

## ABSTRACT

Title of Dissertation:     MULTIFACETED GEOTAGGING  
                                  FOR STREAMING NEWS

Michael David Lieberman, Doctor of Philosophy, 2012

Dissertation directed by: Professor Hanan Samet  
                                  Department of Computer Science

News sources on the Web generate constant streams of information, describing the events that shape our world. In particular, geography plays a key role in the news, and understanding the geographic information present in news allows for its useful spatial browsing and retrieval. This process of understanding is called *geotagging*, and involves first finding in the document all textual references to geographic locations, known as *toponyms*, and second, assigning the correct lat/long values to each toponym, steps which are termed *toponym recognition* and *toponym resolution*, respectively. These steps are difficult due to ambiguities in natural language: some toponyms share names with non-location entities, and further, a given toponym can have many location interpretations. Removing these ambiguities is crucial for successful geotagging.

To this end, geotagging methods are described which were developed for streaming news. First, a spatio-textual search engine named STEWARD, and an interactive map-based news browsing system named NewsStand are described, which feature geotaggers as central components, and served as motivating systems and experimental testbeds for developing geotagging methods. Next, a geotagging methodology is presented that follows

a multifaceted approach involving a variety of techniques. First, a multifaceted toponym recognition process is described that uses both rule-based and machine learning-based methods to ensure high toponym recall. Next, various forms of toponym resolution evidence are explored. One such type of evidence is lists of toponyms, termed *comma groups*, whose toponyms share a common thread in their geographic properties that enables correct resolution. In addition to explicit evidence, authors take advantage of the implicit geographic knowledge of their audiences. Understanding the local places known by an audience, termed its *local lexicon*, affords great performance gains when geotagging articles from local newspapers, which account for the vast majority of news on the Web. Finally, considering windows of text of varying size around each toponym, termed *adaptive context*, allows for a tradeoff between geotagging execution speed and toponym resolution accuracy. Extensive experimental evaluations of all the above methods, using existing and two newly-created, large corpora of streaming news, show great performance gains over several competing prominent geotagging methods.

MULTIFACETED GEOTAGGING FOR STREAMING NEWS

by

Michael David Lieberman

Dissertation submitted to the Faculty of the Graduate School of the  
University of Maryland, College Park in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
2012

Advisory Committee:

Professor Hanan Samet, Chair  
Professor Larry Davis  
Professor William Rand  
Professor Paul Torrens  
Professor Amitabh Varshney  
Professor Amy Weinberg

© Copyright by  
Michael David Lieberman  
2012

## Preface

The material in this dissertation is based in part on the following publications:

### Chapter 2:

- H. Samet, M. D. Lieberman, J. Sankaranarayanan, and J. Sperling. STEWARD: Demo of spatio-textual extraction on the web aiding the retrieval of documents. In *DG.O'07: Proceedings of the 8th Annual International Conference on Digital Government Research, Bridging Disciplines & Domains*, pages 300–301, Philadelphia, PA, May 2007.
- M. D. Lieberman, H. Samet, J. Sankaranarayanan, and J. Sperling. STEWARD: Architecture of a spatio-textual search engine. In *GIS'07: Proceedings of the 15th ACM International Symposium on Geographic Information Systems*, pages 186–193, Seattle, WA, Nov. 2007.
- M. D. Lieberman, J. Sankaranarayanan, H. Samet, and J. Sperling. Augmenting spatio-textual search with an infectious disease ontology. In *IIMAS'08: Proceedings of the Workshop on Information Integration Methods, Architectures, and Systems*, pages 266–269, Cancún, Mexico, Apr. 2008.

### Chapter 3:

- B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A new view on news. In *GIS'08: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 144–153, Irvine, CA, Nov. 2008.

### Chapter 4:

- M. D. Lieberman and H. Samet. Multifaceted toponym recognition for streaming news. In *SIGIR'11: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 843–852, Beijing, China, July 2011.

Chapter 5:

- M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *GIR'10: Proceedings of the 6th Workshop on Geographic Information Retrieval*, Zurich, Switzerland, Feb. 2010.

Chapter 6:

- M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging with local lexicons to build indexes for textually-specified spatial data. In *ICDE'10: Proceedings of the 26th International Conference on Data Engineering*, pages 201–212, Long Beach, CA, Mar. 2010.

Chapter 7:

- M. D. Lieberman and H. Samet. Adaptive context features for toponym resolution in streaming news. In *SIGIR'12: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 731–740, Portland, OR, Aug. 2012.

For Mom

## Acknowledgements

I could not have completed the work leading up to this dissertation without the help of countless friends and family. It takes a village to raise a child, and this child is no exception.

First and foremost, I thank Prof. Hanan Samet, both for convincing me to join the grad school at Maryland, and for being an excellent advisor. I was impressed by his research outlook and philosophy, and his willingness to donate his time and effort to my work. He was also very communicative, thorough, and frank with his criticism—especially related to geotagging errors—which at first I resisted, but later came to appreciate. I also greatly appreciate his enthusiasm for my work. Without his help and guidance I would not have reached this point. I also thank the other members of my proposal and dissertation committees, including Profs. Larry Davis, David Mount, William Rand, Paul Torrens, Amitabh Varshney, and Amy Weinberg. Their questions, comments, and challenges were illuminating and very helpful in clarifying important aspects of my work. I also thank Jon Sperling, who was instrumental in funding the project that eventually became STEWARD, thus enabling my graduate education, and also for his tireless enthusiasm and endless submissions of geotagging error reports.

Next, I thank Jagan Sankaranarayanan for being an excellent mentor and a good friend. He taught me innumerable lessons about how to be a good grad student and researcher, and by emulating his example I've grown immensely. I will always treasure our many daytime and frequently late night discussions. Also, crucially, he introduced me to the sport of squash, for which I am forever grateful. I also thank all of my current and former labmates including but not limited to Marco Adelfio, Brendan Fruin, Yi Nutanong, Daniele



Panozzo, Gianluca Quercini, and Benjamin Teitler. They made those long nights in AV Williams bearable, and I appreciate those discussions we had over a whiteboard, or over a few beers. All of them have also touched various components of NewsStand and our group's other systems, and without their help I would not have had such a wonderful testbed for geotagging. I am indebted to you always.

Countless other students and friends have helped me in many ways. I thank Lidan Wang for her introduction and pointers to techniques in information retrieval, machine translation, and general natural language processing. Also, I thank Dave Shaw for our many useful conversations about machine learning. Aleks Aris introduced me to the concepts of information visualization and further was a very good friend. I had many late-night philosophical discussions with him that I will always remember vividly. I thank Poorvi Patel for being such an amazing friend for so long. I also thank Beiyu, Hyunyoung, Prahalad, Qi, Sunny, Brett and Dave and the pack, Gus, Hang, Kevin, Max, Prakash, Tony, Yan, and many, many others for their friendship and support. I especially thank Rodica Tuduce, whose endless love and encouragement in bad times and good spurred me onward. I will always owe her a huge debt of gratitude.

Throughout grad school I've played squash, a sport that demands not only exceptional fitness and coordination, but also mental toughness, and through squash I was introduced to a host of wonderful, talented athletes who served as role models both on and off the court. The list is too long to capture here, but includes Anand B., Anand V., Abbad, Andy, Christian, Dave, Dot, Jagan, Jane, Kiran, Mike S., Paul, Prasanth, Ruud, Shyam, Taimur, and many, many others. Their athletic prowess and research excellence were great inspirations to me and drove me to improve both my game and my research.

My extended family were a constant presence with me and I am indebted to them always for their love and support. I thank Uncle Norman, Aunt Diane, Joel, Ross, Cynthia, Seth, Aunt Cynnie, Uncle Hank, Edith, Abby, Dani, and especially Grandma Sarah. Their enthusiasm for my work and their boundless love served as constant inspirations for me.

For the countless others that helped me along the way but I've forgotten to include here, please accept my sincerest apologies and deepest gratitude.

