Mining Wikipedia for Geospatial Entities and Relationships

by

Jeremy T. Witmer

B.S., University of Colorado, Colorado Springs, 2005

A thesis submitted to the Graduate Faculty of the University of Colorado at Colorado Springs

in partial fulfillment of the requirements for the degree of Master of Science

Department of Computer Science

2009
This thesis for Master of Science degree by

Jeremy T. Witmer

has been approved for the

Department of Computer Science

by

Dr. Jugal Kalita, Chair

Dr. C. Edward Chow

Dr. Xiaobo Zhou

Dr. Thomas Huber

Dr. Paddington Hodza

Date
This research work seeks to address the challenge of extracting geospatial data from the article text of the English Wikipedia. We develop in this paper a software package which extracts geospatial named entities (place names) from Wikipedia articles, geocodes each geospatial named entity to a coordinate pair, and uses that named entity information to identify relationships between the geospatial named entities and other significant phrases, nouns and verbs in the same sentence.

In the first phase of our work, we create a training corpus and select a set of word-based features to train a Support Vector Machine (SVM) for the task of geospatial named entity recognition. We target for testing a corpus of Wikipedia article pages about battles and wars, as these have a high incidence of geospatial content. The SVM recognizes place names in the corpus with a very high recall, close to 100%, with an acceptable precision.

The set of geospatial named entities is then fed into a geocoding and resolution process, whose goal is to determine the coordinate pair for each place name. As many of the named entities are ambiguous, and do not immediately geocode to a single location, we present a data structure and algorithm to resolve the ambiguous named entities based on sentence and article context, so the named entity can be correctly geocoded. We achieve a high f-measure performance for the first two phases of this research, and create an associated list of geospatial entities for each article,
combining the place names, spatial locations, and an assumed point geometry.

In the final phase, we create a feature set and training corpus to train a second SVM for geospatial relationship extraction. Using the geospatial named entity information from the first two phases of this research, we train an SVM based on the positions of the named entities and simple part of speech, significant phrase, and sentence parsing features. This SVM identifies relationships between the geospatial named entities and other significant phrases, nouns, and verbs within the sentence. We achieve a performance comparable to current relation identification approaches, using only a simple feature set.
Dedicated to my beautiful wife Joanna,

the one I love with all my heart.
Acknowledgments

First, I would like to express my gratitude to Dr. Jugal Kalita, for his help and support as my thesis advisor. For the advice, the editing, the proofreading, the ideas, and for always pushing me to go a little bit farther and excel, thank you.

To the members of my thesis committee, Dr. Edward Chow, Dr. Xiaobo Zhou, Dr. Thomas Huber, and Dr. Paddington Hodza. Thank you for time, feedback, and support in making this thesis possible.

To Suzette Stoutenburg and Mona Habib, thank you for your resources for SVM and NER work, for advice and discussion, and all the time and assistance to my research. It has been a pleasure working with you.

Thanks also goes to Dr. Kalita’s research group, especially Justin Gray, Beaux Sharifi, and Michael Bankston for your feedback, support, ideas, and humor through this process.

To the faculty and staff of the Computer Science department at UCCS, thank you for eight years of learning. It has been my pleasure to learn from and work with you through the course of my BS and MS degrees.

Thanks to the developers of the software packages listed in Appendix A, all of which proved invaluable in this research work. A special thanks goes to the JRuby team for porting Ruby to the JVM, and to the LibSVM team for creating an excellent library for training SVMs.

To my parents, who bought us our first computer when I was 13, and started me down this road. Thank you for your prayers, support, time, and everything
that you’ve done behind the scenes through the years to make this accomplishment possible.

To my brother Nathan for getting me into computers in the first place, and all the late night coding tips and tricks. To my sister Caroline, for your prayers and support.

To John and Debbie, and the rest of the Merritt family, for the support and prayers, thank you.

Last and most important, my love and thanks goes to my wife Joanna, who stood by me through this whole process and all the late nights, and supported me unwaveringly. You are the light of my life.

Finally, to Jesus Christ, who provides grace and strength for each day, and who enabled me to complete this work.
# Contents

1 **Introduction** 1  
1.1 Motivation 4  
1.2 Approach Overview 7  
1.3 System Architecture 9  
1.4 Success Criteria 10  
1.5 Thesis Outline 12  

2 **Named Entity Recognition** 14  
2.1 Overview 14  
2.2 Performance Measures 16  
2.3 Features for Named Entity Recognition 17  
2.4 Approaches and Current Related Research 17  
2.5 Challenges in Named Entity Recognition 23  
2.6 Named Entity Recognition Applied to Wikipedia 25  
2.7 Location-Specific Named Entity Resolution 26  

3 **Support Vector Machines** 29  
3.1 Overview and Theory 29  
3.2 Performance Measures 35  
3.3 SVM Challenges 36  
3.4 Named Entity Recognition using SVM 38  

4 **Wikipedia Background and Corpus Preparation** 41
5 Geospatial Named Entity Extraction

5.1 Overview and Objective ........................................... 51
5.2 SVM Feature Selection ........................................... 52
5.3 Training Corpus Generation .................................... 56
5.4 Testing Corpus Generation ..................................... 58
5.5 SVM Training ....................................................... 61
5.6 Testing and Results ............................................. 62
5.7 Discussion ......................................................... 63

6 Extracted Geospatial Named Entity Resolution

6.1 Overview and Objective ........................................... 67
6.2 Mining Geospatial Entities ..................................... 70
6.3 Google Geocoder ................................................... 75
6.4 Location Tree Data Structure ................................ 78
6.5 Results ............................................................... 85
6.6 Discussion ......................................................... 87

7 Open Geospatial Relation Identification

7.1 Overview and Objective ........................................... 91
7.2 Feature Selection .............................................. 97
7.3 Corpus Generation .............................................. 99
7.4 SVM Training .................................................. 101
7.5 Results .......................................................... 102
7.6 Discussion ...................................................... 104

8 Conclusion ......................................................... 107
  8.1 Known Issues and Solutions ................................. 109
  8.2 Evaluation of Success Criteria ............................. 112
  8.3 Contributions ................................................ 113
  8.4 Future Work .................................................. 115

References .......................................................... 119

Appendix A: Software Packages and Versions .................. 126
# List of Tables

1. Features used for Named Entity Recognition ............................................. 18
2. Results From the CoNLL 2003 Shared Task Sorted by English Performance ................................................................. 22
3. Partial Wikipedia Corpus Article Metrics ............................................... 50
4. NER SVM Results Using Variable Windows in the Text ............................ 54
5. Named Entity Recognition Results from the SVM ..................................... 63
6. Training Corpus Combination Results for the NER SVM .......................... 64
7. Feature Set Combination Results for the NER SVM .................................. 65
8. Places With Multiple Names and Names Applied to More Than One Place in the Getty Thesaurus of Geographic Names ............................. 69
9. Physical $(\phi, \lambda)$ Coordinates for Geospatial Entities from the Operation Nickel Grass Article ................................................................. 86
10. Raw SVM and Resolved Geospatial NE Results ..................................... 86
11. Part of Speech Tag Set for this Research ............................................... 100
12. Open Geospatial Relation Extraction Results ....................................... 103
13. Feature Set Combination Results for the Relation SVM ........................... 106
List of Figures

1. Processing Flow in the Geografikos Package ............................................. 9
2. External Software and Web Service Dependencies ............................ 10
3. Linearly Separable Binary SVM Hyperplane ........................................ 31
4. Linearly Non-Separable Binary SVM Hyperplane .................................. 33
5. Wikipedia Article Database Structure ..................................................... 45
   Article ........................................................................................................... 46
7. Example Wikipedia Article with Geospatial NEs Highlighted ............... 52
8. Battle of Gettysburg Place Name Detail (f-measure = 0.682) ............... 70
9. War of 1812 Place Name Detail (f-measure = 0.816) .......................... 71
10. World War II Place Name Detail (f-measure = 0.870) ........................ 71
11. Liberty Incident Place Name Detail (f-measure = 0.905) ..................... 72
12. The Property Categories of a Geospatial Entity .................................. 73
13. Google Geocoder ..................................................................................... 75
14. Example XML Response from Google Geocoder ............................... 77
<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Color-coding of the Nodes in the Location Tree Examples</td>
<td>80</td>
</tr>
<tr>
<td>16</td>
<td>Initial Location Tree</td>
<td>80</td>
</tr>
<tr>
<td>17</td>
<td>Location Tree After Adding Colorado, USA</td>
<td>80</td>
</tr>
<tr>
<td>18</td>
<td>Location Tree After Adding Denver, CO</td>
<td>81</td>
</tr>
<tr>
<td>19</td>
<td>Location Tree After Adding the USA, before Hierarchy Check</td>
<td>81</td>
</tr>
<tr>
<td>20</td>
<td>Location Tree After Hierarchy Check</td>
<td>82</td>
</tr>
<tr>
<td>21</td>
<td>Example Location Tree Showing Duplicated Node Count</td>
<td>83</td>
</tr>
<tr>
<td>22</td>
<td>Complete Location Tree for the Operation Nickel Grass Wikipedia Article</td>
<td>85</td>
</tr>
<tr>
<td>23</td>
<td>Detailed Results for a Subset of Wikipedia Articles</td>
<td>88</td>
</tr>
<tr>
<td>24</td>
<td>Wikipedia Article with Geospatial NEs and Relation Signifiers Highlighted</td>
<td>96</td>
</tr>
<tr>
<td>25</td>
<td>Detailed Relationship Results for a Subset of Wikipedia Articles</td>
<td>104</td>
</tr>
</tbody>
</table>
1 Introduction

Since the early 2000s, the amount of user-created data available on the Internet has increased at an exponential rate. With the ongoing rise of Web 2.0 and user-generated content, this movement shows no signs of slowing, and a greater and greater proportion of human knowledge is available on the Internet, especially in sites like Wikipedia, the *Free Encyclopedia*. Because most user-generated content is not marked up for semantic understanding, and it seems naive to expect users to do the extra work of semantic markup themselves, the challenge of automatically extracting machine-understandable data must be addressed. Tools and techniques are required that can perform natural language processing (NLP) over the free text of this user-generated content and extract information into a format that is directly machine understandable. In this paper, we present techniques for performing NLP on the free text content of the English Wikipedia to extract geospatial information.

The English Wikipedia represents an amazing amount of human knowledge and judgement. Its greatest strength lies in its openness, that any user may sign in and edit or contribute to any article. This openness has resulted in a collection of over 2.5 million articles in the English Wikipedia alone. However, Wikipedia content remains largely unstructured. While the content is marked up for display, there is very little structure around the content to allow direct machine understanding. Article titles, links between articles, and infoboxes are structured enough to directly impart limited information for machine understanding, but the majority of the text is unstructured. Therefore, more complex operations are required to extract machine-
understandable information from the text. This paper presents our research, results and contributions from the development of the Geografikos system, a software package for extracting and resolving geospatial and locative named entities (abbreviated NE or NEs) in Wikipedia pages, which are simply the names of places. These geospatial NEs are geocode, and assuming a point geometry, used to construct a list of geospatial entities that can be associated with the article text. Finally, the place names are used to extract relationships involving those entities. This set of resolved geospatial entities provides a base from which to build software that extracts further data from Wikipedia, and which can enhance existing search on Wikipedia articles.

For the purposes of our research, locations are described in text by place names, which define physical regions in space, defined by geographical or political criteria [22]. The Geografikos package identifies words and strings of text within the free text of Wikipedia articles that represent locations. For each identified geospatial NE identified in the article, our goal was to resolve the NE to a single (latitude, longitude) coordinate. While many locative NEs in the article text will be straightforward to resolve to a single coordinate pair, many of the entities will be ambiguous. For example, consider the geospatial NE ‘Cambridge’. Does this refer to Cambridge, a city in the UK, or Cambridge, a city in Massachusetts? We present our development of a novel data structure and an algorithm that relies on a small set of rules which uses the sentence and article context of each NE to answer this question, and resolve even ambiguous NEs to a single coordinate pair.

To our knowledge, this is the first time that geospatial NE extraction and resolution has been applied to the complete article text of the English Wikipedia. While
previous efforts have performed NE recognition on Wikipedia articles, we have extended that research a step further to actually geocode each NE and find a (latitude, longitude) coordinate pair.

While we selected Wikipedia to serve as our corpus for this research, we desired to build a tool with no dependence on the structure of Wikipedia. The Geografikos package operates only on the free text of each article, without using the structure, links, or other markup information in Wikipedia. Creating a tool that is corpus agnostic will potentially allow us to run any free text corpus through the Geografikos package for processing.

Furthermore, we targeted no specific domain or geographic area for the NE recognition and resolution process. Development and testing of the Geografikos package was performed against clusters of historical documents from Wikipedia, including battles, because these articles contain significant geospatial information, but the system is not limited to this kind of article, and can be run against any other article or free text corpus.

In the rest of this chapter, we present the motivation behind the development of the Geografikos package, and an overview of the approach taken in our research, software development, and testing. We also present our criteria for success, which guided our research and software development work.

As this work bridges the natural language processing domain with the geospatial and geographic domain, we take a moment here to refine our terminology. For the purposes of this research, the terms “geospatial NE” and “place name” are used interchangeably to describe a text description of a location, such as ‘Cambridge’.
The place name or geospatial NE is simply one attribute of a geospatial entity. The term “geospatial entity” is used to describe the combination of place name, spatial location, and point geometry that make up a geospatial entity that can be used from a geospatial standpoint for information retrieval and other operations. The full definition of a geospatial entity will be discussed in depth later in this paper. For the purposes of this paper, “feature”, “feature set”, and “feature vector” refer to word and sentence textual features for the SVM, not geographic features. Finally, latitude and longitude are often abbreviated as \((\phi, \lambda)\) in the geographic domain.

1.1 Motivation

As the amount of unstructured user-generated content on the Internet increases, the need to refine methods to extract information from it also increases. We must refine tools and techniques for automatically extracting data in a machine-understandable format. While research into the semantic web, and semantic markup of data for machine understandability is providing tools and techniques for this markup as content is created, most data on the Internet still remains unstructured. The extraction of geospatial information represents a good starting point for research into the automatic extraction of information from unstructured text, as a significant amount of queries on the web target geographical data. Google Trends\(^1\), and the increasing popularity of Google Maps\(^2\) indicates that queries that seek to find location data represent a significant fraction of queries for information.

\(^1\)http://www.google.com/trends/  
\(^2\)http://maps.google.com
Our research in this area was motivated by a number of factors. The ultimate goal of the development of the Geografikos package was a software system that will create a database of Wikipedia articles in which each article has an associated structured set of geospatial entities, extracted from the article. This database will allow geospatial querying, information retrieval, and visualization to be applied to the Wikipedia articles, and the Geografikos software will be made available as open source.

This article and location database will allow searches for text content to be filtered by geographic area. For instance, a user would be able to ask for “all articles relating to TV actors, limited to those in Europe”. The search could then be performed in two steps. The first would be a standard free text search on Wikipedia, for articles about TV actors. That list could then be further filtered to those articles with locations in Europe associated with them. Reversing this paradigm, the location data provided by the Geografikos package could also allow location-centric search. If a user were about to go on vacation to Beijing, China, they could be presented with an interface allowing them to select an area around Beijing, and the system could return all articles that have locations that fall into that bounding area.

Furthermore, this database of locations could enable the visualization of Wikipedia articles through a geospatial interface. For instance, consider an interface that would allow a user to select a Wikipedia article, and then present the user with a map of all the locations from the article. Each location on the map would be clickable, and provide the sentences or paragraph around that NE from the text. Imagine putting “World War II” into this interface, and being presented with a map of Europe, Africa,
and the Pacific theatre, with all the locations from the article marked and clickable. This kind of visualization would be an excellent teaching tool, and possibly reveal implicit information and relationships that are not apparent from the text of the article. Applied to other corpuses of information, this kind of information could also be very useful in finding geospatial trends and relationships. For instance, consider a database of textual disease outbreak reports. The Geografikos package could extract all the locations, and then they could be graphically presented on a map, allowing trends to be found much more easily. With additional work, the geospatial data extracted by the Geografikos package could be combined with temporal information. While many of these visualization tools already exist, they are driven from structured databases. Our contribution in this area is the generation of the article and geospatial entity association database, which can then drive any of these visualization tools.

This is just a small set of the possibilities for increased capability in geospatial information retrieval provided by associating a structured list of geospatial entities with a free text corpus.

From a research standpoint, we were motivated by a desire to provide advances in the areas of geospatial NE recognition, geospatial entity resolution, and relationship extraction involving geospatial NEs. While significant research has been done into applying support vector machines (SVMs) to the problem of NER (covered in depth in Chapter 3), we feel that there are advances to be made in using simple features to perform geospatial NE recognition using SVMs. Also, because the SVM will feed into the geocoding process, we wished to demonstrate an SVM with recall performance as close to 100% as possible, to ensure that all possible geospatial NEs were being
recognized in the text. We also focused on the disambiguation and spatial location resolution of place names. While limited research has been done in this area, it has not been heavily applied to geospatial NEs extracted from free text. In this research, we demonstrate a novel data structure and algorithm driven by a rule set that provides excellent performance in the geocoding of geospatial NEs. Finally, we wished to demonstrate advances in identifying natural language relations between place names, using an SVM with simple surface-level grammatical features, along with the extracted geospatial NE information.

1.2 Approach Overview

As stated previously, this research focused on the development of the Geografikos package. The system is designed to process each Wikipedia article individually, in no particular order, but we selected a set of articles about historical battles for the training and testing of the system, due to the high incidence of geospatial information. Geografikos processes each article in three separate phases. First, the NER SVM is run on the clean text of the article, to generate a list of candidate NEs. These candidate NEs are fed to the second stage of the system, which resolves and geocodes each entity. Finally, based on the resolved geospatial NE information, a second SVM is used to identify relations between the geospatial entities and other significant phrases, nouns, and verbs in the sentences.

The objective of the first stage of the Geografikos software was to develop a set of textual word-level features as input to an SVM that performs NER for geospatial
NEs, effectively extracting place names from free text. The feature set and training corpus was designed to provide a high recall on NER, while maintaining as high a precision as possible. Informed by current research and challenges using SVMs, we selected an optimal set of features and generated a training corpus combining hand-tagged data from a variety of sources, and automatically generated data from the Geonames database.

The objective of the second phase of processing was to apply and extend current research in geospatial entity disambiguation to decide whether each NE from the first phase was a geospatial NE, and if so, to choose the correct spatial location for that place name, in context of the article. We developed a data structure and algorithm to use the context surrounding each NE to provide disambiguation to a geocoding process.

The goal of the third phase was to use the geospatial NE information from the first two phases, along with part of speech and surface phrase parse information from the sentences to identify which sentences in each article contained relationships between the geospatial NEs and other significant phrases, nouns, and verbs in the sentence. While research has been done into relationship extraction using both surface features and deep dependency-based methods, we sought to demonstrate a feature set for an SVM that would allow simple surface-level parsing, but still provide better performance than the current approaches for geospatial relations.

