE) addresses this problem by analyzing documents and identifying desired pieces of information within them. Much more sophisticated than a simple keyword search, IE aims to identify rich information such as the proteins mentioned in a biomedical journal or the relevant pieces of data needed to cite an article.

Many methods for IE have been developed over the last couple of decades, but there is still room for improvement as IE remains an open problem for researchers. Over the years, it has been observed that most state-of-the-art methods fall under two camps: rule-based and statistical. This work explores the possibility of combining techniques from these two methods to create a hybrid system. The goal of the hybrid system is to achieve high performance by utilizing the strengths of its parents while at the same time avoiding their pitfalls. Specifically, this system aims to extract information of any type from documents of any text type.

The basic work flow of this system begins with a corpus of labeled documents, which are split into training and testing sets. The training set is used to generate rules for targets using both the actual target text and a data type prediction. Once

potential targets have been extracted using these generated rules, contextual features are collected and passed to a classifier. Finally, the classifier uses this contextual information to predict whether or not the potential targets is a desired result.

Testing has shown that this system performs best when targets belong to highly structured data types, such as times or money values, and when important contextual information is in close proximity to the target itself. In a few cases, the system outperforms other state-of-the-art systems, but in most cases its performance is significantly lower.

Here is a brief outline of this paper. Chapter 2 provides background information on IE, machine learning, classifier algorithms, and performance evaluation standards. Chapter 3 discusses related work. Chapter 4 explains the methodology used to create this system in detail. Chapter 5 outlines the experimental procedures used to test the system. Chapter 6 reviews the testing results. Chapter 7 proposes future work for the project. Chapter 8 offers concluding remarks about the project.

Chapter 2

Background

2.1 A Brief History of IE

IE first became recognized as a valuable tool in the 1980's. Between 1987 and 1998 the Defense Advanced Research Projects Agency (DARPA) hosted and financed a series of Message understanding conferences (MUC [13]) with the intention of stimulating research in the area of IE [28]. At these conferences, research groups were given a corpus of documents such as military reports or news reports, and competed to create a system that could identify information about provided topics such as fleet operations or terrorist activities in Latin America [13]. Similar programs have since been launched, including ACE (Automatic Content Extraction [7]; MUC's successor) and CoNNL (Conference on Computational Natural Language Learning [37]), which have further encouraged progress in the field of IE.

The earliest IE systems utilized manually-crafted rules, which identified pieces of information by looking for specified patterns in documents [31, 28]. Although these systems served their purpose, they faced the major disadvantages of being extremely domain-specific and needing programmers with extensive domain knowledge, linguistic understanding, and programming skills. These issues were somewhat alleviated with the advent of rule-learning systems, which used hand-labeled documents to train machine learning extraction models [31, 28]. Rule-learning systems perform especially

well in documents where data follows some structural rules, such as resumes or receipts.

Statistical approaches to IE were developed in response to the cases where rule-learning systems fall short, that is, when a task involves extracting data from free text [31, 28]. These systems also require labeled training documents, but differ in that they focus on linguistic data rather than formatted patterns to discover information. Although statistical systems are typically slower, more difficult to program, and and harder to interpret than their rule-learning counterparts, they are more robust to noise in unstructured documents.

In spite of all the research that has been done, there is still no clear answer as to whether rule-learning or statistical IE systems are better, since each one performs better in different situations. The purpose of this project is to present a single, hybrid system that works for all types of documents (structured, semi-structured, and unstructured) by reconciling the differences between rule-learning and statistical approaches.

2.2 Machine Learning

Machine Learning (ML) can be defined as "a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty" [20]. In this case, the data of interest is natural language text. This project uses supervised learning, meaning that it uses a set of labeled training documents in order to learn how to make predictions.

The process for supervised learning begins with a set of user-supplied documents and the expected output of the system in relation to those documents. For example, in our system, the user supplies a set of documents containing labeled text that represents the expected output of the program. The next step of the process is feature selection. A feature can be any piece of information that can be determined from the text of a document and any outside resource, barring the expected output labels of course. Examples of a feature include: number of words, most popular

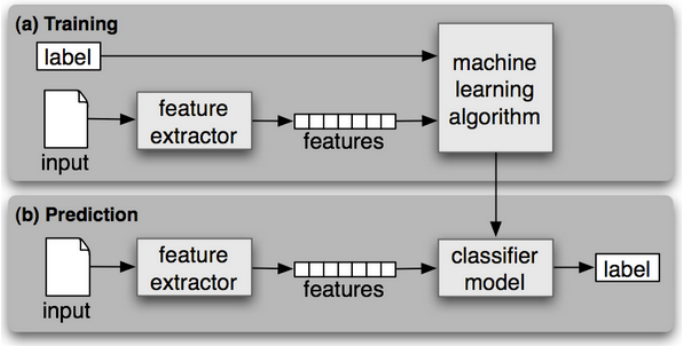

Figure 2.1: Supervised Learning Process [3].

words, most popular groups of three words (commonly referred to as trigrams), least popular parts of speech, or the presence of a specific word. With so much possible data to choose from, selecting which features are useful is a difficult and crucial task, and is currently an open problem in ML. Once the features have been selected, they are extracted from each document.

After the feature sets have been extracted, they are split into two groups: a training set and a testing set. Typically, the training set represents a half or more of the original document set, and is used to train a classifier. In simple terms, a classifier is an algorithm that makes a prediction based on a set of features. During the training process, a classifier looks at both the features and the expected results and learns which features are the best predictors for certain results. Our system uses a Naive Bayes classifier, which will be discussed in further detail in the next section. Once the classifier has been trained, it may be used for testing, during which the classifier only looks at the given set of features and tries to make a prediction. Once the prediction has been made, the expected results can be compared to the prediction.

2.3 Classifiers

Several classification algorithms were tested during the development of this system, including Naive Bayes, Maximum Entropy, Decision Trees, and SVM. Naive Bayes was chosen as the system's classifier because it outperformed all the others. As such, only Naive Bayes will be discussed in detail in this section.

2.3.1 Naive Bayes

Naive Bayes classifiers operate under the "naive" assumption that all features are independent of each other (This is commonly referred to as the independence assumption). This is, of course, very unrealistic since it is highly likely that selected features have some sort of dependence on one another. However, despite this fact, Naive Bayes classifiers often outperform classifiers that use more sophisticated algorithms that try to account for feature relationships. Because of their simplicity and high performance, Naive Bayes classifiers have become one of the most popular and commonly used techniques for ML [3].

Naive Bayes classifiers use the independence assumption with Bayes Theorem (Figure 1.1) to calculate the most likely expected result based on a feature set [3].

$$P(result|features) = \frac{P(features|result)P(result)}{P(features)}$$

Figure 2.2: Bayes Theorem.

Consider this example: pretend that we are trying to classify people as male or female (result) based on their first name (feature). First, we collect some information about first names and the sex associated with them (training). Table 2.1 shows the collected information.

Suppose we then wish to classify the name Alex as male or female. With the collected information, we can use Bayes' Theorem to determine if it is more likely for Alex to be male or female.

$$P(Male|Alex) = \frac{(\frac{2}{6})(\frac{6}{10})}{(\frac{3}{10})} = 0.66$$

$$P(Female|Alex) = \frac{(\frac{1}{4})(\frac{4}{10})}{(\frac{3}{10})} = 0.33$$

Figure 2.3: Since the likelihood that Alex is male is greater than the likelihood that Alex is female, we decide to classify Alex as male.

Name	Sex
Lexi	Female
Alex	Male
John	Male
Alberta	Female
Alex	Female
Katie	Female
Sean	Male
Alex	Male
Peter	Male
Joe	Male

Table 2.1: Information collected about the first name Alex and the corresponding sex.

2.4 Performance Measures

Once a classifier has been created, it is necessary to determine how well it performs. There are four common measurements used for this purpose: accuracy, precision, recall, and F-measure [16]. These values are calculated using the numbers of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) generated by a classifier during testing, which are modeled in Table 2.2.

	Correct	**Incorrect**
Selected	TP	FP
Not Selected	FN	TN

Table 2.2: Contingency table.

Accuracy represents the overall correctness of a classifiers predictions and is calculated by dividing the number of correct classifications by the total number of clas-

sifications (Figure 2.4)[16].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 2.4: Accuracy Equation.

Unfortunately, accuracy alone is not a good measure of a classifier's performance. Consider the case where there are very few correct items in a data set. If a classifier chose not to select any items as correct, it could still have an extremely high accuracy since the number of TN would be very high.