Finally, I thank my mother, Ellen, who taught me never to give up and to “keep writing, keep writing, keep writing”, my father, Alvin, who constantly told me to “keep going for that prize!”, and my sister, Jane, all of whom were endlessly loving and supportive. I will love and cherish you always.

## Table of Contents

List of Figures	xii
List of Tables	xiv
1 Introduction	1
1.1 Toponym Ambiguity	7
1.2 STEWARD and NewsStand	15
1.3 Methodology	16
1.4 Related Work	22
1.5 Summary of Contributions	25
1.6 Dissertation Outline	27
2 STEWARD: A Spatio-Textual Document Search Engine	29
2.1 Architecture	31
2.2 Document Standardizer	32
2.3 Geotagger	33
2.3.1 Feature Vector Extraction	33
2.3.2 Feature Record Assignment	35
2.3.3 Disambiguation via Semantic Analysis	36
2.3.4 Geographic Focus Determination	37
2.4 Web Interface	38
2.5 Application: Disease Monitoring	47
2.6 Open Problems	48
2.7 Summary	51
3 NewsStand: Map-Based Exploration of Streaming News	52
3.1 Architecture	56
3.2 Processing Modules	60
3.2.1 RSS Grabber	61
3.2.2 Downloader	64
3.2.3 Cleaner	67
3.2.4 Clusterer	70
3.2.5 Topic Classifier	73
3.2.6 Geotagger	74
3.2.7 Disease Finder / People Finder	75
3.2.8 Media Extractor	78
3.3 Pipe Server	82
3.3.1 Communication Protocol	84
3.3.2 Checking Module Status	89
3.4 Database Design	89

3.4.1	Documents	92
3.4.2	Geotagging	93
3.4.3	Clusters	94
3.4.4	Cache Database	95
3.5	Web Interface	96
3.6	Database Queries	101
3.6.1	Top Stories Mode	104
3.6.2	Map Mode	106
3.6.3	Single-Cluster Queries	108
3.7	Experiments	111
3.7.1	Data Collection	111
3.7.2	Data Content	113
3.7.3	Data Size	114
3.7.4	Processing Time	115
3.7.5	Query Performance	117
3.8	Open Problems	120
3.9	Summary	123
4	Multifaceted Toponym Recognition	124
4.1	Finding Toponyms	129
4.1.1	Tokenization	129
4.1.2	Entity Tables	130
4.1.3	Entity Dictionary	131
4.1.4	Proper Nouns	133
4.1.5	Named-Entity Recognition	134
4.2	Filtering Toponyms	139
4.2.1	Toponym Refactoring	139
4.2.2	Active Verbs	141
4.2.3	Noun Adjuncts	141
4.2.4	Type Propagation	142
4.2.5	Lat/Long Assignment	143
4.3	Evaluation	144
4.3.1	Existing Corpora	145
4.3.2	Toponym Statistics	147
4.3.3	CLUST: A New Corpus of Streaming News	149
4.3.4	Toponym Accuracy	152
4.3.5	Streaming Evaluation	156
4.4	Open Problems	157
4.5	Summary	159
5	Comma Groups	160
5.1	Comma Group Recognition	162
5.2	Comma Group Resolution	165
5.2.1	Prominence	167
5.2.2	Proximity	170

5.2.3	Sibling . . . . .	172
5.3	Usage Evaluation . . . . .	174
5.4	Open Problems . . . . .	177
5.5	Summary . . . . .	178
6	Local Lexicons . . . . .	179
6.1	Inferring Local Lexicons . . . . .	182
6.2	Resolution with Local Lexicons . . . . .	188
6.2.1	$\mathcal{H}_1$ : Dateline . . . . .	191
6.2.2	$\mathcal{H}_2$ : Relative Geography . . . . .	192
6.2.3	$\mathcal{H}_3$ : Comma Group . . . . .	192
6.2.4	$\mathcal{H}_4$ : Object/Container . . . . .	193
6.2.5	$\mathcal{H}_5$ : Local Lexicon . . . . .	193
6.2.6	$\mathcal{H}_6$ : Global Lexicon . . . . .	193
6.2.7	Postprocessing . . . . .	194
6.3	Experiments . . . . .	195
6.3.1	Evaluation Measures . . . . .	195
6.3.2	Datasets . . . . .	196
6.3.3	Inferring Local Lexicons . . . . .	198
6.3.4	Toponym Recognition . . . . .	202
6.3.5	Toponym Resolution . . . . .	205
6.3.6	Heuristic Usage . . . . .	208
6.4	Open Problems . . . . .	210
6.5	Summary . . . . .	211
7	Adaptive Context Features . . . . .	212
7.1	Geotagging Framework . . . . .	215
7.1.1	Toponym Recognition . . . . .	215
7.1.2	Toponym Resolution . . . . .	215
7.1.3	Resolution Features . . . . .	217
7.2	Adaptive Context Features . . . . .	218
7.2.1	Proximity Features . . . . .	220
7.2.2	Sibling Features . . . . .	221
7.2.3	Feature Computation . . . . .	222
7.2.4	Feature Propagation . . . . .	225
7.3	Experiments . . . . .	226
7.3.1	Gazetteer Ambiguity . . . . .	227
7.3.2	Datasets . . . . .	227
7.3.3	Recognition Accuracy . . . . .	230
7.3.4	Resolution Accuracy . . . . .	231
7.3.5	Adaptive Parameters . . . . .	234
7.4	Open Problems . . . . .	236
7.5	Summary . . . . .	236
8	Conclusions and Future Work . . . . .	237

8.1	Clustering Evidence . . . . .	240
8.2	Streaming Lexicon . . . . .	241
8.3	Toponym Corpora . . . . .	242
	Bibliography	244

## List of Figures

1.1	Examples of ambiguous and unusual toponyms in the United States . . . . .	9
1.2	Geo/non-geo ambiguity: Batman (Turkey) . . . . .	11
1.3	Geo/non-geo ambiguity: Obama (Japan) . . . . .	12
1.4	Geo/geo ambiguity: Java, Georgia . . . . .	12
1.5	Geo/geo ambiguity: Vancouver . . . . .	14
1.6	Locations in news articles about the May 2009 swine flu pandemic . . . . .	14
2.1	STEWARD's architecture . . . . .	32
2.2	STEWARD's primary user interface . . . . .	40
2.3	Keyword searching in STEWARD . . . . .	41
2.4	STEWARD's focus mode . . . . .	44
2.5	STEWARD's highlighted copy browsing mode . . . . .	46
2.6	Using STEWARD to find cases of avian influenza . . . . .	49
3.1	NewsStand's architecture . . . . .	57
3.2	Correlation of RSS feed update rate and poll interval . . . . .	64
3.3	Desktop versus mobile versions of a website . . . . .	66
3.4	An image extracted by the media extractor . . . . .	80
3.5	NewsStand's pipe server's protocol messages . . . . .	85
3.6	NewsStand's pipe and module status screen . . . . .	88
3.7	NewsStand's database schema . . . . .	91
3.8	NewsStand's Web interface in top stories mode . . . . .	98
3.9	NewsStand's Web interface in map mode . . . . .	100
3.10	NewsStand's disease and people layers . . . . .	102
3.11	SQL queries used in NewsStand's top stories mode . . . . .	105
3.12	SQL queries used in NewsStand's map mode . . . . .	109
3.13	Queries for single-cluster information in NewsStand . . . . .	110
3.14	Number of articles retrieved by NewsStand over time . . . . .	112
3.15	Types of article content found by NewsStand . . . . .	114
3.16	NewsStand's backend processing time measured per article . . . . .	116
3.17	NewsStand's database query performance . . . . .	119
3.18	NewsStand's Web interface query time by region and zoom level . . . . .	121
4.1	Geographic distribution of toponyms in the LGL and CLUST corpora . . . . .	151
4.2	Toponym recognition performance for CLUST over time . . . . .	158
5.1	Examples of prominence comma groups . . . . .	169
5.2	Examples of proximity comma groups . . . . .	171
5.3	Examples of sibling comma groups . . . . .	173
5.4	Comma group sizes for our test dataset . . . . .	175
6.1	Local lexicon for Columbus, Ohio . . . . .	181