Figure 1 shows a graphical view of the processing flow in the Geografikos package.
1.3 System Architecture

In the course of this research, for all the different phases, we wrote a set of tools combined together as the Geografikos package. We chose the name “Geografikos” as a combination of the Greek prefix “geo”, meaning earth or land, and the Greek word “graphikos”, which means writing. Geografikos is a software package designed to understand writing (digitally, of course) about geo-related information. Geografikos is a loosely-couple set of tools written in JRuby and Java. Figure 2 shows the external software and services on which Geografikos depends.

Google Geocoder is used for the geocoding of place names extracted from the text. LibSVM is used to train SVMs for phases 1 and 3, and to process the articles through the SVM. LingPipe is used for sentence identification for phase 3, and for
part of speech tagging for phases 1 and 3. The interface from Geografikos to these external packages is written to make it easy to use a different geocoding or NLP toolkit. The feature vector generation code is more tied to LibSVM, but that section of the Geografikos code could be adapted to a different SVM software package, without affecting the rest of the software.

Architecturally, Geografikos provides a series of self contained classes and methods for each step in phases 1-3, all dependent on a MySQL database for data storage. Each of the separate classes and methods for each phase could be changed at will, allowing us much flexibility in adjusting the system as we performed this research. To enable widespread refactoring, all of the critical functionality in the package is described by a series of RSpec unit tests, which define the external behavior of each class and module.

1.4 Success Criteria

The success of this thesis revolved around a set of objective criteria for the three stages defined in the previous section. From an objective standpoint, the performance
of the entity extraction and disambiguation, and relation detection was measured directly, through precision, recall, and f-measure, defined fully in Chapter 2. Precision is a measure of the accuracy of an extraction process. Recall is a measure of how broad the coverage of an extraction process, and f-measure is an overall performance measure, a ratio between precision and recall. Based on our research and stated objectives, we targeted as high a recall as possible for the initial NER extraction, as close to 100% as possible. The precision for the first stage was targeted at above 50%, resulting in a target f-measure of roughly 65%.

For the second stage of the processing, we expected recall to be lower than the first stage, due to information loss in the disambiguation and resolution process, but we targeted a significant increase in the precision for the second phase. We therefore targeted an overall f-measure of 80% for the SVM NER and geospatial resolution and disambiguation phases together. Based on current research, this f-measure represents an increase over current reported performance in the area of geospatial entity recognition and resolution.

Finally, for the third stage of the processing, based on the current best performance in relation extraction techniques to date, we targeted an f-measure of 50% for the identification of geospatial relations between the extracted NEs and other significant phrases in the sentence. This f-measure demonstrates an increase over current best published results to date.
1.5 Thesis Outline

We present a background summary of current and past research and results on NER in Chapter 2. We cover the various methods that have been applied to date, and their results, especially the CoNLL 2003 shared task results. After an overview of the performance measures normally used for NER, we examine feature selection, and the challenges in NER. We complete that chapter with a survey of research on NER applied to Wikipedia and location-specific NER.

We present an overview of SVM machine learning in Chapter 3, along with a simple introduction to the mathematical theory. We present the challenges in using SVMs, and then a brief overview of the current research in applying SVMs to NER.

Chapter 4 gives an introduction to Wikipedia, followed by an overview of the application of Wikipedia to our research. We then cover the database preparation and corpus selection from Wikipedia and tagging of the corpus.

In Chapter 5 we cover in depth our geospatial NER work. We present the features selected for the SVM, the training and testing corpus generation, and the training of the SVM. We conclude with a presentation of the results of the NER process and a discussion of the results and challenges.

We present our detailed approach and techniques for the geospatial NE disambiguation and resolution in Chapter 6. The first section is an introduction and a summary of current research. We then give an overview of Google Geocoder, and how it was applied to our research, and cover our data structure and disambiguation algorithm. We present the overall results of the NE extraction and disambiguation
process, and a discussion of the results and challenges.

Chapter 7 covers our work in geospatial relation extraction. The first section summarizes current research in relation extraction, with an overview of our objective. We then present the features selected for the SVM, and the corpus generation and SVM training. We conclude with the results of our testing, and a discussion of the results and challenges.

Finally, Chapter 8 presents a summary of and conclusions from our research. We discuss known challenges and issues with our research and the Geografikos package, followed by an evaluation of our research in terms of the success criteria specified in this chapter. We conclude with our contributions and possible future work stemming from the research in this thesis. Chapter 8 is followed by our research bibliography and appendices.
2 Named Entity Recognition

2.1 Overview

Named entity recognition refers to the extraction of words and strings of text within documents that represent discrete concepts, such as names and locations. The term “Named Entity” describes the operations in natural language processing (NLP) that seek to extract the names of persons, organizations, locations, other proper nouns, and numeric terms, such as percentages and dollar amounts. This use of the term “Named Entity” was defined by the Message Understanding Conference 6, sponsored by DARPA in 1996 [31]. Throughout this paper, named entity will be abbreviated to NE, and NE recognition (NER) shall refer to the extraction and classification of named entities. The MUC-6 conference originally sought to extract and classify simply persons, organizations, locations, currency expressions, and percentages. Since then, the usage of the term NE has expanded to include the extraction and classification of most categories of proper nouns. The NE extraction work in this paper is focused on the extraction of only location and geospatial entities, but this section will cover the overall performance and approaches to NE extraction since the original formalization of the NE task in MUC-6. The NE recognition task was further defined, and expanded to language independence by the Conference on Natural Language Learning shared task for the CoNLL 2002 conference [57]. As shorthand, the seminal categories of NEs are most often referred to as PER, LOC and ORG, for person, location and organization, respectively.

As an example of NER, consider the sentence:
Bill Gates is the founder of Microsoft, which is headquartered in Redmond, WA.

After processing through an NER system, the named entities in the sentence should be extracted and correctly identified. Bill Gates should be identified as PER, Microsoft as ORG, and Redmond, WA as LOC, which may be represented by marking up the sentence itself:

(Bill Gates)_PER is the founder of (Microsoft)_ORG, which is headquartered in (Redmond, WA)_LOC.

Named Entity extraction is generally divided into two stages. In some processing, the stages are combined, but can still be thought of as separate operations. The first stage of NE extraction is the identification of the NEs, based on word structure, context, and other features, depending on the method in use.

The second stage of NE extraction is the classification of each NE as a LOC, ORG, PER, numeric expression, or another category, depending on the NE categories under consideration. Generally speaking, this is the semantic understanding of the NEs that are extracted from text. Classification can also be performed in a number of ways, often combined with the extraction of the NEs. These methods will be summarized shortly.

Depending on the features that are used, a NE recognition system can be either language-dependent, or language-independent. A language independent approach to NE recognition is a system that can be trained using a dataset from any language, and provide good performance. The CoNLL 2002 and 2003 shared tasks provide
approaches and results on language independent tasks, in [57] and [58].

2.2 Performance Measures

MUC-6 defines three measures for the performance of a NE extraction system on a document, precision, recall, and f-measure [31]. Precision is defined as the number of correctly identified and classified NEs, scaled against the total number of retrieved NEs. Recall is defined as the number of correctly identified and classified NEs, scaled against the total number of NEs that should be identified in the document. F-Measure is an overall performance measure, a ratio of precision and recall. F-Measure is defined in terms of a weighting factor $\beta$ on the precision. For most uses, including that in this paper, $\beta$ is set to 1, so that precision and recall in are weighted equally.

We use the following notation:

- $N_{total}$ is the complete set of possible NEs in a document.
- $N_{retrieved}$ is the complete set of NEs that were identified and retrieved from a document by some NE extraction process, both correct and incorrect.
- $N_{correct}$ is the subset of the retrieved NEs that are correct, that is $N_{correct} \subset N_{retrieved}$ and $N_{correct} \subset N_{total}$.

\[
\text{precision} = \frac{N_{correct}}{N_{retrieved}}
\]

\[
\text{recall} = \frac{N_{correct}}{N_{total}}
\]

\[
\text{f-measure (}\beta=1\text{)} = \frac{(\beta^2 + 1) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}
\]
These measures are used throughout this paper to define the performance of the geospatial NE recognition, the disambiguation and resolution of the extracted entities, and the detection of geospatial relations.

2.3 Features for Named Entity Recognition

Named entity recognition is based on any number of sentence, word, syllable, and character features within the text of interest. Table 1 shows a comprehensive list of features used for NE recognition (list and shorthand representation extracted from [32]).

Depending on the method employed to perform the NE extraction, different subsets of these features are selected.

2.4 Approaches and Current Related Research

Numerous methods for NE extraction have been tried since the original formalization in the MUC-6, including Hidden Markov models, Conditional Random Fields, and Support Vector Machines. All NE methods fall into three general areas, defined by Mansouri, et al. in [46]. NER systems can be based on a rule or template set, statistical machine-learning techniques, or a hybrid approach.

Rule-based systems, are those systems that use hand-generated rules that match named entities. These may include regular expressions, templates, and other boolean rules that match against the text. These systems have been shown to work well in specific problem domains, but tend not to be portable to other problem domains,
Table 1: Features used for Named Entity Recognition

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ort</td>
<td>orthographic information, the set of symbols used, and the rules about how to write these symbols to form words and sentences in a language</td>
</tr>
<tr>
<td>lex</td>
<td>lexical features beyond simple orthographic features</td>
</tr>
<tr>
<td>num</td>
<td>numeric features, numbers, and alphanumeric words and phrases</td>
</tr>
<tr>
<td>aff</td>
<td>affix information, suffixes and prefixes of words, n-character strings</td>
</tr>
<tr>
<td>ws</td>
<td>word shapes, the shape of a word (capitalized, number of capitals, hyphen-separated, alphanumeric content, etc.)</td>
</tr>
<tr>
<td>cs</td>
<td>global case information</td>
</tr>
<tr>
<td>wv</td>
<td>word variations</td>
</tr>
<tr>
<td>len</td>
<td>word length</td>
</tr>
<tr>
<td>pos</td>
<td>general part-of-speech tags</td>
</tr>
<tr>
<td>np</td>
<td>noun phrase tags</td>
</tr>
<tr>
<td>vp</td>
<td>verb phrase tags</td>
</tr>
<tr>
<td>syn</td>
<td>syntactic tags, from an external system</td>
</tr>
<tr>
<td>tri</td>
<td>word triggers</td>
</tr>
<tr>
<td>hyp</td>
<td>words and phrases linked by hyphens</td>
</tr>
<tr>
<td>quo</td>
<td>word or words appearing between quotes, and quotes before or after a word</td>
</tr>
<tr>
<td>par</td>
<td>parentheses handling information, words and phrases before or after parentheses</td>
</tr>
<tr>
<td>pun</td>
<td>other punctuation markers surrounding words and phrases</td>
</tr>
<tr>
<td>ab</td>
<td>abbreviations of words or phrases</td>
</tr>
<tr>
<td>cas</td>
<td>cascaded entities</td>
</tr>
<tr>
<td>doc</td>
<td>global document information, including markup, context, and tagging</td>
</tr>
<tr>
<td>pre</td>
<td>previously predicted entity tags</td>
</tr>
<tr>
<td>bag</td>
<td>bag of words, treating the entire document as a single vector of words, with no consideration of punctuation</td>
</tr>
<tr>
<td>chu</td>
<td>chunk tags, tags on separate phrases within the sentence</td>
</tr>
<tr>
<td>cch</td>
<td>character-level/syllable level chunking and tagging within individual words</td>
</tr>
<tr>
<td>ext</td>
<td>External resources, such as WordNET, Wikipedia, the Penn Treebank corpus, and similar</td>
</tr>
<tr>
<td>gaz</td>
<td>gazetteers (reference lists of known named entities)</td>
</tr>
</tbody>
</table>
because the rules have to be re-written.

Machine-learning based approaches include those approaches that learn patterns or functions identifying the named entities by processing a corpus of pre-tagged examples based on some set of features. These methods have been studied in depth, and have been shown to provide excellent performance.

Finally, hybrid NE recognition systems combine both rule-based and machine-learning approaches, using the best aspects of both to perform the NE recognition. Any number of single processes can be combined together to form an NER system by using a weighting system on the output of each individual process to determine the overall results.

More specifically, the following approaches have been demonstrated with some success. Each bullet describes an approach, and the results of the most recent or best implementation of that approach.

- **Decision Trees**: Black and Vasilakopoulos, in [5], demonstrated the use of decision trees to classify named entities, achieving an f-measure of 0.62 on the Spanish CoNLL 2002 test dataset, slightly lower on the Dutch test dataset. Decision trees, as applied here, learn a series of rules and a lexicon by induction from a previously tagged examples.
- **AdaBoost**: In 2003, Carreras et al. applied AdaBoost binary classifiers to the NE classification task, demonstrating an f-measure of 0.79 on the Spanish test dataset, and 0.77 on the Dutch test dataset from the 2003 CoNLL shared task [11]. AdaBoost is short for Adaptive Boosting, a technique that applies a weighting technique to a series of training examples for any classifier (Car-
rreras et al. used a set of fixed-depth decision trees. After each learning pass, the AdaBoost processor increases the weights on training examples that were incorrectly classified, and decreases the weights on correctly classified training examples. Those examples with increased weights are concentrated on more heavily in future training passes.

- *Hidden Markov Models (HMM)*: HMM is a technique in which the system being modeled is assumed to be a Markov process with unobserved state transitions. For NER, a string of text is treated as the output of a series of state transitions, where the objective of the training process is to learn the state transitions that produce a certain string of words or characters, effectively answering the question “if I’ve seen these words, what word most likely comes next?” Zhou and Su, in [75], demonstrated word-level HMM for NER, using various combinations of four word sub-features, demonstrating an f-measure of 0.966 on the MUC-6 dataset, and 0.941 on the MUC-7 dataset, extracting PER, ORG, and LOC entities. Klein et al. performed NER using HMM at a character level instead of the word level, demonstrating an f-measure of 0.8607 on the CoNLL 2003 dataset [44].

- *Conditional Random Fields*: Krishnan and Manning demonstrated the use of CRF to exploit non-local dependencies to enhance NER, demonstrating an f-measure of 0.8980 on the extraction of LOC entities from the 2003 CoNLL dataset. A CRF is a generalization of an HMM that makes the state transitions into arbitrary functions that vary across the positions in the sequence of hidden states (in this case, words in a sentence), depending on the input sequence.
More simply, a CRF is an HMM that may (and is more likely) to make random state transitions.

- **Maximum Entropy (ME) Model**: Dingare et al. demonstrated an f-measure of 0.832 in extracting NEs from the biomedical BioCreative corpus. They employed an ME model, which essentially uses a logistic regression model to classify each word, overlaid with a Viterbi-style algorithm to find the best sequence of classifications [25]. These models have the ability to incorporate a large number of overlapping features.

- **Neural Networks**: A neural network consists of an interconnected group of artificial neurons and processes information using weighted connections between the neurons. In most cases a neural network is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Neural networks are often used to learn patterns, such as those for NE recognition. Hammerton demonstrated an f-measure of 0.60 using a recurrent neural network on the CoNLL 2003 dataset [34].

- **Support Vector Machines**: As NER using SVM is the focus of this research, the reasons for selecting SVM, and details on NER using SVM will be covered in more depth in the next chapter.

Combination approaches have also been applied to the problem of NER. Table 2 (adapted from [32]) shows a breakdown of the approaches and results from all the participants in the CoNLL 2003 shared task.

As demonstrated in the table, a wide range of approaches have been applied to the task of NE recognition and classification, demonstrating various levels of perfor-
<table>
<thead>
<tr>
<th>Approach/Citation</th>
<th>Methods</th>
<th>English F-Measure</th>
<th>German F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Florian et al. [28]</td>
<td>MEMM + HMM + RRM + TRAN</td>
<td>88.76</td>
<td>72.41</td>
</tr>
<tr>
<td>Chieu and Ng [14]</td>
<td>MEMM</td>
<td>88.31</td>
<td>65.67</td>
</tr>
<tr>
<td>Klein et al. [44]</td>
<td>MEMM + HMM + CMM</td>
<td>86.07</td>
<td>71.90</td>
</tr>
<tr>
<td>Zhang and Johnson [74]</td>
<td>RRM</td>
<td>85.50</td>
<td>71.27</td>
</tr>
<tr>
<td>Carreras et al. [11]</td>
<td>ADA</td>
<td>85.00</td>
<td>69.15</td>
</tr>
<tr>
<td>Curran and Clark [21]</td>
<td>MEMM</td>
<td>84.89</td>
<td>68.41</td>
</tr>
<tr>
<td>Mayfield et al. [49]</td>
<td>SVM + HMM</td>
<td>84.67</td>
<td>69.96</td>
</tr>
<tr>
<td>Carreras et al. [12]</td>
<td>PER</td>
<td>84.30</td>
<td>66.48</td>
</tr>
<tr>
<td>McCallum and Li [50]</td>
<td>CRF</td>
<td>84.04</td>
<td>68.11</td>
</tr>
<tr>
<td>Bender et al. [2]</td>
<td>MEMM</td>
<td>83.92</td>
<td>68.88</td>
</tr>
<tr>
<td>Munro et al. [52]</td>
<td>Voting + Bagging</td>
<td>82.50</td>
<td>67.75</td>
</tr>
<tr>
<td>Wu et al. [72]</td>
<td>ADA (3 stacked learners)</td>
<td>81.70</td>
<td>66.34</td>
</tr>
<tr>
<td>Whitelaw and Patrick [68]</td>
<td>HMM</td>
<td>79.78</td>
<td>54.43</td>
</tr>
<tr>
<td>Hendrickx and Bosch [35]</td>
<td>MEM</td>
<td>78.20</td>
<td>63.02</td>
</tr>
<tr>
<td>De Meulder and Daelemans [23]</td>
<td>MEM</td>
<td>76.97</td>
<td>57.27</td>
</tr>
<tr>
<td>Hammerton [34]</td>
<td>RNN</td>
<td>60.15</td>
<td>47.74</td>
</tr>
<tr>
<td>Baseline [58]</td>
<td></td>
<td>59.61</td>
<td>30.30</td>
</tr>
</tbody>
</table>

mance, with f-measures as high as 0.9439 on LOC entities. The next section discusses the challenges in NER that necessitate such a wide range of approaches.

2.5 Challenges in Named Entity Recognition

Despite the good results (greater than 80% f-measure) demonstrated by the NE classification and recognition systems and approaches enumerated above, NER systems still face significant challenges. These challenges generally fall into one of three areas, training data acquisition, domain and language dependence, and performance.

The majority of NER systems in use today require manually annotated training data for initial model training. A few approaches have automatically acquired training data, but these approaches tend to be domain specific, and do not perform as well as systems trained with manually annotated training data. Manually tagging training data is a long and error prone process. Differences in opinion between the persons doing the tagging can result in inconsistent tagging, and when working with large datasets, it can be easy to make errors. Furthermore, each time NER is performed in a new domain or language, a new corpus of data must be tagged before the actual NER work can begin. The amount of textual data available in digital form expands at an exponential rate compared to the amount of available data that is annotated for NER.