To account for this shortfall, precision and recall are also calculated (Figure 2.5). Precision is the percent of selected items that are correct and is important when the user requires reliable results. Recall is the percent of correct items that are selected and is important when the user wishes to capture the largest possible number of correct results[16].

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

Figure 2.5: Equations for Accuracy, Precision, and Recall.

Since there is a major trade-off between precision and recall, developers often make use of the weighted harmonic mean, or F-measure, of these two values (Figure 2.6)[16].

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \qquad\qquad F1 = \frac{2PR}{P + R}$$

Figure 2.6: Equations for F-Measure and Balanced F1-Measure, where P represents precision and R represents recall. β is the desired weight for precision and recall.

If either precision or recall is more important, a weight (β) can be used to affect the F-measure accordingly. In most cases, however, precision and recall are weighted equally (i.e. $\beta = 1$), resulting in what is known as the Balanced F1-Measure (Figure 2.6)[16]. The F1 measure ranges from 0 (worst) to 1 (best).

2.5 Natural Language Toolkit (NLTK)

NLTK is a Python-based platform that provides tools for natural language processing tasks including text classification, tokenization, stemming, tagging, and parsing [3]. The following features of NLTK are used in this project:

- **Part-of-Speech Tagger**: NLTK has a built-in part-of-speech tagger which uses a maximum-entropy model with the Penn Treebank tag-set [3, 17]. This project uses the output of the part-of-speech tagger to help with the generation of rules and features from text.
- **Stemmer**: This project uses NLTK's Snowball stemmer to make words more generic by reducing them to their roots.
- **Tokenizers**: NLTK's Word and sentence tokenizers are used to break up text along natural boundaries.
- **Stop-words**: Stop-word are common words that do not add much meaning to text, such as "the", "and" or "it". NLTK provides a list of stop-words for the purposes of identification and removal.
- **Classifiers**: NLTK not only offers numerous types of classifier implementations, but also provides an API that can be used to access all of them in the same way. This project primarily uses the built-in Naive Bayes classifier, but other classifiers were used to test performance as well.

2.6 Stanford Named Entity Recognizer (NER)

Stanford NER is a Java-based tool that labels instances of named entities within text [8]. It is primarily recognized for its ability to identify references to the names of people, locations, and organizations, but it is also capable of finding other entities such as dates, monetary values, and percentages.

Chapter 3

Related Work

This section discusses the implementations, strengths, and weaknesses of three current state-of-the-art IE systems: BWI, LP^2, and ELIE. All three systems aim to accomplish the same task: to extract any type of information from documents of any text type. The performance of these three systems is compared in Tables 6.5 and 6.2.

3.1 Boosted Wrapper Induction (BWI)

In 2000, a rule-learning technique called Boosted Wrapper Induction (BWI) was used to create an IE system [11]. In this context, a wrapper simply refers to a contextual pattern. An example of a wrapper that identifies URLs in HTML documents might look like this: `[<a=href‘‘] [’’>]`. Wrapper induction is the process of automatically learning wrappers, instead of hand-coding them. Traditionally, simple wrappers have been very effective for retrieving data from highly structured sources, such as csv files or documents generated by databases. However, wrapper induction has allowed for information extraction from less-structured sources. For instance, patterns such as `[Who:]` or `[Dr.]` could be used to identify the beginning of speaker names within a seminar announcement corpus. Still, although these learned wrappers have very high precision, their recall is very low. In order to raise recall, a process called Boosting is used, which combines many high-precision patterns to create a

single high-recall pattern.

Before going into the details of BWI, a few terms need to be defined. A *field* is the desired piece of information, and a *boundary* consists of the prefix and suffix patterns that surround a *field*. A *boundary detector* is a pair of patterns $d = <p, s>$, where p is a prefix pattern and s is a suffix pattern, and $C(d)$, a confidence value. Using the previous URL example, when presented with the text <a=href‘‘www.coolbeans.com’’>, p = <a=href‘‘ and s = ’’>. In addition to using exact tokens, patterns can also include wildcards, which are special tokens that represent a set of tokens. The wildcards used by BWI are listed in Table 3.1. A wrapper is represented by $W = <F, A, H>$, where F is a set of "fore" boundaries, A is a set of "after" boundaries, and $H(k)$ is a function that represents the probability that a field representing a desired piece of information has length k.

Wildcard	**Description**
<Alph>	matches any token that contains only alphabetic characters
<ANum>	contains only alphanumeric characters
<Cap>	begins with an upper-case letter
<LC>	begins with a lower-case letter
<SChar>	any one-character token
<Num>	containing only digits
<Punc>	a punctuation token
<*>	any token

Table 3.1: Wildcards used by BWI [11].

procedure BWI(example sets S and E)
$F \leftarrow$ AdaBoost(LearnDetector, S)
$A \leftarrow$ AdaBoost(LearnDetector, E)
$H \leftarrow$ field length histogram from S and E
return wrapper $W = \langle F, A, H \rangle$

Figure 3.1: BWI algorithm [11].

```
procedure LearnDetector(example set Y)
    prefix pattern p ← [ ]
    suffix pattern s ← [ ]
    loop
        prefix pattern p' ← BestPreExt(⟨p, s⟩, Y)
        suffix pattern s' ← BestSufExt(⟨p, s⟩, Y)
        if score(⟨p', s⟩) > score(⟨p, s'⟩)
            if score(⟨p', s⟩) > score(⟨p, s⟩)
                p ← the last |p| + 1 tokens of p'
            else return detector ⟨p, s⟩
        else
            if score(⟨p, s'⟩) > score(⟨p, s⟩)
                s ← the first |s| + 1 tokens of s'
            else return detector ⟨p, s⟩
```

Figure 3.2: The LearnDetector weak learner [11].

The algorithm used by BWI is shown in Figure 3.1. BWI begins by accepting two training sets, S and E, which are both constructed from the same documents. S and E both contain sets of *boundaries* labeled Begin(b) and End(b) respectively. AdaBoost calls LearnDetector T times to learn the set of "fore" and "after" detectors, and H is determined by counting the number of fields with length k. LearnDetector creates a single *boundary detector* by determining which pattern of length L or less is most successful over the training set. Each time a detector is returned to AdaBoost, a confidence value is assigned to it, and the training set is updated accordingly, with the misclassified items receiving higher priority for the next iteration. Once the algorithm has completed, a wrapper has been produced that can be used for information extraction.

3.2 Learning Pinocchio (LP^2)

Learning Pinocchio (LP^2) was developed shortly after BWI in 2001 [6]. Also a rule-learning system, (LP^2) has two distinct phases:

1. Sets of **tagging rules** are induced that insert a preliminary tagging.

2. **Correction rules** are induced that refine the tagging by correcting mistakes

and imprecision.

In the first phase, initial rules are generated from the training set. Rules consist of two parts: a left-hand side and a right-hand side. The left-hand side contains a pattern of conditions over a window of words. These conditions include the word, lemma, lexical category, case, and semantic class (from a user-defined dictionary or gazetteer when available). The right-hand side is an action, which inserts a single XML tag indicating the beginning or end of a target. Table 3.2 represents an initial rule that is induced from the text, "the seminar at <stime>4 pm will...".

	Condition	**Some Data**				**Action**
word index	word	lemma	LexCat	case	SemCat	Tag
1	the	the	Art	low		
2	seminar	seminar	Noun	low		
3	at	at	Prep	low		**<stime>**
4	4	4	Digit	low		
5	pm	pm	Other	low	timeid	
6	will	will	Verb	low		

Table 3.2: Initial rule for "the seminar at <stime>4 pm will..." [6].

After the initial rules are generated, LP^2 attempts to generalize them by reducing their length and relaxing constraints from some of the conditions. An example of a generalization of the rule in Table 3.2 is shown in Table 3.3. LP^2 then accepts the k best generalizations of the initial rules. A generalization is accepted if it:

1. covers at least a minimum number of cases on the training corpus
2. has an error rate less than a user-defined threshold

Although *best rules* have very high precision, they also have low recall. In order to adjust for this, LP^2 creates *contextual rules*, which it draws from the pool of rules initially rejected in the creation of *best rules*. In order to increase the reliability of

	Condition	**Some Data**				**Action**
word index	word	lemma	LexCat	case	SemCat	Tag
3		at				**<stime>**
4			Digit			
5					timeid	

Table 3.3: A generalization for rule in Table 3.2. The pattern is relaxed in length (conditions on words 1, 2 and 6 were removed) and conditions on the other words were substituted by other constraints [6].

these rejected rules, they are only used in the context of *best rules*. For example, a rule that places </speaker> between and upper-case and lower-case word is not very reliable on its own. However, if a *best rule* has been able to insert an opening <speaker> tag, this rule can be used to close the tag very reliably. The acceptance of *contextual rules* is calculated in the same way as *best rules*, but only matches in constrained contexts are counted.