6.2	Traffic hotspots in Washington, DC . . . . .	181
6.3	Local lexicon for the Paris News . . . . .	188
6.4	Local lexicon foci for news sources in the USA . . . . .	200
6.5	Local lexicon inference performance when varying $D_{max}$ . . . . .	200
6.6	Local lexicon inference performance when varying $S_{min}$ . . . . .	203
6.7	Toponym recognition performance . . . . .	204
6.8	Toponym resolution performance . . . . .	209
6.9	Heuristic usage in toponym resolution . . . . .	209
7.1	Extreme toponym ambiguity in a news article . . . . .	213
7.2	An illustration of adaptive feature computation . . . . .	219
7.3	Characterization of gazetteer ambiguity . . . . .	228
7.4	Annotated location types in the ACE, LGL, and CLUST corpora . . . . .	230
7.5	Per-feature importance measures used in the adaptive method . . . . .	234
7.6	Resolution performance when varying adaptive window parameters . . . . .	235



## List of Tables

3.1	Cue words for person entities . . . . .	77
3.2	Sizes of objects in NewsStand’s database . . . . .	115
3.3	NewsStand’s backend processing time measured per module . . . . .	117
4.1	Sample entities from the entity dictionary . . . . .	132
4.2	Entity names modified by refactoring . . . . .	140
4.3	Existing corpora intended for geotagging-related research . . . . .	146
4.4	Toponym counts for sampled news days . . . . .	148
4.5	Comparison of the LGL and CLUST corpora . . . . .	152
4.6	Toponym recognition performance for LGL . . . . .	155
4.7	Toponym recognition performance for CLUST . . . . .	156
5.1	Comma group usage . . . . .	174
5.2	Heuristic precision measurements . . . . .	175
5.3	Heuristic usage measurements . . . . .	176
6.1	Heuristics used for toponym resolution . . . . .	190
6.2	Comparison of the ACE and LGL corpora . . . . .	197
6.3	Correctness measure for local lexicon inference . . . . .	198
6.4	Toponym resolution performance . . . . .	207
7.1	Comparison of the ACE, LGL, and CLUST corpora . . . . .	229
7.2	Toponym recognition performance . . . . .	231
7.3	Adaptive context resolution accuracy against competing methods . . . . .	232
7.4	Adaptive context resolution accuracy using different feature combinations . . . . .	232

## Chapter 1

### Introduction

Today's increasingly informed and connected society demands ever growing volumes of information and news. Thousands of newspapers, and millions of bloggers and tweeters around the world post the latest news updates continuously, and the demand for such data is skyrocketing as people strive to stay up-to-date. Blogs, tweets, and other social media have also expanded the realm of news to include citizen journalism. The rise of the Web has allowed their access from anywhere in the world through publishers' online presence, and has fueled intense competition as evidenced by a swift and sometimes tempestuous information cycle. Also, Web-enabled mobile devices are increasingly common, which expands the requirement for location-based services and other highly local content—news that is relevant to where users are, or the places in which they are interested. Our goal is to collect, analyze, and comprehend this streaming, ever-changing mass of information, to make it easily retrievable and accessible by humans. Specialized techniques are required to achieve this goal.

Importantly, news often has a strong geographic component. News sources often characterize their readership in terms of where their readers live, and include articles describing events that are relevant to geographic locations of interest to their readers. To allow readers to retrieve the news that is geographically relevant to them by executing news retrieval queries with a geographic component, we must first understand the geographic content present in the articles. However, currently, online news sources rarely have articles'

geographic content present in machine-readable form. We could assign humans to hand-annotate each article with their geographic content, but this is not scalable with regard to the amount of information being generated all the time. As a result, we must design algorithms to understand and extract the geographic content from the article's text. This process of understanding is known as *geotagging* of text, and amounts to identifying locations in natural language text, and assigning lat/long values to them. Systems using geotagging have recently flourished and have been implemented and used for a wide variety of textual and media domains, such as Web pages [7, 95, 118, 155], blogs [119, 165], encyclopedia articles [62, 109, 141], tweets [130], spreadsheets [4, 75], the hidden Web [78], news photos, and of most relevance to this work, news articles [22, 30, 42, 80, 81, 120, 129, 143]. The methods in this work were developed in tandem with the STEWARD [78], NewsStand [143], TwitterStand [130], and PhotoStand<sup>1</sup> systems, all of which leverage a geotagger to associate unstructured text documents with the geographic locations mentioned in them, thereby enabling users to explore these document collections visually using map query interfaces. STEWARD was constructed as a system for spatially browsing static collections of documents from the hidden Web, while NewsStand and TwitterStand operate on streaming data, namely streaming news and tweets. In addition, several commercial products for geotagging text are available, including MetaCarta's Geotagger [99], Thomson Reuters's OpenCalais [147], and Yahoo!'s Placemaker [162].

The process of geotagging consists of finding all textual references to geographic locations, known as *toponyms*, and then choosing the correct location interpretation for each toponym (i.e., assigning lat/long values); these two steps are known respectively as *toponym recognition* and *toponym resolution*. As with many other problems involving natural language, the central challenge associated with geotagging lies in linguistic ambiguity, in that toponym recognition and resolution involve resolving several kinds of ambiguity present in location names. In particular, many names of places are also names of other

---

<sup>1</sup><http://photostand.umiacs.umd.edu>

type of entities, called *geo/non-geo ambiguity* (e.g., “Stanley Jordan”, “Bristol Palin”, and “Paris Hilton” are persons, while “Bristol”, “Paris”, and “Jordan” are also toponyms), and many different places have the same name, called *geo/geo ambiguity* (e.g., over 40, 50, and 60 places around the world named “Jordan”, “Bristol”, and “Paris”, respectively). Resolving these ambiguities requires a good understanding of the document’s content to make informed decisions as to which words are toponyms, as well as which of the many possible interpretations for a toponym is correct. In addition, the particular text domain may involve other ambiguities that pose additional challenges for geotagging. For example, geotagging blogs may be more challenging than geotagging newswire simply because blog text may have more misspellings and grammar mistakes. Likewise, geotagging tweets would be even more difficult, due to the size limit of 140 characters and potential total disregard of linguistic norms. Section 1.1 elaborates on these types of ambiguity and contains several prominent examples of toponym ambiguity found in news articles. While humans are generally good at resolving these ambiguities, replicating or exceeding their performance with a fully-automated algorithm turns out to be a challenging task.

Put another way, geotagging can be considered as enabling the spatial indexing of unstructured or semistructured text. This spatial indexing provides a way to execute both feature-based queries (“Where is  $X$  happening?”) and location-based queries (“What is happening at location  $Y$ ?”) [12]. Executing these queries involves the efficient retrieval of spatial data, which in turn requires the construction of appropriate spatial indexes, some examples of which are R-trees and quadtrees [127]. These indexes are relatively easy to construct when such data is readily available. However, this is not the case when the data consists of unstructured text, where the spatial data is really words of text that can be (but are not required to be) interpreted as the names of locations. In other words, the spatial data is in the form of toponyms rather than geometry, which implies the ambiguity mentioned earlier. This ambiguity has an advantage in that from a geometric standpoint, the textual specification captures both the point and spatial extent interpretations of the data. On the

other hand, the disadvantage is that we are not always sure which of many instances of geographic locations with the same name is meant, and geotagging is meant to overcome this disadvantage.

As an aside, note that any representation chosen for location interpretations will necessarily have some intrinsic spatial ambiguity. For example, when geotagging a particular toponym corresponding to a city, the centroid of the geographic region covered by the city may be used as the interpretation. This level of precision may or may not be useful, given the application. For reading news about the city, a city-level resolution is probably enough, but for geotagging individual buildings, road intersections, or addresses, a finer resolution is needed. Thus, technically, geotagging's task is to reduce the ambiguity present in toponyms to an acceptable level, dependent on the application in which it is used.