In addition to the data problem, NER systems are not generally applicable to more than one domain or language. Even the results of the CoNLL 2002 and 2003 tasks which specifically targeted language independent NER show this, as the perfor-
mance of the primary language is some 10-15% better than the secondary language used in each task. NER systems perform demonstrably better when provided with domain specific knowledge, but the application of that knowledge inherently includes a trade-off with general applicability. Each time an NER system is used in a new language or domain, the features used must be changed, requiring the coding of new feature extractors. Integral to the extraction of new features in a domain or language is the question of which features to use, to provide the best performance.

With the explosion in unstructured textual information, NER systems must perform their task rapidly, and be highly scalable to be effective. An NER system must perform well at all stages, from the feature extraction, through the NER process, and in any post-processing. The approaches presented in the previous section are generally either processor (time) or memory (space) intensive, often both. Time and space tend to trade off in NER approaches, but both must be minimized to achieve good scalability and throughput, as the performance of any NER system directly dictates the usefulness.

A related challenge in NER is the overall evaluation of the effectiveness of an NER system. While the performance measures described earlier in this chapter are applied to the output of a system given a specific set of test data, there is no consistent way to gauge the performance of a system on the NER task as a whole. Each system must be tested in each domain, on a known set of data. Furthermore, the highest performing NER systems are built out of a series of different NER components (see Table 2), and the performance of each component cannot be easily evaluated separately. The system must be treated as a black box, without knowing the performance
contributions from each component. The performance of each component can be estimated by measuring the performance of the system as a whole as components are isolated, but this is error prone, due to the unknown interactions between each component.

Undoubtedly, NER systems face a daunting set of challenges. However, SVMs have stood up well to all of these challenges, presenting an excellent approach to NER, though not without their own challenges. Chapter 3 discusses the theory behind SVM, challenges in SVM, and their application to NE problems.

2.6 Named Entity Recognition Applied to Wikipedia

NE classification and recognition techniques have been applied to Wikipedia in the past, with good results. In [22], Dakka and Cucerzan use NE techniques to sort Wikipedia pages into the NE categories defined by the CoNLL 2002 task, one of the first known attempts to perform this kind of automatic classification on the English Wikipedia. They demonstrated techniques to classify an entire article page with a single NE category, achieving an overall f-measure of 0.884 across all NE classes, and an f-measure as high as 0.954 on LOC pages.

Cucerzan also presented a technique to do large-scale NE disambiguation by using Wikipedia data, in [18]. Through a process of maximizing the agreement between the contextual information extracted from Wikipedia and the context of a document, as well as the agreement among the category tags associated with the candidate entities, his system showed high disambiguation accuracy on both news stories and
Wikipedia articles.

Addressing the challenge of hand-tagging large amounts of data for NER, Nothman, Curran, and Murphy applied NE techniques to Wikipedia to automatically generate a large corpus of annotated data for the training of NE systems. They transformed Wikipedia links into NE annotations by classifying the target page of the link as a PER, LOC, ORG, or other NE category [54].

These and other ongoing work demonstrate that Wikipedia both provides a good source for NE data, and is an area of intensive ongoing research into NE techniques and uses.

2.7 Location-Specific Named Entity Resolution

While we have covered general NE approaches, techniques, and results in this chapter, this research is specifically focused on the extraction of geospatial NEs, or place names. We further refine our task definition to not only the recognition of the geospatial NEs in the Wikipedia article text, but also the geocoding of each to create a geospatial entity. After processing completion, we wish to have a list of geospatial entities, each with an associated \((\phi, \lambda)\) coordinate pair, so that each LOC entity is associated with a single place on the Earth, in the context of the article. That is, if we see Cambridge in the article, though there are a number of possible correct locations for Cambridge, we determine the correct spatial location based on the text of the article.

If the recognition of a LOC NE is analogous to identifying an object as a car,
the disambiguation and geocoding task we set forth in the rest of this paper is analogous to identifying the make and model of the car. This task, therefore, presents unique challenges. The primary language processing challenge is presented in the determination of entity boundaries. Many LOC entities are actually a combination of place names, such as CITY, STATE and CITY, COUNTRY. Each piece of the place name can stand alone as a place name, but must be resolved as a single string to be correctly geocoded to a coordinate pair. Resolving the coordinates for the place name correctly will also depend on the entity context, leading to the challenge of identifying location context with non-local dependencies. Non-local dependencies are contextually related location entities that are separated by other words. For instance, in the phrase “Monterey, in the state of California”, Monterey is a city in California, but the recognition that Monterey and California are related entities provides a significant challenge.

In this research, we work at a scale such that we resolve each place name to a single coordinate pair, treating everything as a point. Though at a close enough scale, cities, states and countries should be actually treated spatially as polygons, for the purposes of this research, we treat them as points. Somewhat in conflict with correct geographic procedure, we also treat other linear geographic features, such as rivers as points. This is somewhat incorrect. However, There are many databases that contain the correct geometric information for cities, states, countries, rivers, and other geospatial entities, such as the TIGER line data for the United States\(^3\). After the second phase of the research, we have geospatial entities, with “fully-qualified”

\(^3\)http://www.census.gov/geo/www/tiger/
place name attributes. That is, the place names are non-ambiguous, referring only
to a single possible spatial location. Therefore, we could potentially use the place
name of geospatial entity generated by the Geografikos system to look up the correct
genometry (line or polygon) from a separate database.

These challenges and more will be addressed in the next chapters as we discuss
our approach to the NE recognition, disambiguation, and resolution, and our results.
3 Support Vector Machines

3.1 Overview and Theory

Support Vector Machines (SVM or SVMs) are a subset of kernel-based methods used for regression and classification. An SVM views input data as a series of feature vectors in some $N$-dimensional space, with the goal of finding an optimal hyperplane that separates the feature vectors into positive and negative classes, based on a function learned from a set of training vectors. SVMs are a supervised learning method, meaning that the function for separating the feature vectors is learned from a set of previously tagged feature vectors.

The statistical learning theory behind SVMs is based on Vapnik’s work as far back as the 1970s. He formally introduced SVM theory in his books in 1995 and 1998, [16] and [64]. SVM theory in its current form was further defined in Vapnik’s book “The Nature of Statistical Learning Theory”, [65]. Both theoretical and experimental results indicate that SVMs are able to generalize very well and avoid overfitting in high (and even infinite) dimensional feature spaces.

SVMs are inherently discriminative, that is, instead of learning to classify objects based on the objects themselves, the SVM simply learns that things with this feature vector are in class A, and things with that feature vector are in class B, discriminating between two classes. This flexibility allows them to be applied to a wide range of problem domains. SVMs also perform well in feature spaces where the features may overlap, as they do in the NER problem domain. Additionally, SVMs have a high generalization capability, and are able to operate in features spaces of
high dimension, up to and including 100,000s of features [33].

SVMs were developed for many reasons, including the bias-variance tradeoff, capacity control, and overfitting [10]. The bias-variance tradeoff is the problem of training a learning system to recognize the frequent examples well, but also to recognize and process infrequent examples and examples that do not appear in the training set at all. Capacity control and overfitting are training problems where the learning system learns only the training data very well, but the learned function does not apply well to general examples. In an SVM, overfitting is prevented because the SVM maximizes the distance between the training class boundary (the support vector) and the nearest instances. [20]

An ideal solution for the deterministic problem of NER will be both accurate, and generalized. Given a specific training set, the program should have a high accuracy on that training set, while still maintaining the ability to accurately learn any other training set. Furthermore, once learning a classification function from the training set, that function should generalize well to any test data with the same parameters. Research into text classification and NER has found that SVMs provide good performance on NLP tasks despite the inherent difficulties of NLP [42], and perform very well for NER [22,33,46,63].

SVMs can be used in both binary and multi-class forms. In the binary form, which we used for this research, the SVM learns a function that divides any input feature vectors into two classes based on the learned hyperplane. A multi-class SVM learns a more complex, possibly non-linear hyperplane, and can divide the input feature vectors into multiple different classes. We used a binary SVM for this research
because the problem of NER is simpler when formulated as a binary problem. For each word in an article, we generate a feature vector. Each of those feature vectors either does or does not represent a geospatial NE. This binary formulation of our NER problem also makes the SVM training easier, because the binary SVM is easier to train than the multi-class SVM. For a more in-depth discussion on multi-class SVMs, see Chapter 3 in [32].

From an operational standpoint, a binary SVM seeks to learn a function that divides the set of training vectors into positive and negative sets. The SVM does this by learning a hyperplane that divides the positive and negative examples. Figure 3\textsuperscript{4} shows one such hyperplane.

Because any given set of training data may be separated by more than one possible hyperplane, the SVM seeks to maximize the width of the hyperplane, called the margin. The two vectors that define the edges of the hyperplane are referred to

\textsuperscript{4}Image courtesy of Suzette Stoutenburg
as the support vectors. Consider the following formalization, adopted from [10,38].

The training data for an SVM is given in the format

\[
\{(x_1,y_1), \ldots, (x_i,y_i) \mid x_i \in \mathbb{R}^n, y_i \in \{-1, +1\}\}_{i=1}^{n}
\]  

(4)

where \(x_i\) is a vector composed of the features of the \(i\)–th training sample, and \(y_i\) is a binary class label, either +1 or -1. The SVM will separate the examples into positive and negative classes with a hyperplane defined by

\[
wx + b = 0 \mid w \in \mathbb{R}^n, b \in \mathbb{R}
\]  

(5)

Figure 3 shows one resulting hyperplane. Depending on the features used for each training example, there may not be a unique hyperplane that separates the two classes. Therefore, the SVM finds the optimal hyperplane by maximizing the margin, which is the distance between the support vectors. The width of the margin is simply \(2/\|w\|\). In Figure 3, the solid lines passing through the positive and negative examples are referred to as the support vectors, and the margin is the distance between the support vectors. The dotted line is the separating hyperplane. Figure 3 shows a linearly-separable set of data, where the hyperplane completely divides the dataset into positive and negative examples, with no overlap.

Unfortunately, in general practice, the dataset will not be linearly separable. In this case, the SVM will attempt to separate the training examples in such a way that the margin, on average, is as large as possible. Mathematically, this is done by introducing a slack variable \(\zeta_i\) to the support vector equations, for each \(x_i\) in the
training set.

\[wx + b \geq 1 - \zeta_i\]  \hspace{1cm} (6)

\[wx + b \leq -1 + \zeta_i\]  \hspace{1cm} (7)

Figure 4\textsuperscript{5} shows the optimal hyperplane for a non-separable dataset. For any single feature vector in the training set, a \(\zeta_i\) value greater than zero indicates that the position of that vector is within the margin, and a \(\zeta_i\) greater than 1 indicates that the position of the vector is beyond the hyperplane, in the space of the opposite class.

Given the \(\zeta_i\) slack variables for error, the objective SVM is to minimize the following objective function

\[
\frac{1}{2} ||w||^2 + \left(C \sum_{i=1}^{n} \zeta_i \right)^k
\]  \hspace{1cm} (8)

By solving a quadratic programming problem in the case that \(k=1\), the decision

\textsuperscript{5}Image courtesy of Suzette Stoutenburg
A function $f(x) = \text{sgn}(g(x))$ can be derived, where

$$g(x) = \left( \sum_{i=1}^{n} \lambda_i y_i x_i \cdot x + b \right)$$

(9)

The decision function depends only on the support vectors $x_i$, so training examples that do not define the support vectors have no influence on the decision function.

In the case that we have non-linear data, the function can easily be adjusted. The data in the decision function is only represented in the form of the dot product $x_i \cdot x$. Therefore, prior to training, the data can be mapped to another n-dimensional Euclidean space $\mathcal{H}$ using a mapping $\Phi$, where the Euclidean space $\mathcal{H}$ allows the data to be separated by a linear hyperplane (or any other geometric hyperplane form we desire).

$$\Phi : \mathbb{R}^n \rightarrow \mathcal{H}$$

(10)

The training algorithm in this new space will only depend on the existing training data through the dot product $x_i \cdot x$. Therefore, there exists a kernel function $K$ that maps the training data from the original space to the new space, $\mathcal{H}$

$$K(x_i, x) = \Phi(x_i) \cdot \Phi(x)$$

(11)

Only the kernel function $K$ needs to be used in the training algorithm, and the data-mapping can be ignored by the training function [10]. The inner product in the original $g(x)$ function can be replaced with the kernel function, yielding the following equation

$$g(x) = \left( \sum_{i=1}^{n} \lambda_i y_i K(x_i, x) + b \right)$$

(12)

As an example, the radial basis kernel function has been shown to be very effective NLP and NER tasks (see [39]), so the SVMs for this research used this
function, referred to as the RBF.

\[ K(x, y) = e^{-\gamma \|x_i - x\|^2}, \gamma > 0 \]  \hspace{1cm} (13)

When using the RBF as the kernel function, the \( \gamma \) parameter in the RBF, and the \( C \) parameter in objective function (equation 8) must both be optimized. \( \gamma \) represents the degree of the kernel function, normally set to approximately \( 1/k \), where \( k \) is the number of features in each feature vector. \( C \) represents the cost of the error parameter \( \zeta_i \) in the objective function, indicating how much slack is allowed for the error of feature vectors in and across the hyperplane. The optimization of these parameters for each SVM used in this research will be discussed in depth in Chapter 5.

For this research, we chose the LibSVM implementation, version 2.87, from the National Technical University in Taiwan [13]. For further information on SVMs, Burges provides an excellent introduction to SVMs for pattern recognition in [10], and Hsu et al (the authors of LibSVM) provide an excellent classification tutorial in [39]. Habib also provides an excellent overview in [32]

### 3.2 Performance Measures

The performance measures used to determine the performance of an SVM are similar to the performance measures defined in Chapter 2 for NER. Precision, recall, and f-measure are used with the same definitions. However, precision, recall, and f-measure must be calculated separately for the positive and negative classes. To calculate the performance for the positive classification, we use the following notation:
• $N_{total}$ is the complete set of feature vectors.

• $N_{positive}$ is the complete set of NEs that were identified as feature vectors in the positive class, both correct and incorrect.

• $N_{correct}$ is the subset of the feature vectors correctly placed in the positive class.

\[ N_{correct} \subset N_{positive} \text{ and } N_{correct} \subset N_{total}. \]

\[ \text{precision} = \frac{N_{correct}}{N_{positive}} \quad (14) \]

\[ \text{recall} = \frac{N_{correct}}{N_{total}} \quad (15) \]

\[ f\text{-measure } (\beta=1) = \frac{(\beta^2 + 1) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}} \quad (16) \]

To calculate the performance for the negative classifications, simple replace all terms from the positive class in the above equation with the terms from the negative class. For the purposes of this research, we will not be directly calculating the performance of the SVM, but instead using the equations for precision, recall, and f-measure defined for NER to evaluate the system as a whole.

### 3.3 SVM Challenges

While SVMs do perform well in large feature spaces, and feature spaces with overlapping features, their use does present its own set of challenges. Four primary challenges present themselves.

First, the input feature vectors input to an SVM must be purely numeric features. This requires that all features be represented in terms of a continuous decimal
number, or a binary feature \( \{0,1\} \), though \( \{-1,1\}\) is also frequently used). For any problem domain, especially NLP, where most features are word, syllable, and character-based, a translation function must be chosen for each non-numeric text feature, to translate it to a numeric feature. For optimal performance, all the features fed to an SVM need to be scaled to the range \([-1..1]\). While each feature can be scaled individually, the translations from text need to take the scaling into account, and make sure that when the features are translated, the scaling does not introduce artificial weighting of one feature over another.

Additionally, training time can often be a big consideration in the use of SVM. For large sets of training vectors, and for vectors with a large number of features, training time can easily require continuous processing for days. This problem normally comes from broadly overlapping features, such that there is a large amount of error in the objective function. Choosing a strict error cost parameter \( C \) can also lead to significant increases in training time. Training data must also be selected carefully. If a number of the feature vectors in different classes are very close in representation, the SVM will take a significant amount of time to separate them. This problem was encountered during this research, in the training of the SVM for NER.

Despite the inherent resistance of SVM to overfitting to the training data, overfitting can occur if the training data is not balanced, or if the features are not carefully chosen. While SVMs can perform well with few training vectors, the number of positive examples must be proportional to the number of negative training vectors, or the SVM can be biased towards one class or the other. To offset this challenge, most SVM implementations currently available allow the weighting of one class or the other.
to account for unbalanced training examples. A second cause of overfitting can be incorrect selection of features to represent the problem domain. If the features model the training data too heavily, instead of modeling the problem domain in general, the SVM will overfit to the training data. This can be mitigated through careful feature selection, and through cross-validation in the training of the SVM. In cross-validation, the training data is randomly separated into training and test sets in a roughly 75%/25% distribution. The 75% is used to train the SVM, and the 25% is used to test the SVM. This random split and training procedure is repeated 10 or more times, and the results compared.

Finally, the $C$ and $\gamma$ parameters of the objective and kernel functions must be optimized. If these are not chosen carefully, the error margin can be wide enough to either degrade classification performance, or to overfit the SVM to the training data. Normally, for a new SVM training procedure, significant time must be taken to perform a search over a wide range of value pairs for these parameters for find the optimal values for the problem domain and features under consideration.

Despite these challenges, however, SVMs are a promising technology for a wide range of machine learning approaches including natural language processing, justifying their selection for this research.

### 3.4 Named Entity Recognition using SVM

SVMs have shown significant promise for the task of NER. Starting with the CoNLL 2002 shared task on NER, SVMs have been applied to this problem domain
with improving performance. Bhole et al have done significant work in extracting named entities from Wikipedia and relating them over time using SVMs [3]. In the biomedical domain, Takeuchi and Collier have demonstrated that SVMs outperform Hidden Markov Models (HMM) for named entity recognition [63]. Furthermore, Lee et al have shown that SVMs are applicable to multi-class NER and text classification problems [45]. Kazama et al demonstrated in [43] that applying SVMs against parts of speech instead of clear text results in a performance gain in NER. Isozaki and Kazawa provide results in [41] showing that chunking and part-of-speech tagging also provide performance increases for NER with SVMs, both in speed and in f-measure. These papers demonstrate that SVMs are particularly suited to our task of geospatial NE (place name) extraction. In [22], Dakka and Cucerzan demonstrated an SVM that achieved an f-measure of 0.954 for LOC entities in Wikipedia articles, and an f-measure of 0.884 across all NE classes. Using an SVM in conjunction with a fuzzy SVM which allowed greater margin overlap in the cost parameter, Mansouri, Affendey, and Mamat demonstrated excellent performance on NER [47].

This research demonstrates that SVMs are a good choice for NER, as the benefits outweigh the challenges. However, it still presents its own set of challenges, partially described in the previous section. Primarily, the text features of the words must be translated to numeric features for processing by the SVM. Each feature extractor must be individually coded. The set of training data for the SVM must be generated and tagged by hand. Our approaches to these challenges, and the techniques and features used to perform the geospatial NE recognition using an SVM are described in Chapter 5. Our further research using an SVM to extract the open
geospatial relations is covered in Chapter 7.
4 Wikipedia Background and Corpus Preparation

4.1 Wikipedia Overview

Wikipedia is one of the foremost repositories of human knowledge currently on the Internet, and represents an amazing amount of human knowledge and judgement. As Wikipedia is best described by its own content, the Wikipedia:About page\(^6\) reads:

> Wikipedia is a multilingual, Web-based, free-content encyclopedia project. …Wikipedia’s articles provide links to guide the user to related pages with additional information.