The second phase of the LP^2 algorithm is the generation of correction rules. The purpose of a correction rule is to improve boundary detection when tags are misplaced by tagging rules. Consider this example: "at <time>4</time> pm". A correction rule would shift the </time> tag over so that "pm" would be included in the target. Correction rules are very similar to tagging rules with two exceptions. First, the conditions on the left-hand side of correction rules include tags that have been inserted by tagging rules. Second, the action performed by a correction rule shifts the position of a tag, rather than creating a new one. Correction rules are accepted using the same algorithm as tagging rules. Using these generated tagging and correction rules, LP^2 is able to perform information extraction.

3.3 ELIE

ELIE was developed in 2004 and uses a statistical approach to IE. This system is a very powerful example of how relatively "standard" machine learning techniques

can be used to create a competitive system [9].

Similar to BWI and LP^2, ELIE separates the identification of start and end tokens into two distinct tasks. During training, documents are word-tokenized, and all tokens are used to train two SVM classifiers: a start classifier and an end classifier. All tokens that have been labeled as the start of a field are used as positive examples for the start classifier, while all other tokens are used as negative examples. The end classifier is trained in the same way using field endings. The features associated with tokens and their surrounding text include part-of-speech, part-of-speech chunking, orthographic information (i.e. case, punctuation, alphanumeric characters), and gazetteer lookups. Once the start and end classifiers have been trained, they can be used to create a learning model (L1), which contains a set of start tokens and a set of end tokens. This learning model has very high precision, since it is based on a very large number of negative examples, and a small number of positive ones.

The start and end tokens from L1 are passed to a Tag Matcher, which attempts to determine which start tokens belong with which end tokens based on position and average field length. The pairs created by the Tag Matcher are accepted as positive instances, and any tokens that could not be paired are used to create a second learning model (L2). L2 also uses a start and end classifier. To train the L2 start classifier, the unmatched end tokens are used to extract potential start tokens that are within a fixed distance. A similar process is used for the L2 start classifier. Since the training data used by L2 has a much higher ratio of positive to negative examples, L2 is likely to have high recall and low precision. Once L2 has predicted start and end tokens, the Tag Matcher is used once again to pair the unmatched tokens from L1 with the newly predicted tokens from L2. These pairs, along with the first set of pairs are returned as extracted fields. The entire IE process used by ELIE after training is shown in Figure 3.3.

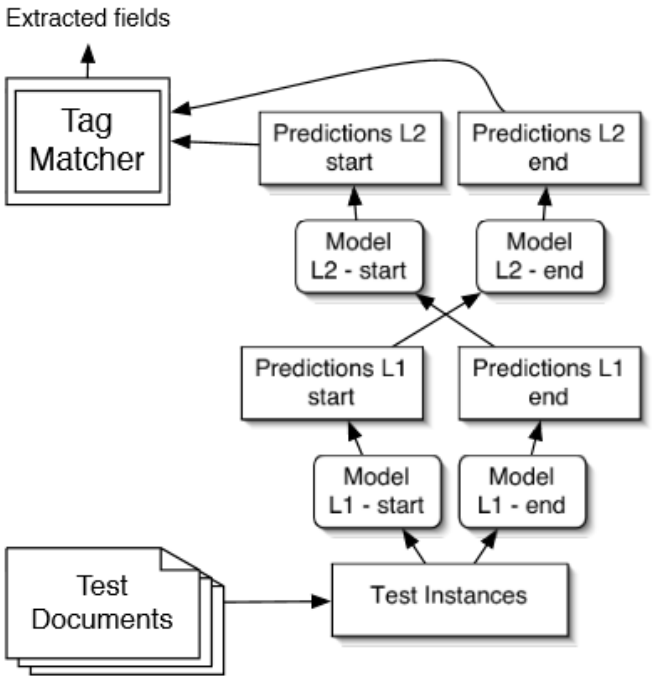

Figure 3.3: The IE process used by ELIE [9].

3.4 Discussion

Although BWI performs competitively with other state-of-the-art methods, it has a few weaknesses that could be improved on. Firstly, it makes no use of linguistic data other than the literal words surrounding a target. It is true that wildcards are used to help generalize text, but they are very limited and a lot of potentially useful linguistic features, such as part of speech or semantic categorization, are lost. In addition, the only feature used for the target itself is the average length. The use of more features such as case, data type, or part of speech for the target field could greatly improve upon the quality of extracted text.

Unlike BWI, LP^2 does very well at making use of all available linguistic information and, as a result, performs slightly better. However, as with BWI and other rule-based systems, the identification of rules is biased towards overfitting to the particularities of the training data [31]. This becomes an important problem when dealing with free text, since there can be many ways to communicate a similar idea.

As a statistical approach, ELIE is able to avoid the over-dependence on patterns demonstrated by BWI and LP^2. One potential weakness however, which was also present in BWI, is the lack of attention paid to target features. Although it is true that ELIE collects features for the first and last words in a target, nothing besides average length is ever noted for the target as a whole.

Chapter 4

Methodology

4.1 Corpora

For this project, corpora were selected to represent each of the structured, semi-structured, and unstructured text types. The intention for this was that good performance across all three corpora would indicate that a system was well-suited for any text type.

Corpus	**# Docs**	**Avg Doc Size**	**Labels**
CMU Seminar Announcements	484	1.0 KB	`speaker`, `location`, `stime` (start time), `etime` (end time)
Seattle Times Rentals	256	0.1 KB	`neighborhood`, `bedrooms`, `price`
GENIA	800	1.4 KB	`protein`, `entity`
U.S. Company 10-K's	49	8.7 KB	`pre` (permanently reinvested earnings)

Table 4.1: The corpora used to develop and test the system.

4.1.1 Carnegie Mellon University Seminar Announcements

The Carnegie Mellon University Seminar Announcements corpus is the most popular corpus for testing IE systems and consists of 485 seminar announcements collected from campus newsgroups in 1998 [31]. These documents are an excellent example of semi-structured text. The labels used by this corpus are presented in Table 4.2. There can be more than one speaker for a seminar, but only one location, start time, and end time. However, when annotating a document, the goal is to find every mention of each. It is important to note that although this corpus is widely used, it is not perfect as it contains "numerous labeling errors and inconsistencies" [9]. As such, a goal of perfect performance on this data set is unrealistic. This corpus can be obtained from the RISE Repository (*http://www.isi.edu/info-agents/RISE/*) [21].

	Description
`speaker`	A person who will speak at a seminar
`location`	The location of a seminar
`stime`	The starting time of a seminar
`etime`	The ending time of a seminar

Table 4.2: The labels used by the CMU Seminar Announcements corpus.

4.1.2 Seattle Times Rentals

This corpus is a collection of 256 apartment rentals listed in the Seattle Times and was created by Stephen Soderland in 1999 in order to test his IE system, WHISK [33]. These documents follow a highly structured format. A list of labels used in this corpus is shown in Table 4.3. Each document represents one advertisement, which can list multiple properties with different neighborhoods, prices, and numbers of bedrooms. During annotation, every instance of a neighborhood, price, and bedroom listing is labeled. This corpus can be obtained from the RISE Repository (*http://www.isi.edu/info-agents/RISE/*) [21].

	Description
`Neighborhood`	The neighborhood(s) of a rental property
`Bedrooms`	The number of bedrooms in a rental property
`Price`	The price of a rental property

Table 4.3: The labels used by the Seattle Times Rentals corpus.

4.1.3 GENIA

The GENIA corpus contains PubMed abstracts on transcription factors in human blood cells and was used in the BioNLP'11 shared task [14]. These documents contain unstructured text, and are particularly interesting because the labeled entities are not associated with traditional data types and are difficult for people with no field expertise to identify. For the sake of time and simplicity, a subset of 800 documents and only two of the eleven possible labels are evaluated in this system. The labels that are used are shown in Table 4.4. Each document can, and usually does, have many `entity` and `protein` labels. This corpus can be obtained from *http://www.nactem.ac.uk/tsujii/GENIA/SharedTask/downloads.shtml*.

	Description
`protein`	A gene or gene product
`entity`	A cell or cell product

Table 4.4: The labels used by the GENIA corpus.