The geotagging process selects location interpretations for each toponym from a *gazetteer*, a database of locations and associated metadata. A gazetteer contains at a minimum the latitude and longitude for each toponym, but usually includes additional information that is useful for geotagging. For example, our gazetteer contains alternate names for each location in various languages, population data, and hierarchical information indicating the country and administrative divisions that contain the location. Note that the choice of gazetteer plays a large role in the quality of geotagging, as well as the geotagging method's apparent performance under evaluation. Since the geotagger can only choose interpretations from the gazetteer, a small gazetteer limits the possible interpretations which must be considered in the toponym resolution procedure, which makes the task easier. On the other hand, a small gazetteer excessively penalizes the effectiveness of the entire geotagging procedure, if it is not complete enough. For example, the Web-a-Where system [7] uses a gazetteer containing 40,000 locations and 70,000 location names. The small size of Web-a-Where's gazetteer effectively reduces the ambiguity present in the toponyms that they can recognize and resolve, which means that Web-a-Where misses many toponyms present in real world articles. In addition, any evaluation of Web-a-Where's geotagging procedure will

be made difficult because it must acknowledge this gazetteer limitation. On the other hand, our methods use a gazetteer based on the GeoNames database [43], which is a comprehensive collection of geographic data from over 100 sources, including the GEONet Names Server (GNS) [105] and Geographic Names Information System (GNIS) [152]. Our gazetteer contains over 8 million geographic locations, and over 10 million location names from around the world. This large gazetteer makes the geotagging process more difficult, but at the same time more flexible due to its ability to decide among many more interpretations.

Our primary focus in this dissertation concerns the geotagging of news articles. In particular, we advocate a combined, multifaceted approach to geotagging these articles using a variety of techniques and many different rules and heuristics, a philosophy which is important in the context of the working systems for which our algorithms were developed. These rules include both domain-independent and domain-specific heuristics, and are based on the content and structure of typical news articles. By understanding this structure, we take advantage of rich contextual clues to effect improved geotagging. One such contextual geographic clue is the article's *dateline*. If present, the dateline appears at the beginning of the news article, and indicates when the article was written, usually not long after the described news event. It can also include geographic information, namely where the author wrote the article or where it was submitted for publication. Because it appears prominently at the beginning of the article, it establishes a geographic context for the article as a whole. A location present in the dateline tends to correspond to the principal or most important location in the article. More generally, news articles are often written in an *inverted pyramid* [131] style, with the most important details (i.e., who, what, where, when, why, how) appearing early in the article, often in the first one or two sentences. Since geography is a major part of many news articles, important geographic details are also likely to appear early in the article. Further details are added in subsequent sentences, in decreasing order of importance. By placing salient details first, the author establishes a global framework for understanding the remainder of the story, including the article's geographic focus.

Other contextual clues serve to establish and specify geographic relationships between specific toponyms, usually nearby in the article’s text. One common relationship found in news articles is that of a location paired with its container, termed an *object/container* form, as in “[College Park], [MD]” or “[College Park] in [Maryland]”. Furthermore, certain *cue words* help the reader determine that a given word or phrase is a toponym, and may also provide assistance in resolving the toponym. For example, a mention of “Franklin County” can be recognized as a toponym by its “County” suffix, and further constrains the set of possible resolutions to only those locations that are counties. Other cue words include direction-based language, such as “[Iskandariyah], 30 miles south of [Baghdad]”, which allows for recognition of both “Iskandariyah” (the Arabic name for “Alexandria”) and “Baghdad” as toponyms, and furthermore establishes a spatial relationship between the two toponyms that will aid in resolving them. We also look at nearby terms for evidence that certain terms are not toponyms and instead are of a different nature. Some examples include “Mr.”, indicating a person, and “University of”, indicating a school (organization). An outline of our methodology which takes advantage of these and other contextual clues is presented in Section 1.3.

The rest of this chapter has the following organization. First, to illustrate the difficulties of performing effective geotagging, we present some examples of toponyms with significant ambiguity, and show real-world news articles prominently featuring toponym ambiguity (Section 1.1). Next, we provide an overview of STEWARD and NewsStand, two systems using geotagging as a central component, that served as motivation for our research (Section 1.2). We then describe the methodology used in our toponym recognition and resolution methods (Section 1.3). After, we provide pointers to related work in geotagging (Section 1.4). We conclude the chapter with a summary of this dissertation’s contributions (Section 1.5) and an outline of the remainder of the dissertation (Section 1.6).

## 1.1 Toponym Ambiguity

As mentioned earlier, removing ambiguity in toponyms is the goal of geotagging. We consider the following forms of ambiguity in toponyms:

1. *Geo/non-geo*. Toponyms may share names with non-location entities.
2. *Geo/geo*. Toponyms may refer to any of multiple places with the same name.
3. *Aliasing*. A given location may have many corresponding toponyms.
4. *Nesting*. Toponyms may be nested within non-location entities.

The first two forms of ambiguity are frequently considered in geotagging research and are the main focus of this work. Resolving *geo/non-geo* ambiguity involves distinguishing between non-location names and toponyms. For example, as we pointed out earlier, “Stanley Jordan”, “Bristol Palin”, and “Paris Hilton” are names of persons, while “Bristol”, “Paris”, and “Jordan” are also toponyms. Resolving *geo/geo* ambiguity involves distinguishing between different interpretations for a given toponym. For example, over 40, 50, and 60 places around the world are named “Jordan”, “Bristol”, and “Paris”, respectively. The remaining two forms of ambiguity are not as problematic, but nonetheless require consideration. Aliasing must be accounted for in geotagging by considering alternate names used for locations. For example, “New York City” may be additionally referred to as “NYC” or simply “New York”. Furthermore, vernacular aliasing frequently occurs as in nicknames such as “Big Apple” and “Gotham”. The gazetteer should include location aliases to account for these variations. In addition, toponyms may be nested inside entities of other types, which imparts a shade of geographic meaning to these entities. For example, “University of [Maryland]”, “[New York] Police Department”, and “Mayor of [El Cenizo]” all contain toponyms, but as a whole refer to a different entity. This problem is further exacerbated by toponyms that are also instances of *metonyms*—i.e., toponyms that frequently refer to



other entities instead of their nominal locations. For example, “Washington”, “Westminster”, and “Hollywood” are prominent locations, but are also frequently used to refer to the US Government, the UK Parliament, and US cinema, respectively. In this work, we limit our analysis to non-nested toponyms.

To illustrate the widespread ambiguity present in location names, we present Figure 1.1, which shows a collection of ambiguous or otherwise unusual toponyms present in the United States. The displayed toponyms all correspond to populated places such as cities and towns. These toponyms are of many categories, and show both geo/geo and geo/non-geo ambiguity in their naming. Note that we found many more such toponyms, but we selected only a subset to display to avoid excess clutter on the map. We include some examples of these toponyms below, with each toponym followed by the US state abbreviation in which it is located. Some categories of geo/geo ambiguity in these toponyms include cities that have names of countries (e.g., “Belgium, FL”, “Canada, KS”, “Chad, KY”, “China, LA”, “Cuba, AL”, “Denmark, AR”, “Egypt, GA”, “Finland, MN”, “Germany, IN”, “Greece, NY”, “Ireland, OH”, “Italy, TX”, “Japan, MO”, “Lebanon, PA”, “Mexico, KY”, “Norway, NE”, “Panama, CA”, “Peru, MA”, “Poland, WI”, “Sweden, SC”, “Togo, MS”), cities that are also names of US states (e.g., “Arizona, LA”, “California, ME”, “Montana, NJ”, “Nevada, OH”, “New York, TN”, “Ohio, TX”, “Oregon, MD”, “Tennessee, TN”, “Texas, GA”, “Wyoming, IA”), cities that share names with other, more prominent cities (e.g., “Alexandria, VA”, “Berlin, PA”, “Cairo, OK”, “Geneva, NE”, “Havana, FL”, “London, KY”, “Paris, TX”, “Rome, KS”, “Tripoli, IA”, “Zurich, CA”), and even names of celestial bodies (e.g., “Venus, CA”, “Earth, TX”, “Moon, MS”, “Mars, PA”, “Jupiter, NC”, “Saturn, TX”, “Pluto, WV”, “Planet, AZ”). A wide variety of toponyms exhibit geo/non-geo ambiguity as well (e.g., “Admire, PA”, “Advance, MI”, “Appeal, MD”, “Bath, NY”, “Bland, FL”, “Cash, MS”, “Climax, KY”, “Cricket, IA”, “Experiment, AR”, “Gas, KS”, “Golf, IL”, “Gravity, PA”, “Igloo, AK”, “Ink, OH”, “Kite, GA”, “Not, MO”, “Okay, OK”, “Racy, MI”, “Roach, NE”, “Rural, WI”, “Santa Claus, AZ”, “Stop, GA”, “Unicorn, MD”,