Wikipedia is written collaboratively by volunteers from all around the world; anyone can edit it. Since its creation in 2001, Wikipedia has grown rapidly into one of the largest reference Web sites, attracting at least 684 million visitors yearly by 2008. There are more than 75,000 active contributors working on more than 10,000,000 articles in more than 260 languages. As of today, there are 2,820,870 articles in English. Every day, hundreds of thousands of visitors from around the world collectively make tens of thousands of edits and create thousands of new articles to augment the knowledge held by the Wikipedia encyclopedia.

Wikipedia is one of the best examples on the Internet of the “wisdom of the crowds”. Because it allows anyone to edit any article on the site at any time, it becomes very easy to make updates and changes to Wikipedia pages. While this

can occasionally lead to bias, and in some cases, vandalism of articles, Wikipedia is well-tended by users, which mitigates these problems. This has resulted in a huge collection of articles on a wide variety topics with up-to-date and generally correct information. For highly technical or obscure information, there is limited peer review, but for most uses, Wikipedia provides an excellent corpus of information.

Wikipedia is structured as a series of articles, or pages, and contains a number of different types of pages:

- **Article Pages**: These are the actual content pages in Wikipedia. Each is a self-contained page on a single topic, such as the page on World War II\(^7\)
- **Redirect Pages**: These pages simply redirect from one page title to another. For example, World War 2 can be referred to by any number of titles, WW2, World War II, and World War 2. For each subject, Wikipedia only provides a single, actual page, in this case, “World War II”. For all other titles referring to the same page, Wikipedia provides a redirect page, which is a simple pointer to the actual article page.
- **Disambiguation Pages**: For titles that could refer to a number of different topics, with different pages, Wikipedia provides disambiguation pages, which are simply a list of all the pages that could refer to that title. For instance, the disambiguation page on “Washington”\(^8\) lists all the possible pages that refer to Washington, including the person George Washington and the place Washington state.

---

\(^7\)http://en.wikipedia.org/wiki/World_War_II

\(^8\)http://en.wikipedia.org/wiki/Washington_(disambiguation)
• **List pages**: These pages provide logical groupings of other articles in Wikipedia, such as a list of battles in a war\(^9\), or a list of US presidents\(^{10}\).

• **Special pages**: These are pages that include media pages, with image uploads, movies, etc., and other kinds of configuration and special pages.

Along with the ease of creating and editing the page types enumerated above, the real strength in Wikipedia lies in the links between the pages. Wikipedia makes it easy to link the pages together, creating a graph between articles, by title. Links in Wikipedia text are created by simply surrounding the title of another page with brackets: `[[Page_Title]]`. This link information makes it easy to follow from one article to another for more information, by topic and title.

#### 4.2 Research Application of Wikipedia

Wikipedia was chosen as the focus of this research for a number of reasons. First, it provides a large corpus of unstructured text in a format that is easy to process, search, and use for research work. We treat the Wikipedia database as unstructured text for the purposes of this research, but the database does contain all the link and structure information inherent in Wikipedia. Second, research has indicated that, because Wikipedia articles are edited continuously by numerous collaborators, their content is believed to have high grammatical correctness compared to the web overall [53]. This grammatical correctness in the text makes the preprocessing for parts of speech and phrases much easier and less error-prone. Additionally, because Wikipedia

\(^9\)http://en.wikipedia.org/wiki/List_of_American_Civil_War_battles

\(^{10}\)http://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States
itself is web-based, simple REST\textsuperscript{11} queries can be performed against various articles. While the page content for each article is stored locally in the Geografikos MySQL database, the existing Wikipedia web site is used to perform some tasks, including article disambiguation and page ID recognition.

The Wikimedia Foundation, the non-profit organization behind Wikipedia, provides a download of the complete page text of Wikipedia. The archive corresponding to each language which has a Wikipedia is available for download. For this research, we downloaded the English language pages and links database from the June 18, 2008 dump of Wikipedia\textsuperscript{12}. This download provides the full text of all the pages in the English Wikipedia at the time that the download package is created. This full text includes all wiki and HTML markup. The June 18, 2008 download contains approximately 7.2 million pages.

The links database is a MySQL database table dump file with all the links between all of the 7.2 million articles in the dump. The links table contains the approximately 260 million links. The Wikimedia Foundation does not guarantee that all links work, but due to the efforts of the Wikipedia editors, an average of 95\% of the links in each page are correct, and point to existing pages within Wikipedia.

Once the article text was parsed and the wiki and HTML markup information was stripped out, the articles provided an easily-used corpus of free text. Each page was tagged with an individual ID in the database. A simple REST interface to the

\textsuperscript{11}REpresentational State Transfer: the use of GET, POST, PUT and DELETE HTTP requests to retrieve and work with resources on the Internet.

\textsuperscript{12}\url{http://download.wikimedia.org/enwiki/20080618/}
Wikipedia website for search allowed resolution of a page ID, and the full text of the page could then be loaded from the database for processing.

4.3 Database Preparation

The articles database and the links database are provided as two separate downloads, in a compressed format to save bandwidth, as the two combine to about 5 GB compressed. The links database is a single text file, a MySQL formatted table dump file. After extraction, it was simply loaded directly in the the database for Geografikos.

The articles download package, however, is more complex. The articles are provided as a single 16 GB XML file that contains the page, revision, and text information for each article. Figure 5 shows the structure of the database that describes each page, which is inherent in the XML file, but requires processing before loading into the database.

Each article has a separate record in the Pages table, with an ID, the title of the page, and various other metadata about the page, including a boolean flag that indicates if this is a redirect page. The Texts table is simply a series of IDs and text contents. Each page in Wikipedia can be revised many times by many different
Figure 6: Partial Listing of the XML Export for the World War II Wikipedia Article

editors, so the actual text of the page must be separated from the page information. This is done through the Revisions table. Each record in the Revisions table links a page to a text, with a date or revision and other assorted metadata. To render an actual article page with a certain title, the most recent text for that page is found in the texts table by looking up the page.id in the revisions table.

The single XML file from Wikipedia has the contents of each article in XML, for example Figure 6 shows a partial export of the World War II article page.

For each page in Wikipedia, the single most recent revision is provided, with the text of that revision. The export files do not provide any history on the pages or past revisions, but this information was not required for this research.

To use this information in this research, the XML was parsed out into three separate MySQL dump files, one each for the pages, revisions, and texts tables.
Fortunately, the Wikimedia Foundation provides a tool, called xml2sql\(^{13}\). The xml2sql tool parses the XML file and creates the three separate dump files. These dump files were then loaded directly in the Geografikos MySQL database. With the existing links information, this resulted in a 60GB database with all the pages from the English Wikipedia as of June 18, 2008.

Once the parsing and preprocessing was complete and the database was loaded, we ran into our initial problem, that querying the database and returning a single page required some 60 seconds. Worse yet, querying for a full-text match on a title required approximately 3 minutes, obviously demonstrating the need for optimization, mostly due to the sheer size of the databases. To do that, we took the following steps:

1. We changed the underlying MySQL table engine for the Texts table to the MyISAM engine. By default, tables in MySQL are created with the InnoDB engine. For most applications, the InnoDB engine is more optimal, and provides much better performance. However, for tables that store a significant amount of text and require full-text searching, the MyISAM storage engine provides better performance, with smaller disk space requirements for the storage and indexes.

2. We created an index on the Pages table, on the ID column. In MySQL, an index is a stored data structure that provides rapid lookup into a table, based on the columns in the index. For this table, the index on the ID column provides a rapid lookup on the pages by the page ID.

3. We created an index on the ID and page_id column in the Revisions table.

\(^{13}\)http://meta.wikimedia.org/wiki/Xml2sql
4. We created an index on the ID column in the Texts table.

5. We created a full-text index on the titles column in the Pages table. A full-text index enables string-based queries to the table that will match all or part of the title of a page, including returning weightings on each word indicating the closeness of the match to the original query. A full text index on the Pages.title column allowed for text search with partial phrases.

6. By default, the MySQL engine creates full text indexes only on words of length four or more. For better performance in certain situations, this length can be reduced. Because we were interested in full-text searches on the titles of the article pages, the minimum length for indexing was reduced to three, which provided better title-only phrase searching.

7. Finally, because this research was interested only in full-text articles, all pages of the Special type were cleared from the database. Also, all the Image: and Media: type pages were cleared from the database, including the various graphics, documents, and other files. This reduced the number of pages in the database by approximately 500,000, and was done primarily to remove those page titles from the full-text index.

Once all these optimizations were complete, query time for individual page was reduced to approximately 1.5 milliseconds, a significant decrease in query time, allowing for many queries to the database in a very short time. Query time for full-text queries for pages by title was reduced to approximately 500 milliseconds.
4.4 Corpus Selection and Tagging

As previously stated, this research was focused on extracting geospatial NEs from free text and creating geospatial entities. The Wikipedia corpus was chosen because of easy accessibility and known-good content. Of course, with some 6 million pages still in the database after all the optimization and preparation, the scope needed to be narrowed further. For this research, we sought pages with a high incidence of geospatial location names. Once such set of pages are all the pages about historical wars and battles in Wikipedia. These pages tend to have a high count of place names.

A selection of 90 pages about battles and wars were selected from Wikipedia and hand-tagged to provide the testing set for the named entity extraction, and to provide training and testing for the relationship extraction portions of this research. The 90 pages resulted in approximately 230,000 words of testing data, an average of 7% of which are geospatial named entities. The pages also provided a total of 5000 sentences for training and testing, an average of 5% of which contain the geospatial relations we sought.

For the NE recognition task, the text of each page, scrubbed of the wiki and HTML markup, was tagged, word by word. For each word in each page, the word was tagged with either a 1 or a 0, indicating that it either was or was not part of a geospatial NE. For the relationship recognition task, the same text for each page, scrubbed of the markup, was used. However, the page was split into sentences instead of words, and each sentence was tagged with a 1 or 0, indicating that the sentence did or did not contain an open geospatial relation, that is, a relation between a place
name and another significant phrase in the sentence.

Table 3 contains a partial list of the pages selected for the corpus, and the metrics on word count and geospatial NE count in each page. The NE count in the table represents the total number of words that pertain to geospatial locations. The relation count is the total number of sentences that contain an open geospatial relation.

The location metrics in the corpus, the results of the NE extraction and resolution, and geospatial relation extraction will be discussed in more depth in the following chapters.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Battle of Antietam</td>
<td>6167</td>
<td>400</td>
<td>260</td>
<td>56</td>
</tr>
<tr>
<td>Battle of Britain</td>
<td>10611</td>
<td>512</td>
<td>229</td>
<td>41</td>
</tr>
<tr>
<td>Battle of the Bulge</td>
<td>7614</td>
<td>339</td>
<td>265</td>
<td>31</td>
</tr>
<tr>
<td>Battle of Chancellorsville</td>
<td>2979</td>
<td>137</td>
<td>133</td>
<td>14</td>
</tr>
<tr>
<td>Battle of Chantilly</td>
<td>1170</td>
<td>50</td>
<td>56</td>
<td>5</td>
</tr>
<tr>
<td>Battle of Chickamauga</td>
<td>2000</td>
<td>99</td>
<td>108</td>
<td>13</td>
</tr>
<tr>
<td>American Civil War</td>
<td>7961</td>
<td>397</td>
<td>298</td>
<td>22</td>
</tr>
<tr>
<td>First Barbary War</td>
<td>1906</td>
<td>88</td>
<td>97</td>
<td>8</td>
</tr>
<tr>
<td>Battle of Fredericksburg</td>
<td>2354</td>
<td>111</td>
<td>77</td>
<td>16</td>
</tr>
<tr>
<td>Battle of Gettysburg</td>
<td>5289</td>
<td>256</td>
<td>274</td>
<td>35</td>
</tr>
<tr>
<td>First Gulf War</td>
<td>11562</td>
<td>895</td>
<td>674</td>
<td>7</td>
</tr>
<tr>
<td>Korean War</td>
<td>10570</td>
<td>462</td>
<td>495</td>
<td>64</td>
</tr>
<tr>
<td>Mexican-American War</td>
<td>5822</td>
<td>275</td>
<td>472</td>
<td>63</td>
</tr>
<tr>
<td>Operation Eagle Claw</td>
<td>1869</td>
<td>60</td>
<td>56</td>
<td>11</td>
</tr>
<tr>
<td>Operation Nickel Grass</td>
<td>1092</td>
<td>52</td>
<td>47</td>
<td>8</td>
</tr>
<tr>
<td>Attack on Pearl Harbor</td>
<td>3414</td>
<td>188</td>
<td>198</td>
<td>24</td>
</tr>
<tr>
<td>Battle of Shiloh</td>
<td>4899</td>
<td>253</td>
<td>157</td>
<td>23</td>
</tr>
<tr>
<td>Spanish-American War</td>
<td>4229</td>
<td>194</td>
<td>296</td>
<td>33</td>
</tr>
<tr>
<td>Battle of Vicksburg</td>
<td>3369</td>
<td>178</td>
<td>119</td>
<td>24</td>
</tr>
<tr>
<td>The Whiskey Rebellion</td>
<td>1475</td>
<td>65</td>
<td>49</td>
<td>6</td>
</tr>
<tr>
<td>World War II</td>
<td>6775</td>
<td>331</td>
<td>661</td>
<td>69</td>
</tr>
</tbody>
</table>
5 Geospatial Named Entity Extraction

5.1 Overview and Objective

As discussed in the introduction to this thesis, the first task of this research was to extract all the candidate geospatial named entities (NE) from each Wikipedia article in the testing corpus. The desired output of this stage of the Geografikos package was a series of words and strings that may be place names. The resolution of this set of candidate NE strings each to a (latitude, longitude) coordinate pair is discussed in the next chapter.

This chapter discusses the use of a Support Vector Machine, as covered in Chapter 3, for this geospatial NER task. The NER background is provided in depth in Chapter 2. This discussion will cover the selection of the features used to to train the SVM, the generation of the training corpus, the preprocessing of the text, and finally, the results on the Wikipedia articles testing corpus.

Figure 7 shows the first part of the article for the Spanish-American war, with the geospatial NEs highlighted in green. This image depicts the strings of text that we recognized as a result of the SVM processing. The words highlighted in red are signifiers of geospatial relationships, and will be discussed in depth in Chapter 7.

The performance measures for this task are defined in Chapter 2, equations 1, 2, and 3, the precision, recall, and f-measure respectively. More specifically, in this initial NE recognition task we desired high recall, even with a limited sacrifice of precision. A high recall means that we are extracting all of the possible candidate NE strings, even if it is at the sacrifice of precision. After training, the SVM achieved a recall of 99.8%,
Spanish–American War

From Wikipedia, the free encyclopedia

(REDirected from Spanish American War)

The Spanish–American War was an armed military conflict between Spain and the United States that took place between April and August 1898, over the issues of the liberation of Cuba. The war began after American demand for the resolution of the Cuban fight for independence was rejected by Spain. Strong expansionist sentiment in the United States motivated the government to develop a plan for annexation of Spain's remaining overseas territories including the Philippines, Puerto Rico, and Guam.\[9\]

The revolution in Havana prompted the United States to send in the warship USS Maine to indicate high national interest. Tension among the American people was raised because of the explosion of the USS Maine, and the yellow journalist newspapers that accused the Spanish of oppression in their colonies, agitating American public opinion. The war ended after victories for the United States in the Philippine Islands and Cuba.

On December 10, 1898, the signing of the Treaty of Paris gave the United States control of Cuba, the Philippines, Puerto Rico, and Guam.

Figure 7: Example Wikipedia Article with Geospatial NEs Highlighted

with precision of 51.0% when tested against our corpus of Wikipedia articles. This indicated that the SVM was indeed extracting almost all of the candidate entities, at the cost of also extracting other words and phrases as well.

5.2 SVM Feature Selection

Before training the SVM for the NE extraction, the features describing the geospatial named entities had to be defined. For this task, we drew primarily from the work done for word-based feature selection by Habib in [32]. She used lexical, word-shape, and other language-independent features to train an SVM for the extraction of NEs in the biomedical domain. Her work in this area was the basis for the features
used in this task. Furthermore, the work done by Cucerzan et al. in [18] on extracting NEs from Wikipedia provided further features.

As stated in the SVM overview in Chapter 3, the SVM feature vectors must be completely numeric, so the features discussed below are numeric translations of the word features. For the SVM, the values of each feature are scaled between -1 and 1. The features for the SVM in this task fell into two categories, either binary or continuous features. The binary feature functions take a string as an argument, and returned either a 0 or a 1. The continuous feature functions also take a string as an argument, but they instead returned a decimal value between -1 and 1. After putting all the features together, the resulting vector was a series of features, all with values between -1 and 1.

Initially, we attempted to use a window-based approach for generating features. The article text was split into five word overlapping “windows”, and a set of features was generated for each window. For instance, the sentence ‘I ran down to the park’ would yield two windows, ‘I ran down to the’ and ‘ran down to the park’. For each of these windows, the center word in the window was the word under consideration, and features for the window as a whole were generated relative to that word. These features included:

- Character count in each word, including vowel and consonant counts separately
- Individual word shape (capitalization, multiple capitals)
- Numeric and alphanumeric word structure
- Punctuation between and around words
- Word shape of all the words in the window together
We selected our feature set based on the above list for these windows. After generating a set of training data from the Reuters Corpus and the CoNLL 2003, 2004, and 2005 datasets, we did preliminary testing on the hand-tagged Wikipedia articles. For the SVM trained with five word windows, the performance (f-measure) did not rise above 50% for the tagging of geospatial NEs. We also tried other odd-numbered windows, including three and seven word windows, with similar results, though both three and seven word windows lost about 10% in performance, with the most significant drop in the precision. In all cases, the recall was not above 70%. As our target recall for this task was 90+, the window-based SVM did not appear to be a viable approach to the NE extraction. Furthermore, in the vectors that were selected by the trained SVM, we were presented with some difficulty in resolving the NEs from the overlapping windows. Table 4 summarizes these results.

Therefore, we re-evaluated our approach to target the structure of the geospatial NEs more specifically. Herskovits has done extensive research into locative expressions, and produced two of the seminal works on the semantics and structure of these expressions. After background research based on the work of Herskovits, in [37] and [36], we postulated that each word in the geospatial NE should be independent.
That is, we should be able to generate feature vectors for the SVM for each individual word in the article text, and that this would provide the best overall performance. This hypothesis was confirmed by the results discussed below. In this approach, the article text was split into an array of individual words, and each word was processed separately to generate a feature vector for the SVM. After testing various combinations of features, and even weighting the features by different amounts, we used the following features in the final implementation, with equal weighting on all features:

- Word length: alphanumeric character count, ignoring punctuation marks
- Vowel count
- Consonant count
- Capital count
- First capital: if the word begins with a capital letter
- If the word contains two vowels together
- If the word contains two identical consonants together (such as dd or mm)
- If the word ends with a comma
- If the word ends with a period
- If the word ends with a hyphen
- If the word ends with a semi-colon
- If the word ends with a colon
- If the word begins with a quote
- If the word ends with a quote
- If the word begins with a parenthesis
- If the word ends with a parenthesis
- If the word is plural
- If the word is a single character
• If the word is alphanumeric (mixed letters and numbers)
• If the word is purely numeric
• If the word is purely text
• If the word is all uppercase
• If the word is all lowercase
• If the word has an English suffix
• If the word has an English prefix
• If the word is a preposition
• If the word is an article
• If the word is a pronoun
• If the word is a conjunction
• If the word is an interrogative
• If the word is an auxiliary verb

5.3 Training Corpus Generation

Two different approaches were taken in the training of the SVM. First, we generated a training corpus for SVM that used the text windows approach, and when that approach did not produce the desired results, we generated a separate corpus for the individual word approach.