4.1.4 2012 United States Company 10-K's

This corpus consists of 49 truncated 10-K's from U.S. companies in 2012 and was created for this project. The text within these documents is entirely unstructured. There is only one label used in this corpus, `pre`, which represents a company's Permanently Reinvested Earnings (PRE) for a certain year. Typically, a PRE is only mentioned once in a document, although multiple PRE's from multiple years can be listed.

The `pre` is used to find every mention of any PRE within a document. This corpus was created using publicly available documents from the U.S. Securities and Exchange Commission EDGAR Database (*http://www.sec.gov/edgar/searchedgar/companysearch.html*).

	Description
`pre`	A company's Permanently Reinvested Earnings (PRE) for a certain year

Table 4.5: The labels used by 2012 United States Company 10-K corpus.

4.2 Annotation

After the corpora had been selected and obtained, they had to be labeled. The seminar announcements and rental ad had already been labeled, but the formatting of the labels for the two was incredibly dissimilar. In order to standardize label formatting, all corpora were labeled using stand-off format. With the exception of the BioNLP corpus, which had already been labeled in this fashion, all corpora were identically hand-labeled using the Brat Rapid Annotation Tool (BRAT)[35]. BRAT is a flexible, web-based tool that allows users to label entities, relationships between entities, events involving entities, and attributes of entities within document corpora. For the purposes of this project, only entity labeling was used. Figure 4.1 shows a screen-shot of the BRAT user interface.

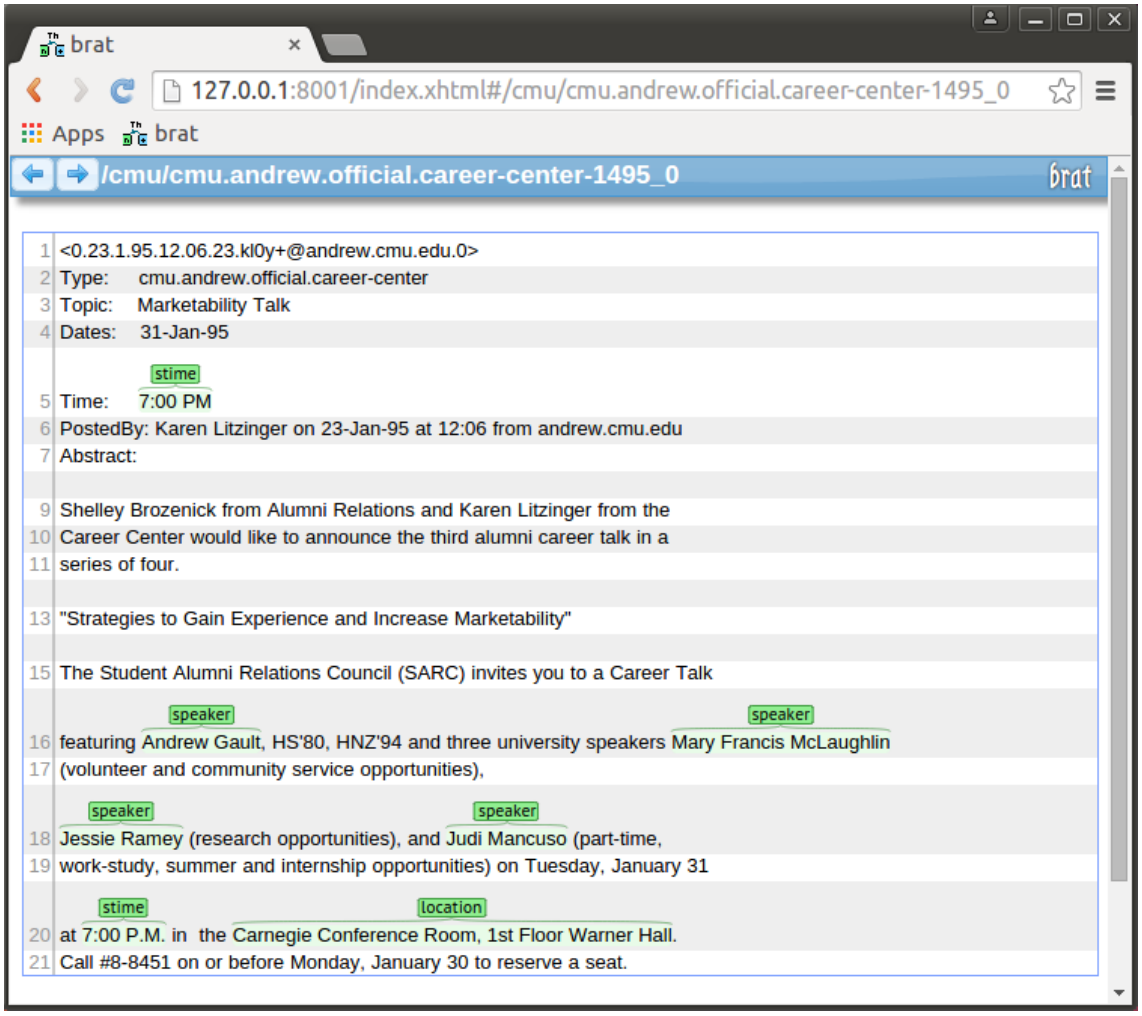

Figure 4.1: Using BRAT to annotate a document from the CMU Seminar Announcements corpus.

4.3 Training

The IE system begins by accepting an labeled corpus, which it splits into parts: one for training and another for testing. This section describes what is done with the training data.

4.3.1 Rules

The first step taken by the system is to generate rules for positive examples of a target in the training data.

Data Type Rules

The system has several built-in *data type* rules that search for specific data types (For a complete list, see Table 4.6). For example, there is a *time* rule that can be used to identify instances of times within a document. The system begins by using these rules to try and identify the data type of the information to be extracted. This is done by checking the number of positive examples within the training data that are covered by the targets extracted by a *data type* rule. If a *data type* rule finds at least 90% of the positive examples in the training data, this is considered sufficient evidence that the information to extract belongs to this data type, and this rule alone is used to generate potential entities. However, if no *data type* rules can produce such excellent coverage, then the rule with the highest coverage percentage above a certain threshold is accepted as a *helper* rule, which are used in conjunction with *basic* rules to generate potential targets. If no *data type* rules achieve a coverage percentage above the minimum threshold for that type, then no *helper* rule is used.

Basic Rules

For each labeled instance of a target in the training data, a *basic* rule is created which consists of three parts: case, part-of-speech (POS), and semantic (SEM). The POS information is provided by NLTK's part-of-speech tagger and the SEM informa-

Data Type	**Method for Extraction**
time	Regular expressions
number, *int*, *float*	Python type checking
named entity, *person*, *organization*, *location*, *date*, *money*	Stanford Named Entity Recognizer

Table 4.6: The built-in *data type* rules.

tion comes from Stanford NER. Table 4.7 shows what a *basic* rule would look like for the text "Prof. John Cool". Once a *basic* rule has been generated for every positive example of a target in the training data, the rules are reduced so that there are no duplicates. All targets found by both the *helper* rule (if one exists) and the collection of *basic* rules are viewed as potential targets.

	Prof.	John	Cool
Case	`UPPPER`	`UPPPER`	`UPPPER`
POS	`NNP`	`NNP`	`NNP`
SEM	`O`	`PERSON`	`PERSON`

Table 4.7: A *basic* rule for the text "Prof. John Cool".

This method of rule-creation is designed to avoid over-fitting in two ways. First, unlike other systems that generate rules using surrounding text, this system uses only target text. The reasoning behind this is that there are many ways to express an idea in free text, and restricting the system to finding contextual patterns from training documents is potentially limiting. Since the data type of the target generally does not change and its context does, it makes more sense just to look for possible instances of the target and then determine if the context is correct later using less restrictive statistical methods. Second, although *basic* rules are created directly from patterns in the text, *helper* rules are generated using patterns in data types, so the system does not rely solely on the way targets are portrayed in the training documents.

4.3.2 Feature Selection

For each resulting potential target, three sets of features are created: *prev*, *post*, and *target* features.