“Waterproof, LA”, “Why, AZ”, “Zulu, AL”). These toponyms give insight as to the extreme ambiguity of toponyms and hint at the difficulty of automatic geotagging, where it is crucial to understand enough context to resolve these ambiguities correctly.

Next, we provide several examples of geo/non-geo and geo/geo ambiguity found in real news articles. These examples are from articles whose subjects reflect the difficulty of geotagging toponyms, and would be especially challenging for a fully automated geotagging process. In some cases, leveraging the streaming aspect of news can resolve the ambiguity present in these toponyms, as explained below.

Our first example, shown in Figure 1.2, comes from an article published in November 2008 in *Variety* [58] about Batman, a small city in southeastern Turkey. In this case, the success of the 2008 movie “The Dark Knight”, about Batman, the comic book superhero, prompted a lawsuit from the mayor of Batman, Turkey. Geotagging this article correctly involves distinguishing between the Turkish city and the superhero for each instance of “Batman” present in the article, which could be accomplished by observing contextual clues such as “town of [Batman]”. However, “Batman” presents an interesting case in that most articles in the news published at the time of the Dark Knight’s movie release that mentioned “Batman” would be referring to the superhero, and so “Batman” is likely not a location. In other words, taking advantage of the streaming, ever-changing aspect of news, and knowing that there is a recent movie about Batman the superhero, provides evidence against the interpretation of Batman as the name of a city.

A similar case can be seen in Figure 1.3, showing a Reuters article published in November 2008 [86] describing celebrations in the town of Obama, Japan due to the election of President Barack Obama, which offered business opportunities due to their shared names. In this case, understanding that Obama refers to a politician of great influence in the news offers a strong source of evidence in determining whether “Obama” refers to the politician (likely) or the location (unlikely).



## Mayor of Batman sues WB, Nolan

**Southeastern city in Turkey fights for name**

By ALI JAAFAR

Batman has a new adversary: Batman.

The mayor of an oil-producing city in southeastern Turkey, which has the same name as the Caped Crusader, is suing helmer Christopher Nolan and Warner Bros. for royalties from mega-grosser "The Dark Knight."

Huseyin Kalkan, the pro-Kurdish Democratic Society Party mayor of Batman, has accused "The Dark Knight" producers of using the city's name without permission.

"There is only one Batman in the world," Kalkan said. "The American producers used the name of our city without informing us."

No one from the town of Batman has explained why it took so many years to take legal action. Batman first appeared as a comicbook character in 1939 and the "Batman" TV series started in 1966. Tim Burton's first bigscreen rendition for Warner Bros. came out in 1989. Undoubtedly the fact that "Dark Knight" is about to pass the \$1 billion mark at the B.O. played a part in stirring the ire of the Turkish hamlet.

Figure 1.2: An example of geo/non-geo ambiguity in "Batman", which can refer to the movie hero or a city in southeastern Turkey.

# Japan's Obama town overjoyed

Wed, Nov 5 2008

By Toshi Maeda

OBAMA, Japan (Reuters) - The sleepy Japanese fishing town of Obama went wild Wednesday as locals gathered to celebrate namesake Barack Obama's victory in the U.S. presidential election.

More than a hundred residents gathered to watch the vote count on television in a public hall in the middle of the day, and chanted "Obama, Obama!" as the result was announced on a news program.

Some were clad in hula costumes in honor of Obama's birthplace in Hawaii. Others showed up wearing "I love Obama" T-shirts.

The town has taken advantage of the name -- one of many named Obama, or "small beach" in Japanese -- to launch products from fish burgers and steamed cakes to chopsticks.

Buoyed by the victory, locals say they hope Obama, who once mentioned the town in a television interview, will visit.

"The next thing we want to do is to go to the White House and dance the hula at Obama's inauguration ceremony," said Tatsuya Sano, 45, who runs a souvenir shop selling locally made Barack Obama souvenirs.

Chikako Shimizu, 35, the leader of an "Obama Girls" hula dance group launched this year, said she was calm while watching the vote count on television because she had no doubt Obama would win.

"I was convinced that he would win. I couldn't be happier," she said.

Obama City residents plan to dance and party more in the evening.

Figure 1.3: An example of geo/non-geo ambiguity for "Obama", referring to the American President or the town of Obama, Japan.



## Russia thrusts into South Ossetia; clashes with Georgia reported

5 hours ago

JAVA, Georgia (AFP) — Russian tanks and troops surged into Georgia's breakaway South Ossetia province on Friday to repel a Georgian offensive to reclaim the region amid fighting said to have left hundreds dead.

"Fierce clashes" between Russian and Georgian troops in the southern suburbs of South Ossetia's capital Tskhinvali were reported by Russian news agencies as night fell on the city.

Moscow had vowed retaliation to defend Russians in Tskhinvali who had come under fire by the Georgian artillery and air assault -- the worst fighting since the 1992-94 separatist war in the region.

"Georgian forces are controlling the entire territory of South Ossetia except Java," a city north of Tskhinvali, Georgian President Mikheil Saakashvili said in a televised address.

"We are fully controlling Tskhinvali," he added, although the rebels shortly after said that they were in control, according to the Interfax news agency.

Georgia accuses Russia of seeking to take over South Ossetia

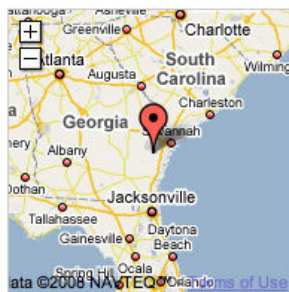


Figure 1.4: An example of geo/geo ambiguity in "Java, Georgia". In this article, "Java" referred to a town in Georgia in the Caucasus region, while the associated map, presumably generated automatically without human intervention, incorrectly shows a resolution to "Java Court", a street in Hinesville, Georgia, US.

Our third example shows the difficulty of resolving geo/geo ambiguity. Figure 1.4 contains an excerpt from an article published in August 2008 [5] about the 2008 war in South Ossetia, Georgia, in the Caucasus region. For this example, the article’s dateline contains a reference to “Java, Georgia”, which refers to a town in Georgia, the country. However, the map associated with this article, shown in bottom left, indicates that “Java, Georgia” was geotagged to a small city southwest of Savannah, Georgia, the US state. In fact, upon closer examination, it turns out that “Java” was placed at “Java Court”, a street in Hinesville, Georgia. Clearly, this map was the output from an automated geotagging algorithm whose output was not manually checked by a human. Note that this error is not overly surprising given that the instance of “Georgia” appearing in the US news usually refers to the US state, and the relative lack of prominence of the US “Java” interpretation when compared to the Caucasus “Java” was not enough to deter a US interpretation. Contemporary news stories also indicate that many humans also were confused about the proper interpretation of “Georgia” in articles about the South Ossetia war. Knowing that the Caucasus “Georgia” interpretation was frequently in the news at the time would allow correct resolution of these toponyms.