We downloaded a number of different named entity recognition data sets that had already been tagged to provide the training materiel for the SVM, along with other resources that would require hand-tagging to provide training data:

• To provide a list of English words, we downloaded version 6 of the Spell Checker-Oriented Word List (SCOWL)\textsuperscript{14} from SourceForge.

\textsuperscript{14}http://downloads.sourceforge.net/wordlist/scowl-6.tar.gz, accessed October 18, 2008
• The Conference on Natural Language Learning (CoNLL) 2003 shared task dataset on multi-language NE tagging, from the conference archive\textsuperscript{15}. This dataset contains tagged named entities for PER, LOC, and ORG, in English. This dataset is assembled automatically from the Reuters Corpus.

• The Reuters Corpus, available with a signed license agreement at the National Institute of Science and Technology website\textsuperscript{16}, as Reuters no longer distributes the corpus. 15,000 lines were selected from the Reuters corpus and hand-tagged for all LOC entities.

• The CoNLL 2004 shared task dataset, on Semantic Role Labeling from the conference archive\textsuperscript{17}. The dataset is intended for a different shared task, but does provide a set of English NEs tagged with LOC tags.

• The CoNLL 2005 shared task dataset, on Semantic Role Labeling from the conference archive\textsuperscript{18}. The dataset is intended for a different shared task, but does provide a set of English NEs tagged with LOC tags.

• The Geonames database, which is a 6.2 million record database of place names, from cities, counties and countries, to land features, rivers, and lakes. The Geonames database is provided at the Geonames website\textsuperscript{19}.

For the initial, window based approach, we required a corpus of five-word windows, each tagged according to whether the center word in the window is or is not a

\textsuperscript{15}\url{http://www.cnts.ua.ac.be/conll2003/ner/} accessed October 18, 2008

\textsuperscript{16}\url{http://trec.nist.gov/data/reuters/reuters.html}. The corpus was received from Reuters November 19, 2008

\textsuperscript{17}\url{http://www.isi.upc.es/~srlconll/st04/st04.html} accessed October 18, 2008

\textsuperscript{18}\url{http://www.isi.upc.es/~srlconll/soft.html} accessed October 18, 2008

\textsuperscript{19}\url{http://www.geonames.org/export/}, accessed January 25, 2009
geospatial named entity. Between the Reuters corpus and the three CoNLL datasets, we generated a corpus of 20,000 hand-tagged five-word windows, of which 10% were geospatial NEs. The SVM was trained and tested, but as mentioned in the previous section, we were forced to switch the individual word approach.

To generate the training corpus for the individual word-based approach to training the SVM, we re-processed the 20,000 word corpus to change each to a single word, yielding a starting corpus. We then added in the the SCOWL wordlist as negative (non-geospatial) examples. Initially, we added the entire Geonames database to the training corpus, approximately 2.5 million unique words. However, we were unable to train an SVM with the resulting corpus, because the examples in the corpus were divergent. We then extracted only the records in the Geonames database that fall into either the P or A Geonames codes (Administrative division and Population center), which provided the names of Cities, Villages, Countries, States, and Regions. Combined with the previous hand-tagged data, this resulted in a training set of 1.2 million unique words. This was the final corpus used to train the SVM.

5.4 Testing Corpus Generation

To generate the testing corpus, we selected 90 articles from Wikipedia on battles and wars around the world. Before the text of each of the articles could be tagged, the actual text returned from the database required preprocessing to remove the Wikipedia-specific markup, HTML, and other extraneous information. Preprocessing of the pages was done in five steps. In this research, we were only concerned with the
text of each article that related directly to the topic, which we refer to as the primary text.

First, extraneous sections were removed from the end of each article text. In many cases, Wikipedia articles will end in sections titled “References”, “See Also”, or “Notes”. These sections were bibliographic references, notes, and external links that did not contribute any information (especially locations) to the primary text. In many cases, the bibliographic references contained publisher addresses, which actually caused problems for the geospatial resolution in the next phase of the processing, because they are country, state, and city locations that were not actually related to the primary text. Therefore, all these sections were stripped from the page text.

Second, all the HTML markup was stripped from the page. While most markup in Wikipedia pages is done in wiki markup, some specific markup in the pages is still done in HTML. These HTML tags and their content did not contribute to the primary text of the page, so in all cases, the HTML was simply stripped out of the page text.

Third, the Wikipedia infoboxes in the page were processed. Infoboxes are special constructs that can be embedded in Wikipedia pages that reference external blocks of text. For instance, the World War II page has the WW2InfoBox. This is actually a separate page in the database, with its own text and markup. Infoboxes in Wikipedia are signified by double-braces: \{\{WW2InfoBox\}\}. A certain number of these infoboxes were system infoboxes, meant to convey information to the MediaWiki engine for processing and display. These infoboxes were removed completely. However, if the infobox contributed to the primary text of the page, one of two operations were performed. In a small set of cases, the infobox would be a ‘main’, ‘seealso’,
‘see details’, or ‘see’ infobox. These infoboxes are links to another full article page. In this case, instead of inserting the full text of another article, these infoboxes were replaced with a Wikipedia link to the other page. For all the other infoboxes, the full text of the infobox was inserted into the primary text. Because infoboxes can also contain infoboxes, this processing was recursive, expanding infoboxes in the text until none remained.

Fourth, Wikipedia-format links in the text were processed. Any link surrounded by a double square bracket [[Page_Link]] represents a link to another article in Wikipedia. A link can take two forms. The first is the [[Page_Link]] form, which is just a link to the other page. The second form adds a caption to the link, [[Page_Link | A caption for this link]], and only the caption is displayed in the rendered text. For all the links, the link text was removed, and replaced with either the name of linked page for links of the first form, or with the caption text for links of the second form. A list of all links removed from the page was also stored and associated with the page, in case the links were needed later in the processing.

Fifth and finally, a simple cleanup was done for any remaining link or infobox symbols in the text. In this step, links to external sites were removed also. In wiki markup, a link to an external site is surrounded by a single square bracket: [http://linkaddress.com]. These were removed completely. Also, special characters, unicode symbols, and some other special symbols were removed from the page text, as these did not add any information, and tended to cause exceptions and problems in the later string processing code.

The cleaned primary text of each page was then split into an array of individual
words, and the individual words were hand-tagged as geospatial NEs or not, either 1 or 0.

5.5 SVM Training

Once the two corpuses were selected, the code was written to generate the feature vector for each word in the training and testing corpus. As mentioned before, each function for the feature generation was either binary, returning -1 or 1, or continuous, returning a decimal value between -1 and 1. Based on the list of features enumerated earlier in this chapter, a function was written to take a word as an argument, and return either a binary or continuous numeric result. The features were numbered from 1 to 60.

After training vector generation, the initial training of the SVM was performed. LibSVM trains an SVM based on the feature vectors, and on two other parameters, $C$ and $\gamma$, which are the cost and degree coefficient of the SVM kernel function. LibSVM provides a Python script to perform a grid search for the optimal $C$ and $\gamma$ values. The grid search script starts at a random value of $C$ and $\gamma$, and then searches in 1-step increments around those values for better values, where the steps are approximately 1 increment of $\log_2$ of the $C$ and $\gamma$ parameter values. It also will randomly jump to a new $(C, \gamma)$ starting point to try and avoid local maximums in the fitness graph.

After using the grid search, the optimal parameters were determined to be $C = 8$ and $\gamma = 0.03125$.

The SVM for the NE recognition was trained with the $C$ and $\gamma$ parameters, as
input to a C-SVC SVM, using a radial basis function kernel. C-SVC is the standard cost-based support vector classification algorithm, based on the $C$ cost parameter. All other parameters to the SVM were left as defaults. The SVM was then trained using the training corpus feature vectors, requiring 36 hours to complete. We tested other kernel functions, and other values of input parameters to the SVM, but the optimal parameters were the $C$ and $\gamma$ parameters described before, the radial basis function kernel, and the C-SVC SVM.

The trained SVM model was then used for the geospatial NER task throughout the rest of this research work.

5.6 Testing and Results

Once the SVM was trained with the training corpus, we were ready to test the NE recognition on the Wikipedia corpus. As stated at the beginning of this chapter, we were targeting a high recall for the performance of the SVM. For each article in the Wikipedia testing corpus, the feature vectors were generated, and the feature vectors were tested through the SVM. Each feature vector corresponds to a word in the article, and the LibSVM library returned either a 1 or a 0 for the feature vector, based on the trained model. This set of result vectors was matched back to the words in the article, resulting in a 1 or a 0 for each word in the article, indicating whether the word was or was not a geospatial NE. Each of the test articles had previously been hand-tagged, either 1 or 0, so the result vector from the SVM was matched against the hand tagging. A correct 1 for 1 or 0 for 0 was marked as a successful NE
recognition, a 0/1 mismatch was identified as an incorrect NE recognition or missed recognition.

The precision, recall, and f-measure numbers were calculated based on these numbers. Table 5 shows the NE recognition results directly from the SVM.

The table shows that we were successful in training the SVM to return high recall numbers, recognizing 99.8% of the geospatial named entities in the Wikipedia articles. Unfortunately, this was at the cost of precision. The SVM returned noisy results with only 51.0% precision, but almost 100% of the geospatial NEs, and the noise was processed out during the NE resolution process, covered in the next chapter.

### 5.7 Discussion

During this phase, training the SVM to perform the geospatial NER, we ran into our most significant challenges in building the training corpus and training the SVM for high recall while maintaining good precision. We drew on the Reuters corpus, the CoNLL 2003, 2004, and 2005 datasets, our hand-tagged Wikipedia data, and the Geonames database for the training corpus, along with the SCOWL wordlist for non-geospatial words. Table 6 shows the precision, recall, and f-measure numbers for corpuses created with different subsets of the available data. Note that in each case, we separately optimized the $C$ and $\gamma$ parameters for each specific training set. The

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.510</td>
<td>0.998</td>
<td>0.675</td>
</tr>
</tbody>
</table>
performance numbers were generated through testing against the Wikipedia corpus. When we used a subset of the hand-tagged Wikipedia data for training, those articles were removed from the testing corpus to avoid bias.

The recall from the SVM went up significantly when we introduced the Geonames database as part of the training corpus, but the full Geonames database, some 2.5 million rows, significantly impacted the precision. We then worked on filtering the Geonames database to the final set, using only administrative division and population center names, which brought the precision back up to acceptable levels.

We also found that when we used the full Geonames database, it greatly impacted the training time for the SVM. We were forced to cancel the SVM training after 5 days, because it was no longer progressing. Obviously, this problem was in part because the training set was so large, but in running metrics on the corpus, we found that there were multiple occurrences of words marked both as positive examples (geospatial entities), and negative examples. We found that the Geonames database is so large it contains many place names that are also normal words in the English

---

### Table 6: Training Corpus Combination Results for the NER SVM

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.38</td>
<td>0.63</td>
<td>0.47</td>
</tr>
<tr>
<td>R + C03 + C04 + C05</td>
<td>0.45</td>
<td>0.68</td>
<td>0.54</td>
</tr>
<tr>
<td>R + C03 + C04 + C05 + SCW</td>
<td>0.49</td>
<td>0.75</td>
<td>0.59</td>
</tr>
<tr>
<td>Wiki</td>
<td>0.65</td>
<td>0.34</td>
<td>0.44</td>
</tr>
<tr>
<td>Wiki + R + C03 + C04 + C05 + SCW</td>
<td>0.60</td>
<td>0.55</td>
<td>0.57</td>
</tr>
<tr>
<td>Geo + R + C03 + C04 + C05 + SCW</td>
<td>TC</td>
<td>TC</td>
<td>TC</td>
</tr>
<tr>
<td><strong>FGeo + R + C03 + C04 + C05 + SCW</strong></td>
<td><strong>0.51</strong></td>
<td><strong>0.99</strong></td>
<td><strong>0.67</strong></td>
</tr>
</tbody>
</table>

language, words not normally used to describe geospatial entities. We theorize that this tagging of words as both positive and negative examples caused a divergence in the training, effectively forcing the SVM into a state where it could not continue. This finding caused us to look at filtering the Geonames database to the geospatial level of interest, resulting in the final training corpus.

After selecting the optimal training corpus, we also spent some time working to optimize the feature set for the SVM. Based on the list of possible NER features (Table 1 in Chapter 2), we tried different combinations of features to find an optimal set. Table 7 shows a breakdown of the feature sets we tested separately. For the purposes of the testing, we grouped the features into the nine sets described below the table. We found that the list presented earlier in this chapter provided the best performance by close to 10% in f-measure over any other combination of features. As with the corpus selection, we optimized the $C$ and $\gamma$ for each feature set.

After selecting the final set of features, we performed some initial experiments in weighting the features differently. We found that in all cases, changing the weighting

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHP+PUN+HYP+SEP</td>
<td>0.44</td>
</tr>
<tr>
<td>SHP+CAP+PUN+HYP+SEP</td>
<td>0.47</td>
</tr>
<tr>
<td>CAP+VOW+CON</td>
<td>0.51</td>
</tr>
<tr>
<td>SHP+CAP</td>
<td>0.54</td>
</tr>
<tr>
<td>SHP+CAP+VOW+CON</td>
<td>0.56</td>
</tr>
<tr>
<td>SHP+CAP+VOW+CON+AFF+POS</td>
<td>0.58</td>
</tr>
<tr>
<td><strong>SHP+CAP+VOW+CON+PUN+HYP+SEP+AFF+POS</strong></td>
<td><strong>0.67</strong></td>
</tr>
</tbody>
</table>

- **SHP**: Word shape, including length and alphanumeric content.
- **CAP**: Capital positions and counts.
- **VOW**: Vowel positions and counts.
- **CON**: Consonant positions and counts.
- **PUN**: Punctuation.
- **HYP**: Hyphens.
- **SEP**: Separators, quotes, parenthesis, etc.
- **AFF**: Affixes.
- **POS**: Part of speech categorization.

Table 7: Feature Set Combination Results for the NER SVM
on the features resulted in an f-measure loss between 0.21% and 3.4%, not a significant impact on the performance. We concluded that, for the scope of this research, we did not need to spend a significant amount of time on working with the feature weighting, and simply weighted all features equally at 1.0.

Once we decided on the optimal feature set for the SVM, we did go back and re-train and test the SVM with some of the other corpus combinations. However, we found that the results of the original corpus selection held true, and the corpus based on the filtered Geonames database provided the best training results for the SVM.
6 Extracted Geospatial Named Entity Resolution

6.1 Overview and Objective

Once we extracted a set of candidate geospatial NEs from the article text, we had a list of strings, each of which was marked by the SVM as a possible NE. For each of these candidate strings, the objective of this stage of the processing was to decide whether each candidate string was, in fact, a geospatial NE, and to determine the correct \( (\phi, \lambda) \) coordinates for the place name in context of the article, and to construct a geospatial entity from the place name, spatial location, and an assumed point geometry.

To resolve the candidate NE, a lookup was made using the Google Geocoder\textsuperscript{20}. If the entity reference resolved to a single geospatial location, no further action was required. However, if the initial lookup did not resolve the geospatial reference, the context of the place name in the article and our data structure and rule set was used to decide the correct spatial location for the geospatial NE for this article text.

Wang et al. and Ding et al. define the specific geospatial context which we considered \([24,67]\). \([67]\) defines three kinds of geospatial references in web documents, provider locations, serving locations, and context locations. This research targeted context locations, defined to be “the geographic location that the content of a web resource describes” \([67]\). Furthermore, \([24]\) draws a distinction between candidate geographic scope (the set of geographic nodes in the entity’s possible geographical hierarchy) and estimated geographic scope (the set of geographic nodes in the candi-

\textsuperscript{20}\url{http://code.google.com/apis/maps/documentation/geocoding/index.html}
date set after pruning the the candidate scope). Ding et al. also proposes a number of algorithms for generating the candidate geographic scope and pruning the scope [24]. These algorithms can also be combined with the context-disambiguation algorithms proposed by Milne in [51]. To narrow each ambiguous geospatial NE, we considered the estimated geographic scope of the article as a whole.

In [48], Martins et al. provide an excellent overview of the current best approaches in geographical information retrieval, including reference disambiguation. Bilhaut et al. [4] present a method for off-line geospatial reference analysis and extraction so that the references serve as indexes for the text. Rattenbury, Good, and Naaman demonstrated an algorithm that used term frequency and geolocation context to extract location references from Flickr tags in [55] which can be adapted to Wikipedia articles. This research also applied techniques from [9], which used context from Wikipedia article categories for disambiguation.

The task under consideration in this chapter is very close to that of word sense disambiguation, as defined by Cucerzan [18], only that we consider the geospatial context and domain instead of the lexical context and domain. Sehgal, Getoor, and Viechnicki have demonstrated good results for geospatial entity resolution using both spatial (coordinate) and non-spatial (lexical) features of the geospatial entities [59]. Zong et al. have demonstrated a rule-based method for place name assignment, achieving a precision of 88.6% on disambiguating place names in the United States, from the Digital Library for Earth System Education (DLESE) metadata. In this chapter, we explain our approach that expanded this to general place names around the world.
The primary challenge to the resolution and disambiguation is that multiple names can refer to the same place, and multiple places can have the same name. The statistics for these place name and reference overlap are covered in depth by Smith and Crane, including Table 8, from [60].

The statistics in Table 8 demonstrate that the largest area with ambiguous place names is North and Central America, making these areas the prime areas for our research. Most of the name overlaps are in city names, and not in state/province names. These statistics are further supported by the location name breakdowns on the individual Wikipedia articles in figures 8, 9, 10, and 11. The graphics in these figures are broken down into countries, states, foreign cities, foreign features, US cities, and US features, where the foreign and US features are any non-state or city geospatial features, such as rivers, hills, and ridges. The performance of the Geografikos system on NER extraction and disambiguation is indicated in the caption of each figure. The Battle of Gettysburg, with over 50% US city names, many of which overlap, resulted in the lowest performance. The Liberty Incident, in the Mediterranean Sea, with less than 10% US city and state names combined, showed significantly better
performance.

The rest of this chapter covers the specific approach, data structures, and rules used to perform the disambiguation on the candidate geospatial NEs provided by the SVM, and the overall results of the entire NE extraction and disambiguation process.