Target Features

Target features are generated from just the text of a potential target (i.e. no contextual information is used). The types of features included in the *target* feature set heavily depends on the data type predicted during rule generation. The purpose of *target* features is to gather information that can be used to tell if the potential target looks like the correct data type. Consider the task of extracting starting times from the CMU Seminar Announcements corpus. Times can be written and understood by humans in various forms (e.g. 4:00pm, 4:00, 4 pm, etc). Assume that during rule-generation, the system encounters a starting time in the form "4 pm". The rule that would be created from this example is shown in Table 4.8. With this rule in place, any piece of text that matches will be extracted as a potential target. This is problematic, since any occurrence of a number proceeded by a lower case singular noun will be considered a potential target. However, we cannot simply discard this rule, because it has already been proven to capture positive examples. In this case, *target* features become very useful, since they can provide information on the similarity of a potential target to positive examples.

	4	pm
Case	OTHER	LOWER
POS	CD	NN
SEM	O	O

Table 4.8: A *basic* rule for the text "4 pm".

Prev and *Post* Features

Prev and *Post* features are collected in order to analyze the context of a potential target within a document. *Prev* features are based on the words that precede a potential target, beginning with the first word of the sentence that contains the entity and ending with the word that directly precedes the entity. Sentence boundaries are discovered using NLTK's sentence tokenizer. If sentence boundaries cannot be detected, as may be the case with structured or semi-structured text, than *prev* features are based on up to ten words directly preceding a potential target. The same process is used to derive *post* features, except that these are based on words that proceed a potential target. The same feature types are collected for both *prev* and *post* features, and they are not affected by the predicted data type of a potential target. Table 4.9 shows the information collected by *prev* and *post* features.

Feature	Description
NUM_WORDS	number of words preceding or proceeding a target (used only if sentence boundaries are found)
CLOSEST_WORD	nearest alphanumeric word preceding or proceeding a target, stemmed, numbers replaced with generic <NUM> tokens
CLOSEST_PUNCT	non-alphanumeric word immediately preceding or proceeding a target, if one exists
CLOSEST_TRIGRAM	trigram immediately preceding or proceeding a target, stemmed, numbers replaced with generic <NUM> tokens
CLOSEST_POS_TRIGRAM	part-of-speech trigram immediately preceding or proceeding a target
CLOSEST_PREPOSITION	preposition nearest to the start or end of a target
CLOSEST_VERB	verb nearest to the start or end of a target
CLOSEST_NOUN	noun nearest to the start or end of a target

Table 4.9: *Prev* and *post* features.

4.3.3 Classifiers

A Naive Bayes classifier is defined for each set of features for a total of three classifiers: a *target* classifier, *prev* classifier, and *post* classifier. Once each classifier has been trained, the system is ready to extract information from unlabeled documents.

4.4 Extraction

When given a set of unlabeled documents, the system uses the rules generated during training to produce a set of potential entities. Features are extracted in the same way they were during training, and passed to the appropriate classifier. If all three classifiers positively classify the feature sets of a potential target, then it is added to the set of results. Figure 4.2 shows an overview of the entire information extraction process.

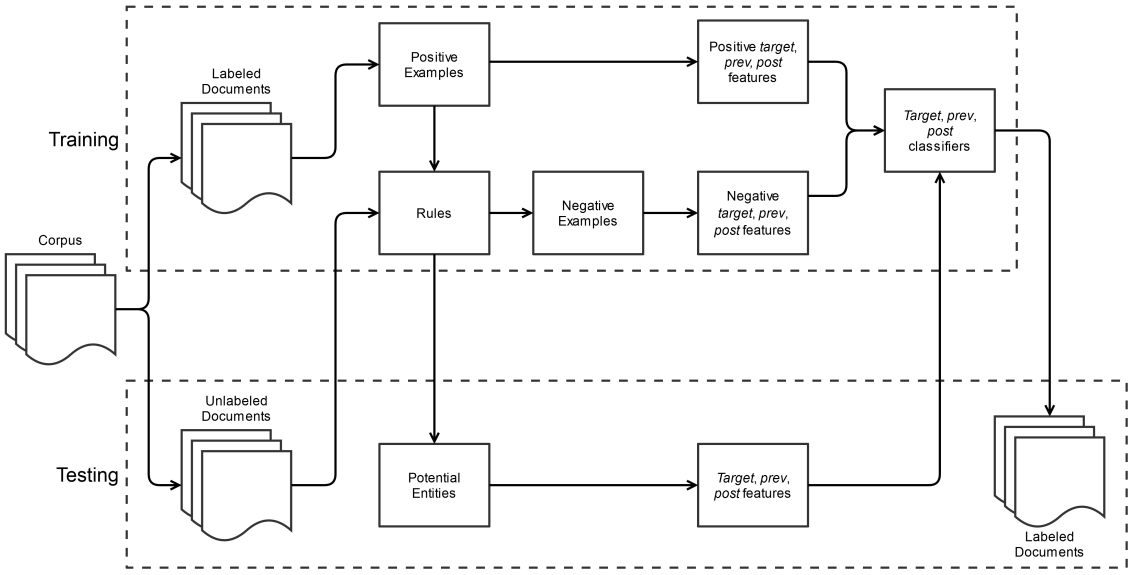

Figure 4.2: The entire information extraction process.

Chapter 5

Experiment

When testing the system, a random 50/50 split of training and testing data was used, since this was the most popular ratio used by comparable IE systems. All reported results were obtained by taking the average of performance measures from ten repetitions of randomly splitting data and extracting results.

5.1 Exact vs. Loose Matches

An issue that requires consideration during evaluation is determining what constitutes a true positive. One standard that can be used is to count only exact matches. An exact match occurs when the starting and ending indexes of a result exactly match those of an expected target. For instance, if "Prof. John Cool" with a starting index of 42 was an expected target and the system returned "Prof. John" with the same starting index, then "John Cool" would be counted as a false positive, and the missing "Prof. John Cool" would be counted as a false negative. This approach is very conservative and is used by all of the systems discussed in the "Related Work" section.

Another method is to allow "loose" matches to count as true positives. A result is a "loose" match if any part of it's text is within the boundaries of an expected target. In that case, "John Cool" from the previous example would be counted as a

true positive, and no false positives or false negatives would be produced. However, if the system returned "John Cool" with a starting index of 213, that would not be considered a match.

The system was tested using both exact and loose matches in order to provide an understanding of how well boundary detection is achieved.

5.2 All Slot vs. Single Slot Occurrences

Another question is how to count correct extractions and errors. One method is to expect the system to return all slot occurrences (ASO). In other words, if a document contains three targets for a single label, even if the value is the same, the system is expected to return all three. For instance, if a document from the Seminar Announcements corpus has two labeled ending times (e.g. "4:00pm" at starting index 23 and "4:00pm" at starting index 55), if the system only returns one, then that one is counted a true positive, and the other is counted as a false negative. One important aspect of this approach is that it is highly dependent upon the consistency of annotations. This is the primary approach used by this system, since most of the corpora allow multiple values for a single field. ELIE also prefers this system [9].

Another approach is to only expect a single slot occurrence (SSO) for each field. This method operates on the idea that there is only one value per label in each document. In the previous example with the two ending times, if the system only returns one, then that is counted as a true positive, and no false negatives are added. Despite the fact that this method limits systems to extracting information that only occurs once in a document, it is used by most IE systems for testing [31].

The system was tested using mostly ASO, although the SSO results for the Seminar Announcements corpus have been provided for the sake of even comparison with other systems.

5.3 Questions

The experiment aims to answer the following questions:

1. **How does the system perform compared to other systems?**
2. **How well does system perform boundary detection?**
3. **How well do rules capture expected results?**
4. **How does the number of training documents impact system performance?**
5. **How does the system perform with other classifiers?**

Chapter 6

Results

6.1 How Does the System Perform Compared to Other Systems?

6.1.1 ASO

Table 6.1 shows the system's precision, recall, and F1 scores using ASO extraction with both exact and loose matching. The Seattle Times Rentals corpus had the best overall F1 scores, averaging 82.8 for exact matching and 83.4 for loose matching. The best F1 performance for a single label was 94.6 for `etime`. The system performed significantly better on structured and semi-structured text than it did on unstructured text.

Corpus	Field	ASO Exact			ASO Loose		
		P	R	F1	P	R	F1
CMU Seminar Announcements	speaker	66.0	43.2	51.8	68.6	44.8	53.8
	location	62.1	47.6	53.6	68.6	52.4	59.1
	stime	95.7	77.7	85.7	96.3	78.2	86.2
	etime	98.5	87.9	92.9	99.3	88.6	93.6
Seattle Times Rentals	neighborhood	97.5	66.3	78.9	99.9	67.9	80.7
	bedrooms	84.8	77.8	81.1	84.8	77.8	81.1
	price	97.4	80.9	88.3	97.4	80.9	88.3
GENIA	protein	26.1	26.0	26.1	43.8	43.7	43.8
	entity	28.5	49.5	36.1	35.0	60.7	44.4
10-K's	pre	27.6	43.8	32.2	27.6	43.8	32.2

Table 6.1: ASO results. The headers P, R, and F1 stand for precision, recall, and F1-score respectively.