Our last example, shown in Figure 1.5, contains an excerpt from an article published in February 2010 [24] about problems resulting from the geo/geo ambiguity of “Vancouver”. In particular, during the 2010 Winter Olympics in Vancouver, British Columbia, Canada, many tourists who bought travel tickets and made hotel reservations actually did so for the US city of Vancouver, Washington. This ambiguity is even more interesting given that the Canada and US interpretations of Vancouver are relatively proximate to each other, with Washington state, US being adjacent to British Columbia, Canada. This example shows that even humans sometimes have difficulty with correctly resolving toponyms.

## Oops, wrong Vancouver

Thu, Feb 4 2010

By Teresa Carson

VANCOUVER, Washington (Reuters) - Sallie Reavey picked up the phone at her charming Briar Rose Inn and the caller asked about rooms in mid-February. "We have a nice selection of rooms for those dates," she replied, to which the caller gasped: "You still have rooms during the Olympics?"

Reavey had to tell him: wrong Vancouver.

The Briar Rose is in Vancouver, Washington, not Vancouver, British Columbia, the Canadian city that will host the 2010 Winter Olympics starting on February 12.

"America's Vancouver," as a former town mayor liked to describe it, sits 250 miles south of the Olympic host Vancouver and has a population of some 165,000 people -- far fewer than the Canadian city.

The Hilton Vancouver Washington has also fielded Olympic enquiries and trained its reservations staff to be sensitive to the possible mistake and, naturally, turn it into a marketing opportunity.

"We absolutely want them to come here," Gerry Link, the hotel's general manager said, adding of the Vancouver mix-ups: "So far it has all been pretty good-natured."



Figure 1.5: An example of geo/geo ambiguity in "Vancouver". During the 2010 Winter Olympics in Vancouver, Canada, many tourists mistakenly booked hotel rooms in Vancouver, Washington state.

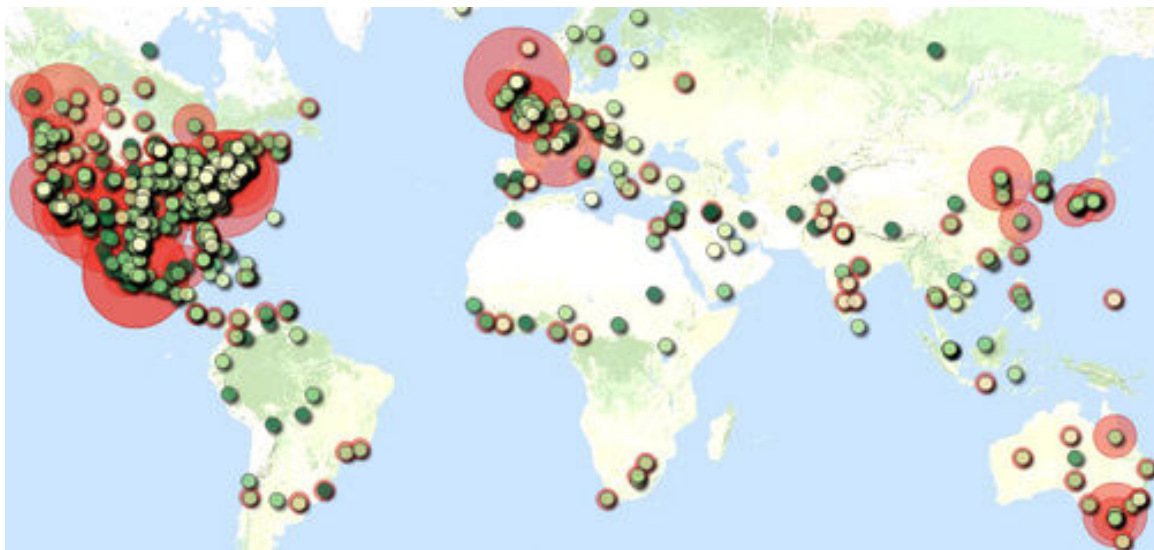


Figure 1.6: Locations mentioned in news articles about the May 2009 swine flu pandemic, obtained by geotagging related news articles. The larger red circles indicate frequency, with larger circles indicating more frequency, and small circles are color coded according to recency, with lighter colors indicating the newest mentions.

## 1.2 STEWARD and NewsStand

In this section we describe two systems that leverage geotagging technology, and served as motivations for our research. As noted earlier, we emphasize the use of multifaceted geotagging techniques, and it was primarily for these systems that our algorithms were developed. As a first system, we designed a *spatio-textual search engine* called STEWARD [78]. STEWARD was developed as a search engine where the query string contains a geographical entity, and we wish to find documents that are related to the location by spatial proximity, rather than exclusively by keyword containment. For example, a document containing “Los Angeles” is deemed relevant to a query string containing “Hollywood”, even though the query string “Hollywood” might not even be mentioned in the document. Note that most search engines rank documents based on their relevance to a user’s search string, which consists entirely of keywords. In particular, strict keyword searches involving location names, such as “College Park, MD”, would search for that exact phrase in document text, being completely ignorant of the underlying geographic information present in such a phrase. On the other hand, STEWARD does not collapse spatial searches into the framework of keyword searches. Instead, it makes use of the spatial information present in the location search to retrieve documents which are both textually and spatially relevant. Geotagging is a core component of STEWARD, and is necessary to judge a given document’s relevance to the query’s geographic component. STEWARD was also leveraged to create a disease monitoring system [79] by geotagging disease incidence reports from around the world. On a related note, Figure 1.6 illustrates worldwide outbreaks of swine flu in May 2009, obtained by geotagging news articles written about that topic, which can then be indexed spatially. The STEWARD system is described in detail in Chapter 2.

While STEWARD was originally designed for processing documents from the hidden Web, it was easily applied to collections of news articles. Eventually, this led to the development of another fully-automated system called NewsStand [143] (denoting “Spatio-Textual Aggregation of News and Display”). NewsStand’s geotagger, which uses the methods de-



scribed in this dissertation, associates news articles with the geographic references mentioned in them, and groups articles into story clusters based on their textual and geographic content. It then places markers representing story clusters on an interactive map interface, thereby allowing meaningful, visual exploration of the news.

NewsStand is designed to be scalable, responsive, and modular, with article processing divided among many independent modules. Central to NewsStand’s operation is its *pipe server*, which acts to coordinate NewsStand’s many backend processing modules by assigning batches of processing work via a communication protocol, and also monitors the system’s health by verifying that documents are being processed in a timely manner. Whereas the pipe server directs modules to perform work, NewsStand’s *SQL database*, based on PostgreSQL, stores information about documents present in the system, as well as the results of their processing, which enables NewsStand’s Web interface to easily retrieve location-associated news clusters for display by executing variants of what are known as *top-k window queries*. NewsStand’s architecture, design, and implementation are described extensively in Chapter 3.

### 1.3 Methodology

In this section, we briefly describe the techniques and geotagging methodology that will be explored in later chapters. Our goal is to design and implement a fully-automated, multi-faceted geotagging process, with no humans involved in any phase of processing. As noted earlier, geotagging consists of toponym recognition, wherein all toponyms in the document are found, and toponym resolution, where location interpretations are selected for each toponym. To understand our strategy for toponym recognition, consider that toponym recognition can be viewed as a subset of a more general problem studied in natural language processing, called *named-entity recognition* (NER). Whereas toponym recognition involves finding entities in text that correspond to geographic location names, named-entity recognition involves finding entities of several types, often including locations (e.g., names

of people, organizations, locations, dates and times, businesses, stock symbols, genes and proteins). For example, in the sentence “Jordan visited London last Friday”, the output from a toponym recognizer would include the location “London”, while correct output from a named-entity recognizer would also include “Jordan” as a person, and possibly “Friday” as a day of week. Sometimes evidence is stronger for a particular entity interpretation versus another interpretation. For example, in the pattern “ $X$  visited  $Y$ ”, the “visited” verb lends credence to  $X$  being a person and  $Y$  being a location, since locations are visited by people. Machine learning-based NER systems will often discover patterns like these from corpora of entity-annotated documents, and use them to build a language model through which entities and entity types can be predicted, given the linguistic context.