### 6.2 Mining Geospatial Entities

Throughout this thesis until now, we have discussed only the recognition of geospatial NEs, or place names, from the text of our corpus. As the title of this thesis involves mining geospatial entities, and not mining geospatial NEs, we must formally define geospatial entities, and resolve the differences between geospatial NEs and geospatial entities.
Figure 9: War of 1812 Place Name Detail (f-measure = 0.816)

Figure 10: World War II Place Name Detail (f-measure = 0.870)
In [71], Worboys provides an excellent definition for a geospatial entity, which we have adapted with [6] to bring in the topological properties. A geospatial entity is a representation of a geospatial concept, defined by the properties that fall into 5 categories. Figure 12, adapted from [71], shows these categories.

The five property sets that define a geospatial entity are:

- **Attributes**: These are the textual or numerical attributes of a geospatial entity, such as place name, population, or other attributes inherent to the entity.
- **Spatial**: This property set represents the coordinate system, datum, and other information used to locate a geospatial entity on a map or other representation. Iliffe provides an excellent overview of this as it relates to GIS in [40].
- **Geometric or Graphical**: This is the set of properties that define the shape of
the entity, normally a point, polygon, or line. The Open Geospatial Consortium (OGC) Geographic Markup Language (GML) specification, a data model for representing geospatial information, contains a comprehensive set of geometric primitives [17].

- **Temporal**: These properties of a geospatial entity are those having to do with time, such as the date that a city is incorporated, or that a country is established. Worboys gives an excellent overview of these properties in [70].

- **Topological**: The topologic properties of a geospatial entity are not actually inherent properties of the entity, but instead describe the relationships between this and other geospatial relationships. See [6] for an excellent overview on topology.

As mentioned previously, geospatial NEs are simply the place name attribute of

---

21 This citation is for version 2.0, the current version of GML is 3.2.1, which can be found at http://www.opengeospatial.org/standards/gml
the larger geospatial entity. In phase two of the processing, covered by this chapter, we use the place name from phase one to geocode the correct spatial location for the place name based on the article context. We create a geospatial entity with the place name and spatial location, and assume a point geometry. While this only fulfills three of the five aspects of a geospatial entity described above (the green circles in Figure 12), it does provide enough information to construct a useful geospatial entity. For now, we disregard the temporal and topological aspects of the geospatial entity, though further research to include this information would certainly make the geospatial entities more useful.

In constructing the geospatial entities, we work at a scale such that we resolve each place name to a single coordinate pair, assuming a point geometry. Though at a close enough scale, cities, states and countries should be actually treated spatially as polygons, for the purposes of this research, we treat them as points. Somewhat in conflict with correct geographic procedure, we also treat other linear geographic features, such as rivers as points. This is somewhat incorrect. However, there are many databases that contain the correct geometric information for cities, states, countries, rivers, and other geospatial entities. After this phase of the research, we have geospatial entities, with non-ambiguous place name attributes, referring only to a single possible spatial location. Therefore, we could potentially use the place name of geospatial entity generated by the Geografikos system to look up the correct geometry (line or polygon) from a separate database. Certainly, for proper GIR in some contexts, we must have the correct polygon or linear geometry for a geospatial entity. However, because our focus is the initial construction of the geospatial enti-
ties from the free text of the Wikipedia articles, we feel that we can overlook these inconsistencies for now.

6.3 Google Geocoder

This research relied on Google Geocoder as the gazetteer and geocoder because the focus of the project was not to implement a gazetteer and geocoder, as much research has already been done in this area, with many free gazetteers already available. Vestavik and D’Roza et al provide an excellent overview of existing technology in this area [26, 66]. Simply put, Google Geocoder takes a place name as input, and returns 0 or more possible \((\phi, \lambda)\) coordinates for the place name, as shown in Figure 13.

Google Geocoder provides a simple REST-based interface that can be queried over HTTP, which returns data in a variety of formats, including XML (KML Schema), JSON (JavaScript Object Notation), and CSV (Comma-Separated Variable). A request to the geocoder is made through a GET request to the geocoder url with the address string:

```
http://maps.google.com/maps/geo?q=Mountain+View,+CA&output=xml&oe=
```
utf8&key=your_api_key

The ‘q’ parameter is the string that we wish to geocode, the ‘output’ parameter specifies the format in which we want the information returned, the ‘oe’ parameter is the character encoding, and the ‘key’ parameter is the API key that uniquely identifies a user with Google.

For each address string query, Google Geocoder returns 0 or more placemarks as a result. Each placemark corresponds to a single \((\phi, \lambda)\) coordinate pair. If the address string is unknown, or another error occurs, the geocoder returns no placemarks. If the query resolves to a single location, Google Geocoder returns a single placemark. If the query string is ambiguous, having more than one possible location, it may return more than one placemark, corresponding to a variety of locations. A country code can also be passed as part of the request parameters to bias the geocoder toward a specific country. For our research, each set of placemarks was returned in KML as shown Figure 14, which is the response to a query for the address “1600 amphitheatre mountain view ca”, Google headquarters. This example was taken from the Google Geocoder documentation\(^2\).

Google Geocoder returns locations at roughly four different levels, Country, State, City, and Street/Feature, depending on the ambiguity of the address query string and the area of the world. Each location is returned with a location at one of the four levels and a (latitude,longitude) coordinate. This four-level hierarchy directly fed the logic that was used to disambiguate the candidate NEs through our location

\(^2\text{http://code.google.com/apis/maps/documentation/geocoding/index.html, accessed April 6, 2009}\)
<kml xmlns="http://earth.google.com/kml/2.0">  
  <Response>
    <name>1600 amphitheatre mountain view ca</name>
    <Status>
      <code>200</code>
      <request>geocode</request>
    </Status>
    <Placemark>
      <address>
        1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA
      </address>
      <AddressDetails Accuracy="8">
        <Country>
          <CountryNameCode>US</CountryNameCode>
          <AdministrativeArea>
            <AdministrativeAreaName>CA</AdministrativeAreaName>
          </AdministrativeArea>
          <SubAdministrativeArea>
            <SubAdministrativeAreaName>Santa Clara</SubAdministrativeAreaName>
          </SubAdministrativeArea>
          <Locality>
            <LocalityName>Mountain View</LocalityName>
            <Thoroughfare>
              <ThoroughfareName>1600 Amphitheatre Pkwy</ThoroughfareName>
            </Thoroughfare>
            <PostalCode>
              <PostalCodeNumber>94043</PostalCodeNumber>
            </PostalCode>
          </Locality>
        </Country>
      </AddressDetails>
      <Point>
        <coordinates>-122.083739, 37.423021, 0</coordinates>
      </Point>
    </Placemark>
  </Response>
</kml>

Figure 14: Example XML Response from Google Geocoder
tree data structure.

Through the rest of this discussion, the placemarks returned by Google Geocoder are referred to as locations. In the Geografikos package, the address string returned from Google Geocoder is separated into the four parts, and each part is stored separately, with the coordinates. The coordinates are also checked against the existing set of locations in the database for uniqueness. For this research, coordinates must be more than 1/10th of a mile apart to be considered a unique location, which is slightly below a city block in resolution.

6.4 Location Tree Data Structure

The output from the Google Geocoder query for each candidate named entity was fed through our novel location tree data structure, which, along with an algorithm driven by a simple set of rules, ordered the locations by geospatial division, and aided in the disambiguation of any ambiguous place name.

The location tree operates at four levels of geospatial hierarchy, Country, State/Province, City, and Street/Geographic Feature. As mentioned previously, each placemark response from the geocoder corresponded to one of these geospatial divisions. A separate set of rules governed the insertion of a location at each level of the tree. For simple additions to the tree, the rules are as follows:

1. If the location is a country, add it to the tree
2. If the location is a state or province, add it to the tree
3. If the location is a city in the US, and the parent node in the tree is a state, add it to the tree
4. If the location is a city outside the US, add it to the tree
5. If the location is a street or feature inside the US, and the parent node in the tree is a state or city, add it to the tree
6. If the location is a street or outside the US, and the parent node in the tree is either a country or city, add it to the tree

Rules 3 and 5 are US-specific because Google Geocoder has much more detailed place information for the US than other areas of the world, requiring some extra processing for locations in the US. In rules 3, 5, and 6, “parent node” refers to the node in the tree that is the predecessor in the tree to the location under consideration. For example, the node consisting of the state of Colorado would be the parent node to the node consisting of the location Denver, CO.

Any locations that did not match one of these rules were placed on a pending list. Each time a new location was added to the tree, the tree was re-sorted to ensure that the correct hierarchy was maintained, and each location on the pending list was re-checked to see if it now matched the rules for insertion into the tree.

The following simple example demonstrates this process. Given a simple location tree, and a small list of pending locations, shown in Figure 16, this example demonstrates what happens when the location “Colorado, USA” is added to the tree. Figure 15 shows the color coding of the nodes in the tree.

Initially, in Figure 16, we have a simple location tree with a feature in the UK, and nothing else. Both Seattle and Denver are on the pending list because they cannot be added to the tree unless the parent node under which they are to be added is a state.
To add “Colorado, USA” to the tree, the first step is to determine the parent node for this location. The logical parent node for Colorado is the country USA, but because the USA is not in the tree, the parent node for Colorado is the root node of the tree. Therefore, Colorado is added directly under the root, shown in Figure 17.

The tree is now checked to make sure that the hierarchy is correct. It is, so next, each of the locations in the Pending list can now be re-processed. Since Colorado has
been added to the tree, the pending location Denver now has a parent node in the tree, and can be added to the tree, shown in Figure 18.

As an example of the re-sorting process, Figure 19 shows the tree after USA is added to the tree, but before the hierarchy check is run. USA is added directly underneath the root node of the tree, because it is a country. Once the hierarchy check is run on the tree, the tree is re-sorted to make sure that each node is underneath its correct parent node, resulting in the tree shown in Figure 20. The rest of the pending list (Seattle) would be processed in the same way.

In contrast with the previous examples, if the location returned from the geocoder had multiple placemarks (that is, it corresponded to multiple physical coordinates), another set of calculations was required before running the simple rule set to estab-
lish which set of physical coordinates was correct based on the article context. First, any ambiguous locations were automatically placed on the pending list until all other locations had been processed. This ensured that we had the most complete context possible, and that all non-ambiguous locations were added to the Tree before the ambiguous locations were processed.

Once we finished processing all the non-ambiguous locations in the candidate list, we had a location tree with some number of locations in it, and a set of pending locations, some of which were ambiguous, and could fit more than one place on the tree. As each node was inserted into the tree, a running count of nodes, and repeated nodes was kept. That is, for each node in the tree, we knew how many occurrences there were of the location in the article itself. Using this information, when each ambiguous location was considered for insertion, a weighting was placed on each possible node in the tree that could serve as the parent node for the ambiguous location. As an example, we consider the location “Cambridge”. This could either be Cambridge, MA, in the US, or Cambridge, UK. We consider the tree shown in

Figure 20: Location Tree After Hierarchy Check
Figure 21: Example Location Tree Showing Duplicated Node Count

Figure 21, where each node in the tree is shown with the number of occurrences of that node in the article text.

The location “Cambridge” can be added underneath either leaf node in the tree. This is determined by the same ruleset used to insert all other nodes. The weight calculation determines the correct node to use. The weight calculation is shown in equation 17. The weight for node \( n \) is determined by summing the insertion count for all the parent nodes of \( n \) in the tree (up to three parents), and dividing by the total insertion count for the tree. \( N_{i-1}, N_{i-2} \) and \( N_{i-3} \) are the possible parent nodes for the node under consideration, since the tree can be up to four levels deep. A parent node \( N_j \) is set to zero if the tree is not that many levels deep. The insertion count for each node is actually the total number of occurrences of the location represented by that node in the original article, so the insertion count for the whole tree is equal to the number of location occurrences in the article.

\[
Weight_n = \frac{(Ct_{country} + Ct_{state} + Ct_{city})}{Ct_{total}} \tag{17}
\]

Calculating the weight for the two possible parent nodes of “Cambridge”, we use 3 as the insertion count for UK, and \( 2+2 = 4 \) for the insertion count for Massachusetts, 2 from the Massachusetts node and 2 from the USA node. Dividing both by the
insertion count of 7 for the entire tree, we find $Weight_{UK} = 0.428$ and $Weight_{MA} = 0.572$. As the $Weight_{MA}$ is the higher of the two, “Cambridge” is associated with Cambridge, MA, and placed in the tree underneath Massachusetts.

This equation for the weight calculation was chosen based on empirical testing with the corpus of Wikipedia pages. We originally calculated the weighting factor based only on the insertion count for each possible parent node, which would be 3 for the UK, and 2 for Massachusetts. However, in testing, we found that some 40% of the ambiguous locations were placed under an incorrect parent node, compared to only 22% incorrect placement using the current equation. Dividing the insertion count by the total insertion count did not make a difference, and is done only to scale each weighting factor to the range $[0, 1]$. Our testing showed that we place 78% of the ambiguous locations correctly in the tree. This does not factor in locations that are not ambiguous, as they are always placed correctly in the tree.

As a demonstration of a complete location tree, Figure 22 shows the complete location tree for the Operation Nickel Grass Wikipedia article. The tree is produced using the GraphViz software package.\textsuperscript{23} Note that the Operation Orchard node is the only incorrect node placed in the tree. This is an NE that was incorrectly identified as a geospatial NE and geocoded. Corresponding to the information in the location tree article, Table 9 shows the geocoded $(\phi, \lambda)$ coordinates for each geospatial entity in the figure, arranged alphabetically.

\textsuperscript{23}http://www.graphviz.org
After the location tree rules had been completed and tested, we ran the SVM NER, described in Chapter 5, followed by the disambiguation and geospatial resolution, described in this chapter, on the entire Wikipedia article corpus, and calculated the precision, recall, and f-measure performance of the system. As with the performance of the raw NE extraction task, the performance of the resolution system was defined in terms of the same precision, recall, and f-measure equations defined in
Table 9: Physical (φ, λ) Coordinates for Geospatial Entities from the Operation Nickel Grass Article

<table>
<thead>
<tr>
<th>Place Name</th>
<th>Latitude</th>
<th>Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alava</td>
<td>42.91</td>
<td>-2.69839</td>
</tr>
<tr>
<td>Egypt</td>
<td>26.8206</td>
<td>30.8025</td>
</tr>
<tr>
<td>Ejido</td>
<td>36.7751</td>
<td>-2.81274</td>
</tr>
<tr>
<td>Europe</td>
<td>54.526</td>
<td>15.2551</td>
</tr>
<tr>
<td>GA</td>
<td>32.1574</td>
<td>-82.9071</td>
</tr>
<tr>
<td>Golan Heights</td>
<td>32.9918</td>
<td>35.6897</td>
</tr>
<tr>
<td>Iraq</td>
<td>33.2232</td>
<td>43.6793</td>
</tr>
<tr>
<td>Israel</td>
<td>31.0461</td>
<td>34.8516</td>
</tr>
<tr>
<td>Lajes Field</td>
<td>38.7612</td>
<td>-27.0942</td>
</tr>
<tr>
<td>Lajes Seia</td>
<td>40.4837</td>
<td>-7.71647</td>
</tr>
<tr>
<td>Mediterranean Sea</td>
<td>35.0</td>
<td>18.0</td>
</tr>
<tr>
<td>Operation Orchard</td>
<td>35.7078</td>
<td>39.8336</td>
</tr>
<tr>
<td>Portugal</td>
<td>39.3999</td>
<td>-8.22445</td>
</tr>
<tr>
<td>Sinai</td>
<td>29.5</td>
<td>33.8333</td>
</tr>
<tr>
<td>Syria</td>
<td>34.8021</td>
<td>38.9968</td>
</tr>
<tr>
<td>USA</td>
<td>37.0902</td>
<td>-95.7129</td>
</tr>
<tr>
<td>Vietnam</td>
<td>14.0583</td>
<td>108.277</td>
</tr>
</tbody>
</table>

Table 10: Raw SVM and Resolved Geospatial NE Results

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw Results</td>
<td>0.510</td>
<td>0.998</td>
<td>0.675</td>
</tr>
<tr>
<td>Resolved Results</td>
<td>0.869</td>
<td>0.789</td>
<td>0.820</td>
</tr>
</tbody>
</table>

Chapter 2, equations 1, 2, and 3.

Table 10 shows the raw results from the SVM NE recognition along with the results after processing through the geospatial resolution and disambiguation. While the geocoding resolution process does bring up the precision significantly, up 25% from the raw SVM, it does so at the cost of some recall, because some of the actual location strings are filtered out through the process. Across the Wikipedia corpus of 90 articles, we obtained an f-measure of 82.0%, meaning that we were able to
successfully geocode 82% of the location NEs in the articles to a coordinate pair. The lowest f-measure was the Battle of Gettysburg article, with and f-measure of 68.2%, up to the highest f-measure of 96.1%, for the article about the Dec. 7, 1941 attack on Pearl Harbor (not shown in the graph).

Figure 23 shows a more detailed breakdown of the precision, recall, and f-measure for a subset of the articles processed. The word count is shown in parentheses after the article name. The results in this chart show, in general, the same trend shown by figures 8, 9, 10, and 11 earlier in this chapter. The Geografikos package is generally able to extract locations with a higher performance from articles that have a higher percentage of non-ambiguous locations. The word count of the article does not appear to have any effect on the overall performance.

6.6 Discussion

During this phase, we found that the NE disambiguation and geocoding process generally worked as we desired. However, we did encounter challenges with name resolution in the geocoder, and with some areas of the world. We utilized the Google Geocoder, and it did provide an excellent geocoding service. There are a list of places in the world, however, where the geocoder returned errors. We were forced to manually return results for this subset of locations, approximately 15 locations in the Wikipedia testing corpus. More importantly, Google Geocoder returned US state and Canadian province names in a variety of formats, from two letter postal codes, to abbreviated names, to full names, depending on the input query, and the
location within the US. To account for this, we performed post-processing on the Google Geocoder results. For locations inside the US and Canada, we replaced all occurrences of state and province names in any format with the two letter state code, so that unified state and province names were passed into the location tree and context resolution process.

Additionally, we found that the geocoder performed much better on land-based locations than ocean and sea-based locations. This is primarily because land tends to be better divided and named than the ocean. This problem primarily rises to the fore in articles about naval battles, like the Battle of Midway article. Often, the locations in these articles are relative to a fixed land point in the ocean. These locations require

Figure 23: Detailed Results for a Subset of Wikipedia Articles
a different process for extraction and recognition, and were generally not resolved by our process.

We also found that Google Geocoder has much better data in its gazetteer for the US and Western Europe than other areas of the world. While we were able to resolve major cities in Asia and Africa, smaller-scale features were often returned as “not found” from the geocoder. Russian and Eastern Asian locations also introduced the problem of other character sets in Unicode characters. Google Geocoder seems to handle partial Unicode in other languages, but tended to get the actual locations incorrect. Fortunately, this problem was somewhat mitigated by the fact that most place names in the English Wikipedia are, in fact, in English. We also found that the articles in our corpus did not refer to very many features at smaller than a city level in general, so these geocoding problems did not significantly impact our performance.