Table 6.2 displays the performance results from LP^2 and ELIE on the Seminar Announcements corpus using ASO and exact matching. Both LP^2 and ELIE perform significantly better for `speaker` and `location`. They also outperform the system for `stime` and `etime`, but the results are much closer.

Field	**System**			**LP²**			**ELIE**		
	P	R	F1	P	R	F1	P	R	F1
speaker	66.0	43.2	51.8	71.4	68.7	70.0	84.6	85.2	**84.9**
location	62.1	47.6	53.6	87.2	68.3	76.6	90.0	82.2	**85.9**
stime	95.7	77.7	85.7	89.0	87.7	88.3	84.7	96.3	**90.2**
etime	98.5	87.9	92.9	95.4	86.5	90.8	94.8	94.4	**94.6**

Table 6.2: A comparison of precision (P), recall (R), and F1-measures (F1) for LP^2 and ELIE on the CMU Seminar Announcements corpus using all slot occurrence (ASO) with exact matching [9].

Unfortunately, it was impossible to obtain any of the other corpora used to test BWI, LP^2, and ELIE, with the exception of the Jobs corpus, which consists of 300 newsgroup messages detailing jobs available in the Austin area. However, it seemed redundant to compare performance on the Jobs corpus, since it is also semi-structured text and it was decided that the time it would have taken to hand-annotate the corpus in standoff format could be better used. In addition, although the Rental Ad and GENIA corpora did come from outside sources, the performance of other systems on that data was incomparable to the results from this system.

Since not all corpora used by this system were tested by other systems, a baseline for performance was created for the sake of comparison. To produce baseline measurements, documents were word-tokenized and the pool of potential targets consisted of all possible n-grams within range of a label's average word length. The features passed to the *prev*, *post*, and *target* classifiers were the respective unigrams, stemmed and with stop-words removed. Baseline results are shown in Table 6.3. Generally, the system greatly outperforms the baseline.

Corpus	Field	Baseline		System	
		Exact	Loose	Exact	Loose
CMU Seminar Announcements	speaker	16.2	32.4	51.8	53.8
	location	3.5	45.3	53.6	59.1
	stime	0.0	33.9	85.7	86.2
	etime	1.5	15.1	92.9	93.6
Seattle Times Rentals	neighborhood	69.8	79.8	78.9	80.7
	bedrooms	48.7	48.7	81.1	81.1
	price	39.4	39.4	88.3	88.3
GENIA	protein	2.3	29.5	26.1	43.8
	entity	40.1	42.4	36.1	44.4
10-K's	pre	10.2	23.4	32.2	32.2

Table 6.3: ASO Baseline performance for exact and loose matching.

6.1.2 SSO

Table 6.4 shows the system's precision, recall, and F1 scores using SSO extraction with both exact and loose matching. Only the Seminar Announcements corpus is shown because it does not make sense to use SSO for the other corpora. Technically, it does not make sense to use SSO with `speaker`, but since other systems have reported SSO results for it, they are included. The system's SSO performance is much better than its ASO performance. This is expected, since it is much easier to return a single correct result for a label than finding all possible correct results. The best performance for a single label was achieved by `stime`, and `etime` came in very close second.

Corpus	**Field**	**SSO Exact**			**SSO Loose**		
		P	R	F1	P	R	F1
CMU Seminar Announcements	`speaker`	73.4	49.3	58.8	75.2	50.5	60.2
	`location`	68.1	44.1	53.4	77.9	50.4	61.0
	`stime`	99.3	99.3	99.3	99.4	99.4	99.4
	`etime`	99.3	97.6	98.4	99.6	97.9	98.7

Table 6.4: SSO results. The headers P, R, and F1 stand for precision, recall, and F1-score respectively.

Table 6.5 displays the performance results from BWI, LP^2, and ELIE on the Seminar Announcements corpus using SSO and exact matching. All systems perform better for `speaker` and `location`. However, this system outperforms LP^2 and ELIE for exact matching on `stime` as well as all three systems for `etime`.

Field	BWI			LP²			ELIE		
	P	R	F1	P	R	F1	P	R	F1
speaker	79.1	59.2	67.7	87.0	70.0	77.6	91.0	86.0	**88.5**
location	85.4	69.6	76.7	87.0	66.0	75.1	93.1	80.7	**86.5**
stime	99.6	99.6	**99.6**	99.0	99.0	99.0	98.6	98.5	98.5
etime	94.4	94.4	94.4	94.0	97.0	95.5	95.7	97.3	**96.4**

Table 6.5: A comparison of precision (P), recall (R), and F1-measures (F1) for BWI, LP², and ELIE on the CMU Seminar Announcements corpus using single slot occurrence (SSO) [11, 6, 9].

6.2 How Well Does the System Perform Boundary Detection?

Boundary detection is the process of finding the correct starting and ending points for a target. If a system performs well at boundary detection, the targets returned will be match exactly to the expected output, while a less adept system will return targets that only loosely match.

The gaps between exact and loose F1 scores in Table 6.6 provide some information on how well boundary detection is accomplished. The gaps are largest for `protein`, `entity`, and `location`, whose targets do not have highly formatted data types, if any. Conversely, `bedrooms`, `price`, and `pre` have no gaps at all, closely followed by `stime` and `etime`. Based on this data, it can be said that the system performs boundary detection very well, and in some cases perfectly, on targets with structured data types such as times or dollar amounts. However, as targets become less structured, boundary detection performance deteriorates.

Corpus	Field	ASO		
		Exact	Loose	Difference
CMU Seminar Announcements	speaker	51.8	53.8	2.0
	location	53.6	59.1	5.5
	stime	85.7	86.2	0.5
	etime	92.9	93.6	0.7
Seattle Times Rentals	neighborhood	78.9	80.7	1.8
	bedrooms	81.1	81.1	0.0
	price	88.3	88.3	0.0
GENIA	protein	26.1	43.8	17.7
	entity	36.1	44.4	8.3
10-K's	pre	32.2	32.2	0.0

Table 6.6: A comparison of the gaps between F1 scores for exact and loose matches using ASO.

6.3 How Well do Rules Capture Expected Results?

Table 6.7 shows what precision and recall would be if the system just used the potential targets returned by rules without collecting features and running them through classifiers. Columns A and B show the results of using both *basic* and *helper* rules with exact and loose matching, respectively. Columns C and D show the results of using only *basic* rules to produce potential targets with exact and loose matching, respectively. The goal of rules should be to achieve the highest recall possible, since they provide the pool of possible results for the classifiers to choose from. In other words, the maximum F1 score achievable by the entire system is directly related to the recall score of the rules alone. As expected, loose matching allows for higher recall than exact matching in almost all cases. Additionally, all recall values for loose matching with both *basic* and *helper* rules are above 96%, which is very good. The data also shows that in almost every case using *helper* rules is beneficial to performance.

Corpus	Field	A	B	C	D
CMU Seminar Announcements	speaker	64.6	**98.5**	68.8	98.3
	location	69.6	**96.7**	67.2	94.1
	stime	97.7	**99.8**	95.9	99.6
	etime	97.6	**99.5**	95.7	98.7
Seattle Times Rentals	neighborhood	95.7	**99.3**	92.7	98.2
	bedrooms	**99.0**	**99.0**	**99.0**	**99.0**
	price	**100.0**	**100.0**	99.3	99.3
GENIA	protein	35.2	**99.6**	35.4	99.5
	entity	36.5	**98.8**	33.3	96.3
10-K's	pre	94.4	**100.0**	79.6	83.2

Table 6.7: Recall results when only rules are used without classifiers to make predictions. Four types of ASO results are shown: (A) exact matches found with helper and basic rules, (B) loose matches found with helper and basic rules, (C) exact matches found using only basic rules, (D) loose matches found using only basic rules.

Table 6.8 shows how much precision is improved by using classifiers instead of just rules. As expected, classifiers vastly improve precision in all cases.