Given toponym recognition’s status as a subproblem of NER, tools developed for the more general problem of NER can be used as a means of toponym recognition, or at least as processing filters. In this case, the general strategy is to take an input document, execute an off-the-shelf NER system on the document (e.g., LingPipe [6], Stanford NER [37], ANNIE [31]), and retrieve location entities as the output. Once location entities are found, location interpretations are assigned from a gazetteer, and in the toponym resolution step, one of the interpretations is chosen for each toponym. However, this simple strategy is problematic. Because NER is a more general problem, systems developed for NER tend to be tuned for this more general problem, rather than specifically for locations and location-based evidence. As a result, they may be less accurate in detecting locations. Second, this strategy is inflexible in that the toponym recognition and toponym resolution procedures are completely independent, and thus cannot share evidence. For example, it may happen that a supposed toponym  $t$  found by the toponym recognition procedure is incorrect, i.e.,  $t$  should not have been selected as a toponym. However, the toponym resolution procedure is then forced to consider  $t$  as a toponym and select one of the incorrect location interpretations of  $t$ , even if none of these interpretations are evidenced by  $t$ ’s context. Returning to our example, in the sentence “Jordan visited London last Friday”, if “Friday” were in-

correctly recognized as a toponym, the toponym resolution procedure would necessarily select the interpretation in Texas. A better option would be to allow the toponym resolution procedure to drop toponyms discovered by the toponym recognition method that are not evidenced. Also, on a practical note, when evaluating NER systems on our domain of news articles, we found that they tended to be biased toward precision, at the significant expense of recall. This may be due to the small size and homogeneity of corpora used in training NER systems, which do not adequately capture the fast moving and ever changing nature of the streaming news cycle. While this bias is not unacceptable for the NER problem, it is problematic when used in a system for geotagging, since the toponym recognition procedure imposes an upper bound on the recall for the entire geotagging process (i.e., toponym recognition and toponym resolution). Put simply, a prerequisite for the geotagging procedure’s correct resolution of a toponym is its recognition of the toponym. Thus, any missed toponyms in the initial toponym recognition phase will negatively impact the recall for the entire process. As a result, the low recall of these NER systems imposes a severe limit on the entire geotagging process’s recall. Of course, it is worth noting that information about entities other than toponyms is useful for toponym recognition, since it may offer a means of resolving type ambiguities.

Bearing these considerations in mind, the toponym recognition process we designed for processing streaming news has a considerably more flexible architecture. Rather than solely relying on an off-the-shelf NER system, we include NER software as part of a multifaceted toponym recognition system involving many recognition methods, of potentially varying quality. We include rule-based recognition in the form of entity dictionary tables, cue word matching (e.g., “X County”), and toponym refactoring. In addition, we leverage statistical NLP tools in the form of NER software with postprocessing filters, and part-of-speech (POS) tagging with additional recognition rules. Essentially, we designed this multifaceted toponym recognition procedure in keeping with our goals to be flexible enough to capture variations that occur in streaming news, as well as to be as inclusive as possible when rec-

ognizing toponyms, in order to maximize the toponym recognition procedure’s recall (i.e., to miss as few toponyms as possible). Our toponym resolution methods, described in Chapters 5–7, serve to restore precision to the process by dropping supposed toponyms with no supporting evidence for any of their possible interpretations. In this way, we treat toponym recognition as only the first part of an integrated geotagging process also involving toponym resolution, rather than simplistically treating each step independently. Furthermore, during and after the recognition procedure, we allow entities to overlap, and even to exactly cover the same words in the text but with different types. For example, consider a document containing “Chad”. We may have evidence that “Chad” is a person entity, as well as different evidence that “Chad” is actually a location entity. Rather than keeping only one type, we create two entities with the same boundaries but different types. In other words, we defer the resolution of entity and interpretation conflicts to later stages of processing, so that we leverage as much evidence as possible to resolve these conflicts. Of course, recall is not the only factor in designing a robust toponym recognition system, and precision also plays a role. In our example document, if we came across a sentence containing “Mr. Chad Johnson”, the “Mr.” and “Johnson” provide strong evidence that “Chad” is a person entity and can be safely disregarded. However, in general, recall must be emphasized over precision due to toponym recognition’s aforementioned status as the initial stage of a combined geotagging process.

After toponym recognition, we use several toponym resolution methods. The central theme underlying these methods stems from the observation that news articles (and more generally, documents on the Web) are written to be understood by a human audience, and therefore geotagging will benefit from processing (i.e., reading) the document in the same way as an intended reader. Good writers understand what their audiences know, and will tailor their writing to appropriately address their audiences’ knowledge. By understanding the assumptions made by the writer about what the reader knows, and how they exploit that relationship, we leverage this knowledge for proper toponym resolution.

One type of resolution evidence that we use leverages a technique that is commonly used by article authors: *lists*, which for the purposes of exposition we refer to as *comma groups* (though commas need not always separate list items). Comma groups are a natural way to organize groups of related information. In fact, we note that each comma group unifies the entities it contains via a *common thread*—attributes that are shared by all entities in the group. This reasoning leads to our observation that for comma groups of toponyms, these common threads greatly aid in resolving the toponyms correctly. For example, despite each toponym in the comma group “Rome, Paris, Berlin and Brussels” having many possible interpretations (over 40, 60, 130, and 10, respectively), recognizing that these are large, prominent capital cities allows us to select the correct interpretations. Similarly, the group “Hell’s Kitchen, Chinatown, Murray Hill, Little Italy, and SoHo”, despite containing individually ambiguous location names, exhibits the common thread of neighborhoods in southern Manhattan, New York City, and this knowledge allows for their correct resolution. Comma groups are explored in detail in Chapter 5.

Another form of evidence we use for toponym resolution makes use of other assumptions made by article authors about their audiences. In particular, writers know the approximate boundaries of their audiences’ geographic knowledge, and will use toponyms in ways that are easily understood by their audiences. For example, a common method of referring to locations that are unfamiliar to the audience is the *object/container* form described earlier, where a pair of toponyms exhibit a containment relationship (e.g., “Paris, Texas”). On the other hand, toponyms that are well known to the audience will frequently be used in isolation (e.g., “Paris”), which can be troublesome due to their frequent ambiguity. However, by understanding the geographic knowledge of the intended reader, the geotagger’s seemingly daunting task of identifying the correct instance of “Paris” out of the more than 60 possible interpretations will be much easier when we note that the reader is unlikely to even be aware of most of these interpretations, and thus there is no need to even consider them as possibilities in the toponym resolution step.

This leads to a key point in our toponym resolution method which is that the reader’s *spatial lexicon*—those locations that the reader can identify and place on the map without any evidence—is very limited. In fact, even more importantly, this inherent limitation means that a common spatial lexicon shared by all humans cannot exist. Unfortunately, virtually all existing geotagging systems assume the existence of such a common lexicon. On the other hand, in our system, a key premise is the existence of a reader’s *local spatial lexicon* or simply *local lexicon* that differs from place to place, and that it is separate from a *global lexicon* of prominent places known by everyone. In other words, to readers in Texas, “Paris” primarily implies a reference to “Paris, Texas”, rather than to the distant, but more prominent, geographic location—“Paris, France”. Furthermore, in most cases, the local lexicon supersedes the global lexicon. Our detailed exploration of local lexicons in Chapter 6 shows their great utility in correct toponym resolution.