Historical location names caused some small issues with the geocoder, but our research demonstrated that, for North America and Europe at least, Google Geocoder has historical names in its gazetteer going back at least 300 years. We had few problems in this area.

Finally, we found that we also had problems with the context-based disambiguation in North America in areas where the city and feature names have significant overlap. We found that at a scale smaller than the state level, North America has a significant number of city and feature names that overlap. The ultimate result of this was that places with globally unique city and feature names performed the best, as there is no need to run the context disambiguation algorithms. We found that the worst areas in North America were the northeastern US (it seems that every state
has at least one city named York, etc.), and in the southwestern US, where there is significant overlap between place names in the US and place names in Mexico. In general, our context disambiguation algorithm itself performed at 78%, but in these areas of the country, the performance dropped to around 64.5%, due to the significant place name overlap. This reduction in f-measure primarily arose because, for articles about battles in the northeast and southwest US, multiple state names are mentioned in equal proportion, so the context algorithm resulted in equal weighting on the city and feature names, with no clear place for the location. More work in this area is certainly warranted.
7 Open Geospatial Relation Identification

7.1 Overview and Objective

Relation extraction is an up and coming field, the objective of which is to extract a semantic relation (or relationship) between two entities in free text. Examples of relations could include “Client-Of”, “Located-At”, “Sibling-Of”, and “Subsidiary”. For information retrieval and comprehension tasks, these relations in the text can be used to build a network or graph of the way that the entities relate to one another.

As an example, consider the following sentence:

Dr. James Randolph is the head scientist at Harrington Biomedical Corporation.

A general purpose relation-extraction system should recognize that there is a person-affiliation relation between Dr. James Randolph and the Harrington Biomedical Corporation. It should furthermore recognize that Dr. James Randolph is a head scientist.

In this phase of the processing, we target natural language relations between the geospatial entities we extracted in phase two, which are marked in the text by the place names from phase one. We define an open geospatial relation as a relationship between a geospatial NE in a sentence and other significant phrases, nouns, and verbs in the same sentence. For example:

The communist North Korean Army assaulted South Korea at Seoul.
Our system identifies that there is a relationship between the place name “South Korea” and “Seoul”, and there exists a relationship between those place names and the noun phrase “The communist North Korean Army”.

Relation extractors can fall into three areas, as defined by Cohen in [15]. The first is systems that use simple regular expressions. The second area is slightly more complex, systems that use more complex templates. These can all be grouped under the heading of rule-based systems, as they all use some form of hand-generated ruleset for the detection and classification of the relations, such as the Brin DIPRE system [7]. The third area is that of NLP methods, which perform a substantial amount of sentence parsing to decompose the sentence into a structure that can be fed to a learning system. The overview here focuses on the third area, as the first two areas have been show to have good performance only in very restricted problem domains, not scaling well as the problem domain expands or changes [1].

Research in the field of relation extraction really came to the fore in 2003, and has been increasing since that time. Zelenko, Aone, and Richardella did some of the first work to demonstrate good performance in relation extraction, using an SVM to detect relations between persons and affiliations, and organizations and locations. They demonstrated an f-measure of 0.833 on ORG-LOC relations, with a precision of 0.917, and a recall of 0.763. They broke each sentence down into a set of subsequences, with each subsequence containing only two entities, of any kind. Each subsequence was then modeled as a series of features, $f_1..f_n$. These features were fed to an SVM which decided whether the subsequence contained a relation or not [73]. Research into using kernel methods with subsequences for relations was extended into the
biomedical field by Bunescu and Mooney, in 2006. They also broke sentences down into subsequences with only two entities, and used an SVM to detect relations that described protein interactions from the A1med corpus, with an f-measure of about 0.50. They also tested the same system on the ACE corpus, with an f-measure of 0.477 [8].

Good performance has also been demonstrated using dependency tree subgraphs as input to SVMs for the detection of relations, by Culotta and Sorensen in [20]. Each sentence was parsed out into its dependency graph, and the dependency graph was broken down into subtrees. An SVM was used to detect and classify the relations between entities, based on a feature vector describing each of the subtrees. Testing against the ACE corpus, they achieved precision of 0.812, recall of 0.518, and f-measure of 0.632 on the detection of relations, and precision of 0.675, recall of 0.350, and f-measure of 0.458 on the classification of relations [20].

Culotta, this time with McCallum and Betz, also demonstrated good results using linear-chain conditional random fields for learning the contextual and relational patterns that define relations [19]. Instead of extracting relations between any two entities in an article, they redefined the task as the extraction of relations between the principle entity in an article, and the other entities in the article. The principle entity in an article is single entity that an article describes. For example, the principle entity of the Wikipedia article about Microsoft, is the Microsoft Corporation. This refocusing of the task allows relation extraction to be approached as a sequence labeling task, like NER or part of speech tagging, which allows the conditional random fields to be applied. They achieved precision of 0.7177, recall of 0.5531, and f-measure
of 0.6136 using the CRF with relational patterns [19].

In the biomedical domain, entities are easier to extract with the relations between them, owing to the increased local context that the relations provide, so significant work has been done in this area, seeking to extract the relations between entities such as protein sites, gene codes, and protein locations, all of which exhibit the same text classification problems as relation extraction. In [15], Cohen gives a good summary of the current state of the art in this area, including the following: For relations between amino acids, the Protein Active Site Template Acquisition (PASTA) system demonstrated an f-measure of 0.82 (Gaizauskas et al.) [29]. Using text classifiers in combination with maximum entropy models to predict Gene Ontology codes achieved an f-measure of 0.72 (Raychaudhuri et al.) [56]. Finally, an SVM with a combined text and genome sequence kernel achieved an f-measure of 0.782 in predicting protein subcellular locations (Eskin and Agichtein) [27]. Giles and Wren demonstrated in [30] a method to extract the directional relations between genes, chemical, metabolites, phenotypes, and diseases from the MEDLINE corpus, achieving an f-measure of 0.57 using an SVM trained to classify based on sentence dependency graphs.

Specifically targeting Wikipedia, Nguyen, Matsuo, and Ishizuka demonstrated the use of the same kind of dependency subtree mining as Culotta and Sorensen, applying those techniques to the text of Wikipedia, to extract 13 different relations. They built the dependency subtrees by extension from the dependency paths for each sentence, guided by limited keyword recognition. As with the 2006 work of Culotta, McCallum, and Betz, they target relations between the principle entity in the Wikipedia article and other relations in the article text. Using an SVM trained
to classify feature vectors based on the dependency subtrees, they achieved precision of 0.2907, recall of 0.5386, and f-measure of 0.3776 with a hand-tagged training and testing corpus [53]. Stoutenberg and Kalita have also performed research into targeting the link structure of Wikipedia to acquire semantic relationships between entities, demonstrating precision of 0.8308 and recall of 0.8723 in general relationship extraction [61, 62].

In this research, we focused on open geospatial relations, which we define as relations between the extracted and resolved geospatial NEs and any other entities in the sentence. The relations are limited to a single sentence, and cannot be implicit. These relations are purely based on features of the natural language in each sentence, and not related to the earlier definition of topology in discussing geospatial entities from Chapter 6. Because the work covered in Chapters 5 and 6 was focused on extracting geospatial NEs, it was a logical extension of that work to feed the NE information back into this relation extraction process. Focusing on just geospatial NEs also allowed us to avoid the problem of relation classification, and work only on relation detection.

We sought to test out a simple feature set for geospatial relation recognition that combined some of the aspects of both the subsequence and dependency tree approaches described earlier. We desired to avoid the complicated deep parsing of the sentences, and only use NLP methods to do shallow parsing, combined with an SVM to learn the patterns that define the open geospatial relations. An SVM was chosen for this task because they tend to show the best results for this kind of relation extraction work. Also, kernel methods represent a search space much larger than could
Spanish–American War

From Wikipedia, the free encyclopedia
(Redirected from Spanish American War)

The Spanish–American War was an armed military conflict between Spain and the United States that took place between April and August 1898, over the issues of the liberation of Cuba. The war began after American demand for the resolution of the Cuban fight for independence was rejected by Spain. Strong expansionist sentiment in the United States motivated the government to develop a plan for annexation of Spain’s remaining overseas territories including the Philippines, Puerto Rico, and Guam.[3]

The revolution in Havana prompted the United States to send in the warship USS Maine to indicate high national interest. Tension among the American people was raised because of the explosion of the USS Maine, and the yellow journalist newspapers that accused the Spanish of oppression in their colonies, agitating American public opinion. The war ended after victories for the United States in the Philippine Islands and Cuba.

On December 10, 1898, the signing of the Treaty of Paris gave the United States control of Cuba, the Philippines, Puerto Rico, and Guam.

Figure 24: Wikipedia Article with Geospatial NEs and Relation Signifiers Highlighted

be represented by a straightforward feature-extraction approach, because the kernel function can explore an implicit feature space when calculating the similarity between two instances [20].

Based on current published work in this area, our research target for this chapter was to achieve an f-measure of 50% on the detection of the open geospatial relations in the same Wikipedia corpus used throughout this research. Figure 24 shows an example marked-up Wikipedia article. Words highlighted in green are the geospatial NEs detected and resolved by the Geografikos package. The words in red are signifiers that those sentences contain the open geospatial relations that we targeted in this research.
7.2 Feature Selection

In contrast to the simple single word features used to train the SVM for the NE recognition, the features required for relation extraction required the consideration of more complex features, including the following:

- Parts of Speech
- Sentence phrase chunks
- Absolute NE and part of speech positions
- Phrase positions relative to each other
- Phrase positions relative to the geospatial NEs
- Phrase positions relative to other parts of speech
- Part of speech and NE positions relative to each other

The features we selected for the relation extraction were based primarily on subsequences of the sentence. Using features that encoded the absolute and relative positions of the NE and key part-of-speech positions, we sought to have the SVM learn the features that defined the dependencies that most likely signified the presence of open geospatial relations.

After training and testing multiple combinations of the features enumerated above, up to and including using the full set of features, we found that simpler was better. We settled on the following features for the final training and testing:

1. Absolute part of speech positions
2. Absolute geospatial NE positions
3. Around each preposition, a four-word window with the position of any NE relative to the preposition
4. Around each preposition, a four-word window with the position of any noun phrase relative to the preposition

5. Each sentence could contain up to 8 of the relative phrases in feature 3 and 4

To demonstrate a little more fully, consider the following sentence. Prepositions are highlighted in red, noun phrases in green, and geospatial NE positions in blue.

After failing to strengthen their cause in the free elections held in South Korea during May 1950 and the refusal of South Korea to hold new elections per North Korean demands, the communist North Korean Army assaulted the South on June 25, 1950.

Consider the phrase “in South Korea”. “in” is a preposition, and the geospatial NE is one position after it in the sentence, and the noun phrase “the free elections” precedes it by two positions in the sentence. These relative positions were encoded into the final feature vector, which was identified as containing an open geospatial relation.

After extensive combination testing, we settled on this set of features as optimal. We found that a smaller feature set resulted in too few features for the SVM to pick out features that actually signified the presence of an open geospatial relationship. This caused the precision to drop rapidly, as the SVM was selecting many more sentences. In contrast, if we trained the SVM with more features than this set, the feature set became too specific to the set of training sentences, resulting in an overfit SVM. While the performance was good on the test set, the f-measure performance of
the SVM dropped to below 20% on the testing corpus, because the SVM learned the features that exactly defined only the geospatial relations in the training set.

For part of speech tagging, the LingPipe library was used, from the Alias-I software company.\footnote{http://alias-i.com/lingpipe/index.html} Provided with a part of speech model, LingPipe uses a Hidden Markov Model-based approach to determine the part of speech tag for each word in the sentence. For this research, we used a part of speech model generously provided by Suzette Stoutenburg, using the part of speech tags shown in Table 11.

\section*{7.3 Corpus Generation}

Drawing from the Wikipedia article corpus selected for the parts of this work, we drew the training and testing corpus from the same material. We selected 23 of the 90 articles, approximately 5000 sentences from which to draw the corpus. We selected the articles with the highest performance on the NER and disambiguation task from the first two phases, because the accuracy of that phase directly affected the performance in this phase.

The clean text of each article was obtained through the process described in Chapter 5, cleaning the HTML and wiki markup from the article texts. The clean text of each article was split into sentences using the IndoEuropeanSentenceModel in the LingPipe software. LingPipe provides a number of sentence models for determining the boundaries of each sentence in a text. Each sentence model is effectively a large set of rules about which character chunks, prefixes, suffixes, parts of speech, and punctuation marks do and do not end sentences in the specific model language. We
Table 11: Part of Speech Tag Set for this Research

<table>
<thead>
<tr>
<th>Tag</th>
<th>Type</th>
<th>Tag</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordinating Conjunction</td>
<td>VBZ</td>
<td>3rd Person Singular <em>is</em></td>
</tr>
<tr>
<td>CS</td>
<td>Subordinating Conjunction</td>
<td>VDB</td>
<td>Base <em>do</em></td>
</tr>
<tr>
<td>CSN</td>
<td>Comparative Conjunction</td>
<td>VDD</td>
<td>Past <em>did</em></td>
</tr>
<tr>
<td>CST</td>
<td>Complementizer (that)</td>
<td>VDG</td>
<td>Participle <em>doing</em></td>
</tr>
<tr>
<td>DB</td>
<td>Predeterminer</td>
<td>VDI</td>
<td>Infinitive <em>do</em></td>
</tr>
<tr>
<td>DD</td>
<td>Determiner</td>
<td>VDN</td>
<td>Participle <em>done</em></td>
</tr>
<tr>
<td>EX</td>
<td>Existential <em>there</em></td>
<td>VDZ</td>
<td>3rd Person Singular <em>does</em></td>
</tr>
<tr>
<td>GE</td>
<td>Genitive Marker 's</td>
<td>VHB</td>
<td>Base <em>have</em></td>
</tr>
<tr>
<td>II</td>
<td>Preposition</td>
<td>VHD</td>
<td>Past <em>had</em></td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
<td>VHG</td>
<td>Participle <em>having</em></td>
</tr>
<tr>
<td>JJR</td>
<td>Comparative Adjective</td>
<td>VHI</td>
<td>Infinitive <em>have</em></td>
</tr>
<tr>
<td>JJT</td>
<td>Superlative Adjective</td>
<td>VHN</td>
<td>Participle <em>had</em></td>
</tr>
<tr>
<td>MC</td>
<td>Number or Numeric</td>
<td>VHZ</td>
<td>3rd Person Singular <em>has</em></td>
</tr>
<tr>
<td>NN</td>
<td>Noun</td>
<td>VVB</td>
<td>Base Form Lexical Verb</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper Noun</td>
<td>VVD</td>
<td>Past Tense</td>
</tr>
<tr>
<td>NNS</td>
<td>Plural Noun</td>
<td>VVG</td>
<td>Present Participle</td>
</tr>
<tr>
<td>PN</td>
<td>Pronoun</td>
<td>VVI</td>
<td>Infinitive Lexical Verb</td>
</tr>
<tr>
<td>PND</td>
<td>Determiner as Pronoun</td>
<td>VVN</td>
<td>Past Participle</td>
</tr>
<tr>
<td>PNG</td>
<td>Genitive Pronoun</td>
<td>VVZ</td>
<td>3rd Person Singular <em>has</em></td>
</tr>
<tr>
<td>PNR</td>
<td>Relative Pronoun</td>
<td>VVNJ</td>
<td>Prenominal Past Participle</td>
</tr>
<tr>
<td>RR</td>
<td>Adverb</td>
<td>VVGJ</td>
<td>Prenominal Present Participle</td>
</tr>
<tr>
<td>RRR</td>
<td>Comparative Adverb</td>
<td>VVGN</td>
<td>Nominal Gerund</td>
</tr>
<tr>
<td>RRT</td>
<td>Superlative Adverb</td>
<td>(</td>
<td>Left Parenthesis</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol</td>
<td>)</td>
<td>Right Parenthesis</td>
</tr>
<tr>
<td>TO</td>
<td>Infinitive Marker</td>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>VM</td>
<td>Modal</td>
<td>.</td>
<td>End-of-sentence Period</td>
</tr>
<tr>
<td>VBB</td>
<td>Base <em>be, am, are</em></td>
<td>:</td>
<td>Dashes, Colons</td>
</tr>
<tr>
<td>VBD</td>
<td>Past <em>was, were</em></td>
<td>&quot;</td>
<td>Left Quote</td>
</tr>
<tr>
<td>VBG</td>
<td>Participle <em>being</em></td>
<td>&quot;</td>
<td>Right Quote</td>
</tr>
<tr>
<td>VBI</td>
<td>Infinitive <em>be</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBN</td>
<td>Participle <em>been</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
applied the IndoEuropeanSentenceModel, for the English language. After splitting, each sentence was considered twice, once as pure English text, and the second time with the positions of the geospatial NEs marked in the sentence, so that we obtained both noun phrase position information, and geospatial NE position information.

Each sentence was then hand-tagged with a 1 or a 0, indicating the presence or absence of an open geospatial relation. We hand-tagged approximately 5000 sentences, with about 10% of the sentences containing an open geospatial relation. This set of 5000 was split into the training and testing corpus, 4000 for training, 1000 for testing.

7.4 SVM Training

After selection of the training and testing corpuses, the training corpus was used to generate feature vectors for each sentence. Unlike the previous SVM, the features were not simply binary, and not continuous in the range $[-1, 1]$. Instead, the features could be any integer value. After complete generation of the complete feature vector, each vector was scaled as a whole. The scaling process individually re-scaled the range of each feature to $[-1, 1]$, converting each feature vector to the continuous decimal ranges that provide the best performance in LibSVM.

After training vector generation, the initial training of the SVM for relation detection was performed. A LibSVM SVM was trained based on the feature vectors, and on the parameters $C$ and $\gamma$, the cost and degree coefficient of the SVM kernel function. As explained previously, the grid search Python script provided with
LibSVM was used to find the optimal parameters, as described in Chapter 5. After using the grid search, the optimal parameters were determined to be $C = 512$ and $\gamma = 0.001953125$.

An SVM model was trained with the training vector set and the parameters found in the grid search. Because we only had a relatively small set of vectors for training, we used 10-fold cross validation to find the best model. The training corpus of 4000 vectors was randomly split 80%/20% into two sets. The 80%, 3200 vectors, was used to train the SVM, and the other 800 vectors used to test the SVM. This process was repeated 10 times, and the best of the 10 trained SVM models was taken as the final model for testing.

### 7.5 Results

After the cross-validation process to find the best SVM model, we then ran that model against the 1000 vectors that were initially (randomly) selected as the test corpus. Based on the hand-tagging, we knew that, as with the training set, approximately 10% of the sentences in the test set contained open geospatial relations. After generating feature vectors for each sentence, the sentences were fed to the SVM, which tagged each sentence with a 1 or 0, indicating the presence of an open geospatial relation. Table 12 shows the precision, recall, and f-measure statistics for the test corpus. The relation-extraction SVM used the same equations for precision, recall, and f-measure used throughout this research, defined in Chapter 2.