Corpus	Field	A	B	C	D	E	F
CMU Seminar Announcements	speaker	3.1	4.9	3.4	5.0	66.0	**68.8**
	location	2.0	2.8	2.3	3.1	62.1	**68.6**
	stime	2.4	2.5	3.2	3.3	95.7	**96.3**
	etime	1.8	1.9	10.3	10.6	98.5	**99.3**
Seattle Times Rentals	neighborhood	22.4	23.2	21.1	22.5	97.5	**99.9**
	bedrooms	4.3	4.3	4.6	4.6	**84.8**	**84.8**
	price	12.1	12.1	8.6	8.6	**97.4**	**97.4**
GENIA	protein	3.0	8.9	3.0	8.6	26.1	**43.8**
	entity	0.2	0.5	0.2	0.6	28.5	**35.0**
10-K's	pre	3.4	3.6	4.9	5.1	**27.6**	**27.6**

Table 6.8: A comparison of precision results when only rules are used without classifiers to make predictions versus the precision results of the entire system. Four types of ASO results using rules only are shown: (A) exact matches found with helper and basic rules, (B) loose matches found with helper and basic rules, (C) exact matches found using only basic rules, (D) loose matches found using only basic rules. The E and F columns represent the precision of the entire system using exact and loose matching respectively.

6.4 How Does the System Perform with Other Classifiers?

Table 6.9 shows the ASO results with exact and loose matching using Maximum Entropy (20 iterations), Decision Tree, and Naive Bayes classifiers. Naive Bayes is clearly the dominant choice, outperforming the other classifiers in most cases.

Corpus	**Field**	**MaxEnt**		**DecTree**		**NBayes**	
		E	L	E	L	E	L
CMU Seminar Announcements	`speaker`	31.1	40.5	25.6	25.6	**51.8**	**53.8**
	`location`	51.6	**63.5**	36.9	38.8	**53.6**	59.1
	`stime`	**90.5**	**91.6**	75.3	75.5	85.7	86.2
	`etime`	91.9	92.9	79.2	79.5	**92.9**	**93.6**
Seattle Times Rentals	`neighborhood`	78.1	79.8	52.4	52.4	**78.9**	**80.7**
	`bedrooms`	**83.4**	**83.4**	33.0	33.0	81.1	81.1
	`price`	86.8	86.8	88.1	88.1	**88.3**	**88.3**
GENIA	`protein`	9.5	27.6	8.3	9.2	**26.1**	**43.8**
	`entity`	7.7	13.0	**71.7**	**72.5**	36.1	44.4
10-K's	`pre`	27.9	27.9	8.1	8.1	**32.2**	**32.2**

Table 6.9: ASO results for Maximum Entropy (MaxEnt), Decision Tree (DecTree), and Naive Bayes (NBayes) classifers using exact (E) and loose (L) matching.

6.5 How Does the Number of Training Documents Impact System Performance?

Figures 6.1, 6.2, 6.3, and 6.4 show how the F1 scores change with different numbers of training documents for each corpus. In all cases, F1 score demonstrates logarithmic growth and generally begins to level out after 40 to 80 documents are used for training. In some cases, F1-scores drop when the number of training documents increases. This could be attributed to increased noise in the data.

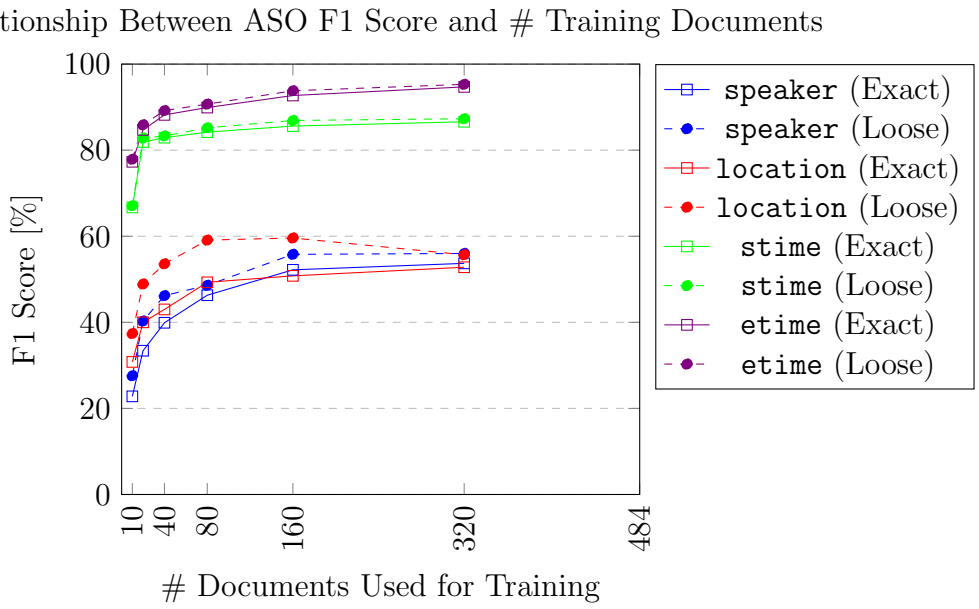

Figure 6.1: F1 scores using increasing amounts of training data for the CMU Seminar Announcements corpus.

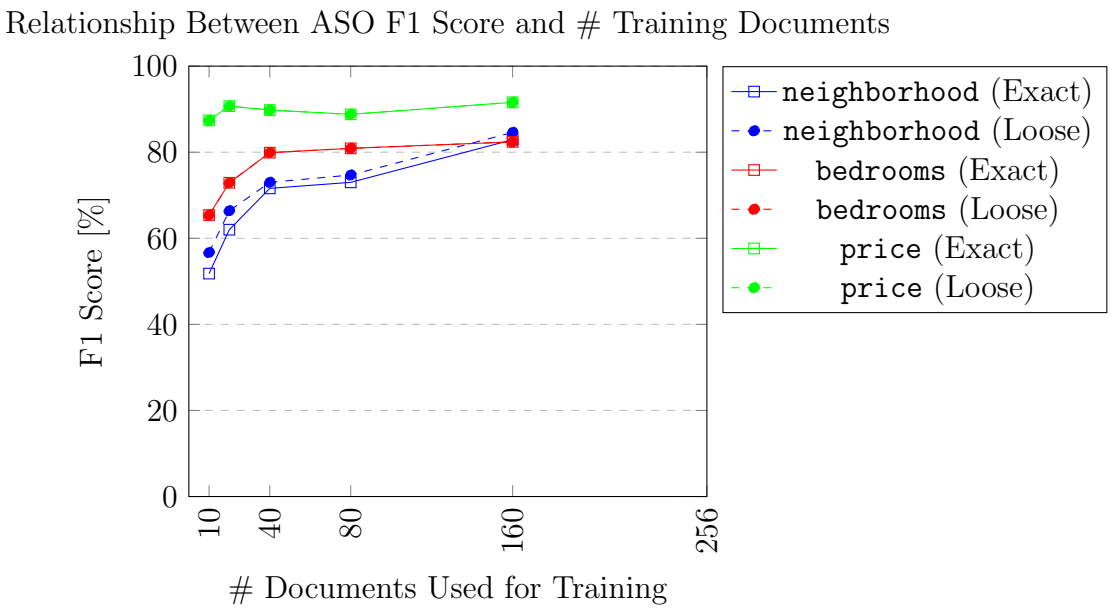

Figure 6.2: F1 scores using increasing amounts of training data for the Seattle Times Rentals corpus.

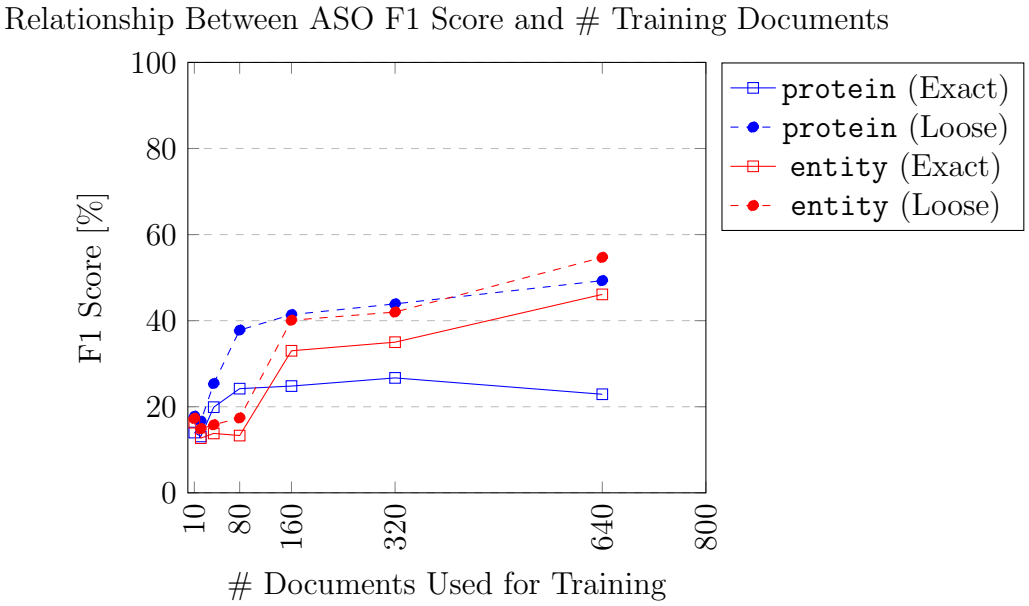

Figure 6.3: F1 scores using increasing amounts of training data for the GENIA corpus.