The table demonstrates that the results we obtained line up well with the results
previously obtained by research in this area, also demonstrating some of the same challenges. Our precision in detection of the relations is very good, indicating that for sentences that are identified by the SVM as containing a geospatial relation, a high percentage actually do contain one. However, the recall for the system is rather low. Figure 25 shows the details of the relationship extraction from a selection of the Wikipedia articles. The count of actual relations in the article follows the article name in parentheses in the figure. Note that in all cases, the precision is up around the 80% mark, but the recall is significantly lower.

The system misses more than 50% of the sentences that it should identify as containing a relation. This is primarily due to the size and variety in the negative set for the relations. That is, the task of the SVM is to divide all the sentences into two classes, either with or without a geospatial relation. The set of sentences without relations, however, represents a large, heterogenous space that defies easy feature selection and classification with any method, including feature modeling with an SVM, yielding the low recall numbers. The set of sentences that do contain the geospatial relations is (comparatively) easier to classify, yielding the higher precision numbers. Note also that the performance generally goes up as the number of relations in the article itself goes down. From Table 3 in Chapter 4, we can see that the relation count in an article is closely related to the word count in the article, yielding the

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.812</td>
<td>0.418</td>
<td>0.549</td>
</tr>
</tbody>
</table>
SRV.

jigure TW” hetailed velationship vesults for a wubset of –ikipedia erticles

conclusion that the performance of the relation identification process is somewhat related to the size of the article.

7.6 Discussion

In this phase of the research, we primarily ran into two challenges. The first revolved around the fact that the performance of the NE-based parts of this stage rely heavily on the NE recognition and resolution process in the first two stages. The second challenge arose from the feature selection for the SVM.

To generate the absolute and relative geospatial NE features for the SVM, we wrote code to replace all instances of geospatial NEs in each sentence we considered.
with an NE marker for the SVM feature vector builder. This processing relied heavily on the correct identification of the geospatial NEs from the first two stages of the Geografikos package. While the first two stages did perform at an f-measure of 82%, they still missed effectively 18% of the geospatial NEs, which directly impacted the performance of this stage. Our research into feature selection demonstrated that the relative position of the geospatial NE in a relation was a very important feature, as would be expected. Therefore, an increase in the NER performance of the first two stages in the Geografikos package would directly impact the performance of this stage, for the better.

We also spent time optimizing the feature set for the SVM. We tried different combinations of the features described earlier in this chapter, settling on the final list enumerated earlier in this chapter. Table 13 shows the combinations we considered and the resulting f-measure. As with the NER SVM, we optimized the $C$ and $\gamma$ parameters for each feature combination. We separate each set of features into a shorthand class, defined below the table. Note that RPP and RCF are subsets of RPS, and that RNP is a subset of the RPH feature set.

We found the the best performance came from a combination of the absolute positions of the NEs and parts of speech, combined with the relative locations of the NEs and parts of speech. We further narrowed this to the relative positions of just the prepositions, and added the relative positions of the noun phrases to form the feature set with the best performance. We found that adding in the complete relative parts of speech and complete relative significant phrase positions resulted in fairly extreme overfitting of the training data, resulting in severely reduced test
performance. Overall, we feel that we found a feature set that is fairly simple to
generate for a sentence, which resulted in good performance compared to current
relation extraction systems.
8 Conclusion

Leveraging previous research on applying SVMs to NER, and previously published techniques for geospatial reference disambiguation, we explored using an SVM for geospatial NER, and a new algorithm and data structure for disambiguating and geocoding extracted geospatial NEs to construct geospatial entities. We also explored the use of this NE information, along with sentence part of speech and significant phrase information for geospatial relation extraction. Our research included the development of the Geografikos software package. In this chapter, we summarize the results from all three phases of this research, with a holistic discussion of the performance, challenges, and an evaluation of the success criteria. We conclude with a discussion of our research contributions, and cover possible future work based on this research.

In the first phase of our work, we reviewed current research and challenges to the task of NE recognition in general, how those techniques apply to the extraction of geospatial NEs or place names specifically, and the current set of research into applying NER to Wikipedia. We gave an overview of NER, and the features normally selected for performing NER with a variety of techniques, along with the performance measures for NER, and NER challenges. We also covered the basic mathematical theory behind SVMs, performance measures, SVM challenges, and the specifics in applying SVMs to NER. From this basis, we proposed and tested a set of simple features for training an SVM to perform high-recall extraction of geospatial NEs, while maintaining precision at acceptable levels. Using a training corpus selected
from the Geonames database, the Reuters corpus, the CoNLL 2003, 2004 and 2005 datasets, and the SCOWL wordlist, we trained an SVM based on the selected feature set. We tested the SVM on a corpus of 90 Wikipedia articles about battles and wars, achieving an overall precision of 51.0% and a recall of 99.8%. This SVM also provided a set of candidate NEs to the resolution process in the second phase. We referred to them as “candidate” NEs until we have a resolved set of (latitude, longitude) coordinates for each NE.

In the second phase of our work, based on current research into geospatial entity position resolution, we took the candidate NE list from the NER SVM, and applied a novel context-based resolution algorithm and data structure to the information to geocode each location. We used the geospatial NE, the geocoded spatial location, and an assumed point geometry to construct a geospatial entity for each geospatial NE. We gave an overview of geospatial resolution, and then discussed our procedure and results. Utilizing the Google Geocoder service, and a novel location tree data structure, we applied an algorithm and simple ruleset to each candidate NE to resolve it to a single coordinate pair. For those place names that were ambiguous, we demonstrated that our data structure and algorithm used the sentence and article context of the entity to resolve it to the correct entity location. Building on the high-recall SVM NER process from the previous phase, our resolution process resulted in an overall precision of 86.9%, and an overall recall of 78.9% for the recognition, resolution, and geocoding of the NEs from our Wikipedia article corpus.

Finally, in the third phase, we covered our approach to identifying relations between the geospatial NEs resolved in the first two phases and other significant phrases
within the same sentence. We gave an overview and summary of the current research and performance results in the area of relation extraction. We covered the feature selection process and training of an SVM for performing the relation identification, and our results in this area. Using a simple set of features based on the absolute and relative positions of the NEs and parts of speech in the sentences, along with information about the relative position of significant phrases, the SVM identified open geospatial relations with a precision of 81.2% and a recall of 41.8%, an overall f-measure of 54.9%. This demonstrated an increase over current published performance in this area of 3%, and demonstrated very high precision.

8.1 Known Issues and Solutions

Research-related issues aside, the primary software engineering issue with the Geografikos package at this point is that of performance. Currently, it takes between 30 seconds and 3 minutes to process a single page, depending on the length of the page text. We did perform significant optimization on the database to reduce the page query time, but the system itself has a number of bottlenecks that could be addressed to speed up the processing time. A significant bottleneck exists in the interaction with the Google Geocoder service. Round trip for each page requires approximately 500 to 1500 milliseconds. Due to request per second and request per day limitations on the use of the free Google Geocoder, the requests cannot be sent faster. With a different geocoder, or a paid subscription to Google Geocoder, the requests could be multi-threaded, reducing the overall time to complete all the requests. This would,
however, require that the location tree data structure and the resolution algorithm be adapted to be thread-safe. The request to the geocoder could also be cached to increase performance. Furthermore, simply running a profiler on the Geografikos process would indicate the largest bottlenecks in processing time, and allow the code to be refactored for greater throughput. This optimization will be required if we wish to process even a fraction of the complete articles in Wikipedia.

Currently, the NER process using the SVM in Geografikos does provide high recall, almost 100%, which is in accordance with our stated research goals. However, the precision is only 51%. An increase in this precision would effect an increase in the overall performance in both the NE resolution phase, and in the relation extraction phase. This precision increase could be accomplished in a number of ways. First, the training corpus for the SVM could be examined more closely, and detailed metrics created, to provide a less noisy set of data for training. Second, principal component analysis could be performed on the feature set to determine which features are the most important to the process, and investigate weighting these features. While we performed a limited analysis on the feature weighting, a more in depth analysis might reveal performance increases. Finally, the NER component of the Geografikos package could be completely replaced with a different NER engine entirely.

The geocoding process itself has three issues. The first is a noticeable bias towards US and North American locations. Google Geocoder will find a location somewhere in the US for almost any string sent to it. This resulted in a somewhat noisy geocoding process, and required extra code to be written to handle the unnecessary locations. Second, the geocoder does not handle duplicate place names well.
Third, the geocoder and the Geografikos package treat all locations as having a point geometry, ignoring that rivers are linear, and cities and countries are polygons. While Google Geocoder will return multiple placemarks for an ambiguous location, it could not always be counted on to return results outside the US for an ambiguous place name like “Cambridge”. This problem has a number of potential solutions. First, multiple requests could be made to the geocoder for each NE, with biasing of the request to a particular country. Google Geocoder allows the request to be limited to a specific country code. Alternately, the geocoder request could be limited to a bounding box, where the geocoder would return all locations that match the query within that bounding box. A different geocoding service could also be utilized. We tested the TinyGeocoder service\textsuperscript{25}, but found that Google performed better on ambiguous queries. We could also use the Geonames database and construct our own geocoder, with a web service interface backed by a full-text search engine and the Geonames or other data. It may also be possible to build a number of separate modules based on our resolution process, where each module handles geocoding in a specific region of the world. Each NE could be run through all modules, and a weighting or voting system could be used to select the correct location. Finally, to account for geospatial entities that do not actually have a point geometric shape, we could build a second database of these shapes (from sources such as the TIGER line data), and perform lookups into that database by the place name attribute of the geospatial entities constructed by the Geografikos package.

\textsuperscript{25}http://tinygeocoder.com/
cation phase of our research, the recall was much lower. Our overall f-measure for the relation identification was acceptable based on our evaluation criteria, but this is certainly an issue. We theorize that this is primarily due to the fact that it is much easier (relatively) to determine the features that do define a sentence with an open geospatial relation. That is, the set of sentences that do not contain geospatial relations is very heterogenous, and a large enough set to defy easy classification. By comparison, it is much easier to determine what does constitute a geospatial relation.

The SVM has difficulty in defining what is not a relation, so it tends to default to defining any ambiguous sentence as not containing a relation, yielding the low recall numbers. It may also be the case that, though we performed cross-validation, our training set is not varied enough to provide examples of all the feature combinations that define geospatial relations. We could address this issue with a much larger training set, though the hand-tagging of that much data presents its own challenges. Also, because the relation identification was directly dependent on the NE extraction phases, increasing the NER performance would directly result in a relation identification performance gain, as more place names in the text would be correctly marked and used in the generation of feature vectors and processing to identify the geospatial relations.

8.2 Evaluation of Success Criteria

Based on the objective success criteria we laid out in Chapter 1, we achieved all of our research goals in this work. Based on the background research in geospatial
NER, we targeted a recall of 100% for the NER process, with a precision of at least 50%, resulting in an f-measure of at least 65%. After training and testing, our SVM reached a precision of 51.0% and a recall of 99.8%, an f-measure of 67.5%. We do not compare these results directly with current research, but instead compared the results of this and the NE resolution stage together, because the high recall is the important metric for this phase, returning close to 100% of the geospatial NEs from the article text, despite the noise introduced by the lower precision.

For the NE disambiguation and geocoding phase of our research, we targeted an overall f-measure of 80% based on current research and techniques in this area. We expected a reduction in recall from the first phase, but a significant increase in the precision as the noise from the SVM was filtered from the candidate NEs. Our system, after performing context-based disambiguation and geocoding of the candidate geospatial NEs, resulted in a precision of 86.9% and recall of 78.9%, an overall f-measure of 82.0%. While our f-measure is lower than the 94.5% on LOC entities demonstrated in 2008 by Dakka and Cucerzan (in [22]), we have not just identified all the geospatial entities in the article text as in their work, but have actually resolved each entity to a coordinate pair, giving much more overall utility in geographic information retrieval. We feel that our f-measure of 82.0% provides an excellent baseline for future research in this area.

Finally, in the third phase of our research, we targeted an f-measure of 50%. Because relation identification and extraction is a relatively new area of research, current published techniques demonstrate f-measure between 40% and 50% using a variety of techniques, from surface feature parsing (as we presented in our work), to
deep sentence parsing for dependencies. Using our simple surface-based feature set along with the NE information from the previous, we achieved precision of 81.2% and recall of 41.8%, with an overall f-measure of 54.9%, well above our target criteria. The precision numbers for our approach are very promising, but the recall is somewhat more of challenge, inviting further research and refinement. Despite that, we have demonstrated an innovative approach to the extraction of relations between geospatial NEs and other significant phrases in the sentences.

8.3 Contributions

Our research work in this thesis has contributed to the state of the art in both geospatial and computer science in a number of ways:

1. In combination with our high-recall SVM for NER, a novel data structure and context-based algorithm with a simple ruleset for disambiguating and geocoding geospatial NEs. This algorithm and dataset provide context information from the surrounding text for the geocoding of a geospatial NE, leading to an increase in performance for geospatial NE resolution from free text. This can provide machine readable data to geographic information retrieval processes and visualization.

2. A simple feature set for geospatial relation identification in sentences, based on NEs within the sentence, and various part of speech and significant phrase relative locations within the sentence. This feature set can be generated with an NER engine and a simple NLP toolkit, like the LingPipe software used in this research, and requires no deep parsing of the sentence or custom toolkits.
3. We published a paper titled *Mining Wikipedia Article Clusters for Geospatial Entities and Relationships* based on this research work in the proceedings of the Association for the Advancement of Artificial Intelligence Spring Symposia [69]. We also gave a talk at the conference based on the paper.

4. We plan to make the Geografikos software available as an open source software package, available for download.

5. We also plan to provide the hand-tagged corpus of Wikipedia pages for download, for future geospatial NE research.

### 8.4 Future Work

The foundation techniques and methods presented in this paper provide the potential for future work in a number of areas. Based on the three separate phases of this research, future work could proceed in many directions, from geospatial information retrieval (GIR) and geovisualization to further work in using machine learning techniques for NER and relation identification and extraction. Further research may also proceed in the disambiguation and geocoding of extracted geospatial NEs. Note that, because the Geografikos package is not fundamentally tied to Wikipedia, it can be used against any free text corpus, so many of the research suggestions here that apply to Wikipedia could also be used with other corpuses.

Increasing the efficiency of the Geografikos package itself can provide a number of research opportunities. A speedup in the processing time, combined with access to a more powerful server cluster would allow for location extraction and resolution from a significant fraction of the English Wikipedia. With a database of locations
linked to Wikipedia articles, a number of research paths open up in the area of GIR. First, Wikipedia searches could be filtered by geographic area. The user could specify a search term and an area and the search could be performed in two steps. The first would be a standard free text search on Wikipedia, for articles matching the search term. That list could then be further filtered to those articles that match the geographic criteria. Reversing this paradigm, the user could provide a location or bounding area, and the system could return all Wikipedia articles with locations that fall into that bounding area. Furthermore, the database of locations could enable the geovisualization of Wikipedia articles, individually or in a cluster, on a map interface. A user could see a map of all the locations in the article or articles, which could reveal implicit information and relationships not apparent in the article text.

The development of GIR visualization tools can take many forms, all driven from a database of locations and free text articles.

With the rise of smartphones and mobile applications, location context for search is also very useful area for research. Imagine a tourist with a smartphone searching for all Wikipedia articles that have location references within a 10 mile radius, to find points of interest. With the GPS location from the phone, and the locations database generated by the Geografikos software, this would be a straightforward query to perform.

Considering the internal performance of the software, the use of different NER engines for the first phase of processing should also be investigated. With increased performance in the NER phase, both the disambiguation and geocoding, and the relation extraction performance would be positively impacted. A number of open
and commercial NER packages exist, some with very good performance in NER, especially for LOC, PER, and ORG entities. While we have contributed to the area of geospatial NER using SVMs, research in this area is ongoing, and there are certainly other approaches to geospatial NER that can provide equivalent, if not better results in the NE recognition area, the first phase of the processing. The only consistent requirement in this area is that the NER process have a high recall, to provide a good set of candidate NEs to the disambiguation and geocoding process.

Our disambiguation process and location tree data structure is directly impacted by the performance of the geocoder. With a more efficient geocoder, we could perform more queries to the geocoder for each location we wished to geocode, providing better information to the context algorithm and ruleset. Along these same lines, the sentence or paragraph surrounding the NE under consideration could be parsed into a complete dependency tree for the sentence, allowing dependency information and non-local dependencies between the NE and other words in the sentence to be used for the disambiguation process. Culotta and Sorensen provide more information on dependency trees in [20].

From a Wikipedia-specific standpoint, there is a significant amount of information present in the structure of Wikipedia that can provide context and disambiguation information. Though we used Wikipedia as our corpus, we focused on general free text geospatial NER and resolution, but there is potential for forking the Geografikos software and applying it more specifically to the structure of the English Wikipedia. Many of the geospatial NEs in the first part of each article we considered are actually
links to the Wikipedia page for that location. Combined with the GeoHack project\textsuperscript{26}, which applies coordinates to Wikipedia pages that have a single locations (such as article pages for cities, states, etc.), the coordinates for that NE can be determined without geocoding or requiring any disambiguation. This information can be fed back to the general disambiguation and context algorithm, and should provide increases in performance.

Further research should also be performed on geospatial relation extraction, based on the work in this thesis. Our research in this area simply identified sentences that contained geospatial relations between a geospatial NE and another significant phrase, noun, or verb in the sentence. Further research should be performed to increase the recall for relation extraction, either through a better training corpus, or through a different set of features. This is an area that may also benefit from the generation of deep dependency trees for the sentences. Once the precision and recall for the process has been increased, research can also be done in actually extracting the relations and storing them in a structured format, such as the Web Ontology Language (OWL)\textsuperscript{27}, which is designed to capture precisely this kind of information.

TALK ABOUT EXTENDING THE GIR aspects with geometric databases

\textsuperscript{26}http://stable.toolserver.org/geohack/
\textsuperscript{27}http://www.w3.org/TR/owl-features/
References


Appendix A: Software Packages and Versions

- Alias-I LingPipe 3.7.0 – http://alias-i.com/lingpipe/
- GraphViz 2.2.2 – http://www.graphviz.org/
- Java 1.5, 1.6 – http://java.com/en/
- JRuby 1.6 – http://jruby.codehaus.org/
- l2p 1.1.1 – http://redsymbol.net/software/l2p/
- LibSVM 2.88 – http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Skim 1.2.1 – http://skim-app.sourceforge.net/
- TextMate 1.5.8 – http://macromates.com/
- xml2sql 0.5 – http://meta.wikimedia.org/wiki.Xml2sql
- Ruby gems (software packages for Ruby and JRuby), available at gems.rubyforge.org and through the Ruby and JRuby gem command
  - activerecord 2.2.2
  - activerecord-jdbc-adapter 0.9
  - activerecord-jdbcmysql-adapter 0.9
  - activesupport 2.2.2
  - google-geocode 1.2.1
  - hpricot 0.6.164
  - jdbc-mysql 5.0.4
  - rake 0.8.4
  - rspec 1.2.2
  - rspec-rails 1.1.12

All development and testing was performed on a MacBook, with a 2.4 GHz dual core processor, 4GB memory, and OSX 10.5.