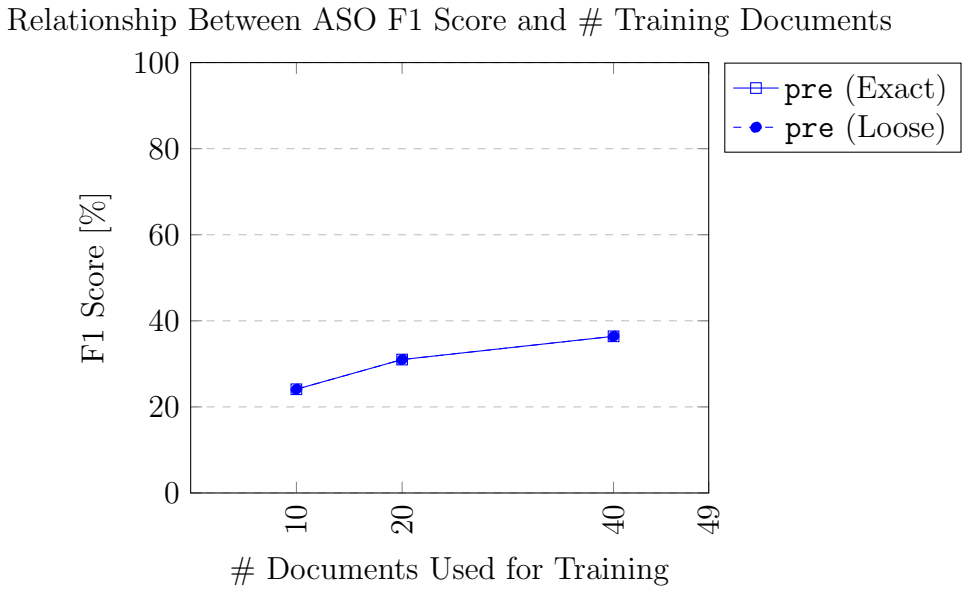

Figure 6.4: F1 scores using increasing amounts of training data for the 10-K corpus.

Chapter 7

Future Work

The primary goal of any future work would be to improve performance. One possible solution might be to try using different sets of features. Although many combinations of features were experimented with during development, it is possible that there exist features that were not considered.

Another way to improve the system is to allow custom data types. Consider the `speaker` label from the Seminar Announcements corpus. It seems correct that this label falls under the `PERSON` data type, but this actually leads to poor recall for exact matching, since the labeled targets typically include titles such as "Professor" or "Dr.", which are not considered part of a person's name by Stanford NER. In this case, a custom data type such as TITLED PERSON could improve performance.

Chapter 8

Conclusions

In this project, an IE system that combines techniques from rule-based and statistical IE systems is developed. It begins with a rule-based approach by creating rules for targets based on training data, and also introduces a new approach of using target data types as a secondary form of rule. Once the rules have been used to extract potential entities, statistical methods are introduced by using classifiers to analyze target and contextual data to predict results. This hybrid system is able to avoid certain pitfalls faced by its singular counterparts, including over-fitting to training data and ignoring target features.

Best performance is observed when targets belong to structured data types and when important contextual details are nearby. In these cases, results from the system were close to, if not better than, other state-of-the-art systems, especially when using SSO. Unfortunately, in other cases the system performs considerably worse and significant improvement is needed before the system can be considered a competitive solution.

BIBLIOGRAPHY

[1] Cal Poly Github. http://www.github.com/CalPoly.

[2] C. Aone, L. Halverson, T. Hampton, and M. Ramos-Santacruz. SRA: Description of the IE2 system used for MUC-7. 1998.

[3] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[4] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. *Working Papers of ACL-97 Workshop on Natural Language Learning*, 1997.

[5] H. L. Chieu and H. T. Ng. A maximum entropy approach to information extraction from semi-structured and free text. *AAAI/IAAI*, 2002.

[6] F. Ciravegna and A. Lavelli. LearningPinocchio: Adaptive information extraction for real world applications. *Nat. Lang. Eng.*, 10(2):145–165, June 2004.

[7] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *LREC*, 2004.

[8] J. R. Finkel. Named entity recognition and the stanford ner software. *2007-03-01]. http://nlp. stanford. edu/software/jenny-ner-2007. pdf*, 2007.

[9] A. Finn and N. Kushmerick. Information extraction by convergent boundary classification. In *Proceedings of the AAAI Workshop on Adaptive Text Extraction and Mining*, 2004.

[10] D. Freitag. Toward general-purpose learning for information extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 404–408. Association for Computational Linguistics, 1998.

[11] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *AAAI/IAAI*, pages 577–583, 2000.

[12] D. Freitag and A. McCallum. Information extraction with HMM structures learned by stochastic optimization. *AAAI/IAAI*, 2000:584–589, 2000.

[13] R. Grishman and B. Sundheim. Message understanding conference-6: A brief history. In *COLING*, volume 96, pages 466–471, 1996.

[14] J.-D. Kim, Y. Wang, T. Takagi, and A. Yonezawa. Overview of genia event task in bionlp shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 7–15. Association for Computational Linguistics, 2011.

[15] N. Kushmerick. *Wrapper induction for information extraction*. PhD thesis, University of Washington, 1997.

[16] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[17] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[18] A. McCallum and D. Jensen. A note on the unification of information extraction and data mining using conditional-probability, relational models. *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, 2003.

[19] S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 226–233, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[20] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[21] I. Muslea. Rise: Repository of online information sources used in information extraction tasks, 1998. *Published on http://www.isi.edu/info-agents/RISE*, 2005.

[22] I. Muslea et al. Extraction patterns for information extraction tasks: A survey. In *The AAAI-99 Workshop on Machine Learning for Information Extraction*, volume 2, 1999.

[23] I. Muslea, S. Minton, and C. A. Knoblock. Active learning with strong and weak views: A case study on wrapper induction. In *IJCAI*, volume 3, pages 415–420, 2003.

[24] L. Peshkin and A. Pfeffer. Bayesian information extraction network. *arXiv preprint cs/0306039*, 2003.

[25] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049, 1996.

[26] E. Riloff et al. Automatically constructing a dictionary for information extraction tasks. In *AAAI*, pages 811–816, 1993.

[27] D. Roth and W.-t. Yih. Probabilistic reasoning for entity & relation recognition. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[28] S. Sarawagi. Information extraction. *Found. Trends databases*, 1(3):261–377, Mar. 2008.

[29] T. Scheffer, C. Decomain, and S. Wrobel. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[30] C. Siefkes. Incremental information extraction using tree-based context representations. In *Computational Linguistics and Intelligent Text Processing*, pages 510–521. Springer, 2005.

[31] C. Siefkes and P. Siniakov. Journal on data semantics iv. chapter An Overview and Classification of Adaptive Approaches to Information Extraction, pages 172–212. Springer-Verlag, Berlin, Heidelberg, 2005.

[32] M. Skounakis, M. Craven, and S. Ray. Hierarchical hidden markov models for information extraction. In *IJCAI*, pages 427–433, 2003.

[33] S. Soderland. Learning information extraction rules for semi-structured and free text. *Mach. Learn.*, 34(1-3):233–272, Feb. 1999.

[34] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a conceptual dictionary. *arXiv preprint cmp-lg/9505020*, 1995.

[35] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii. BRAT: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics, 2012.

[36] J. Tang, M. Hong, D. Zhang, B. Liang, J. Li, et al. Information extraction: Methodologies and applications. *Emerging Technologies of Text Mining: Techniques and Applications*, 2007.

[37] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

[38] J. Zavrel and W. Daelemans. Feature-rich memory-based classification for shallow NLP and information extraction. In *Text Mining, Theoretical Aspects and Applications*, pages 33–54. Springer Physica, 2003.