As a third source of evidence, we note that the two steps comprising geotagging can be considered as *classification* [36] problems: Toponym recognition amounts to classifying each word in the document’s text as part of a toponym or not, and toponym resolution amounts to classifying each toponym interpretation as correct or incorrect. With this understanding, and with appropriately annotated datasets, we leverage techniques from supervised machine learning to create an effective geotagging framework. In particular, we consider a new class of machine learning–based features to improve the accuracy of toponym resolution, termed *adaptive context features* [77]. These features construct a window of variable size around each toponym  $t$ , and use the other toponyms in the window to aid in resolving  $t$  correctly by considering the geographic relationships between interpretations  $l_t$  of  $t$  and those of other toponyms in the window. In particular, we search for interpretations that are geographically *proximate* to  $l_t$ , or are *siblings* of  $l_t$  in terms of a geographic hierarchy (e.g., cities in the same state). The more such relationships appear in the window, the greater evidence there is that  $l_t$  is the correct interpretation of  $t$ . We call these features *adaptive* because the window’s parameters can be varied for different domains, or

to achieve different ends, which affords us flexibility. We set parameters which we term the window’s *breadth* and *depth*, named analogously to breadth-first and depth-first search, which control the number of toponyms in the window and the number of interpretations examined for each toponym in the window, respectively. By varying these parameters, we control a tradeoff between feature computation time and resolution accuracy, which becomes important in the context of scalable streaming data processing. We describe our adaptive context features in more detail in Chapter 7.

## 1.4 Related Work

Recall that geotagging consists of toponym recognition and toponym resolution. We now provide a survey of existing work related to geotagging; for further overviews, refer to the surveys by Leidner [69] and Purves et al. [118]. Prior to our discussion, note that geotagging researchers have not yet settled on a uniform terminology, which reflects the different fields in which their work is rooted. Toponym recognition has also been called geoparsing [28, 60, 117, 118], toponym extraction [35, 132, 167], toponym detection [140], and georeferencing [27]. Similarly, toponym resolution has a variety of names, including toponym disambiguation [42, 78, 110, 132, 134, 135, 143, 167], geocoding [28, 117, 118], grounding [19, 60], and location normalization [35, 72]. Confusingly, a variety of names have also been given to the entire geotagging process, including georeferencing [134, 160], geoparsing [96], and geocoding [9]. To avoid confusion, in this work we use the terms toponym recognition, toponym resolution, and geotagging.

To be effective, a toponym recognition procedure must cope with *geo/non-geo ambiguity*, i.e., deciding whether a mention of “Paris” refers to a location or some other entity such as a person’s name. Many different approaches to toponym recognition have been undertaken, but share similar characteristics. The most common strategy is simply to find phrases in the document that exist in a *gazetteer*, or database of geographic locations, and many researchers have used this as their primary strategy [7, 19, 117, 132, 154, 155, 160].

Many of the gazetteers used by these researchers have small sizes, which in turn impose serious limitations on these systems' practical geotagging capabilities, as they are unable to recognize the small, highly local places that are commonplace in articles from local newspapers. In contrast, our own gazetteer contains over 8 million locations and thus is suitable for recognizing highly local toponyms.

To deal with the ambiguity inherent in larger gazetteers, researchers (e.g., [42, 73, 96, 121, 134, 135, 140]) have proposed a variety of heuristics for filtering potentially erroneous toponyms. MetaCarta [121] recognizes spatial cue words (e.g., "city of") as well as certain forms of postal addresses and textual representations of geographic coordinates. Unfortunately, this strategy causes serious problems when geotagging newspaper articles, as often the address of the newspaper's home office is included in each article. Given MetaCarta's primary focus on larger, prominent locations, these properly-formatted address strings play an overly large role in its geotagging process, resulting in many geotagging errors.

Other approaches to toponym recognition are rooted in solutions to related problems in natural language processing (NLP), namely named-entity recognition (NER) and part-of-speech (POS) tagging [61]. These approaches can be roughly classified as either rule-based [9, 23, 27, 28, 35, 60, 118, 124, 167] or statistical [69, 70, 72, 78, 97, 143] in nature. While statistical NER methods can be useful for analysis of static corpora, they are not well-suited to the dynamic and everchanging nature of the news, as has been noted by Stokes et al. [140]. Therefore, for our own toponym recognition procedure, we do not overly rely on any single method, instead opting for a hybrid approach involving multiple sources of evidence (see Chapter 4).

Once toponyms have been recognized, a toponym resolution procedure resolves *geo/geo ambiguity*, i.e., decides which "Paris" is the correct interpretation. Perhaps the simplest toponym resolution strategy is to assign a default sense to each recognized toponym, using some prominence measure such as population, and many researchers (e.g., [7, 19, 27, 28, 72, 73, 96, 118, 121, 140, 167]) have done so in combination with other methods.



MetaCarta [121] assigns default senses in the form of probabilities based on how often each interpretation of a given toponym appeared in a precollected corpus of geotagged documents. It then alters these probabilities based on other heuristics such as cue words and cooccurrence with nearby toponyms. This probability-based paradigm makes it nearly impossible for the less prominent places that so often frequent articles in local newspapers to be selected as correct interpretations, since these smaller places will have appeared in very few existing corpora of news articles. By contrast, our understanding of readers' local lexicons captures these smaller locations and allows their use for toponym resolution (described in Chapter 6).

Another very popular [7, 19, 28, 60, 73, 96, 117, 118, 132, 140, 155] strategy for toponym resolution is to settle on a "resolving context" within a hierarchical geographic ontology, which involves finding a geographic region in which many of the document's toponyms can be resolved. Web-a-Where [7] searches for several forms of hierarchical evidence in documents, including finding minimal resolving contexts and checking for containment of adjacent toponyms (e.g., "College Park, Maryland"). Note that the central assumption behind finding a minimal resolving context is that the document under consideration has a single geographic focus, which will be useful for resolving toponyms in that focus, but will not help in resolving distant toponyms mentioned in passing. Our adaptive context features (described in Chapter 7) capture a variant of this idea such that the windows of context can vary with the situation or application. Window-like features and heuristics have been used in other work related to geotagging (e.g., [70, 72, 97, 121, 135]), but these features' adaptive potential has not been explored elsewhere. Other resolution strategies involve the use of geospatial measures such as minimizing total geographic coverage [69, 70, 135] or minimizing pairwise toponym distance [72, 78]. Our own toponym resolution methods use a variety of heuristics inspired by how humans resolve toponyms.

Inferring local lexicons for a given news source's audience is related to finding the *geographic focus* of a document, i.e., the geographic coverage of toponyms in the document.

A number of approaches [7, 35, 134, 155], including Web-a-Where [7], again use a hierarchical ontology to determine geographic focus, with each resolved toponym contributing a score to its parents in the hierarchy, and settling on the ontology node with highest score as the geographic focus. This approach suffers from the same problem outlined above for situations where the document contains multiple geographic foci. Another common strategy is to select the most frequent toponyms as geographic foci [35, 73, 78, 143, 154]. Our local lexicon inference procedure, described in Section 6.1, which essentially determines the geographic focus of a news source, relies on several innate properties of local lexicons to aid in their discovery.

## 1.5 Summary of Contributions

In summary, the main contributions of this dissertation include:

- The design and development of the *STEWARD* system, a spatio-textual search engine that uses geotagging methods to extract toponyms from unstructured text documents. Contributions include the design of an initial geotagger, spatio-textual query processing, an intuitive user interface, and several novel applications, including tracking the spread of infectious diseases by geotagging PubMed and ProMED-mail documents.
- Contributions to the design and development of the *NewsStand* and its sister *TwitterStand* systems that enable the spatio-textual processing and retrieval of streaming news and tweets, respectively. Innovations include:
  - A highly modular and scalable processing architecture with modules for geotagging, and also for online clustering, image understanding, document topic classification, and disease and person name recognition.
  - The design and implementation of a central *pipe server* and a communication protocol to coordinate and delegate processing to individual modules.

- An SQL database design with appropriate schemas, tables and indexes that enhance spatio-textual query optimization, and facilitate the mapping of actions in these systems’ Web interfaces to SQL queries.
- A multifaceted toponym recognition procedure, using a combination of rule-based and machine learning–based methods to ensure high toponym recall.
- Toponym resolution algorithms that incorporate several novel types of evidence that improve resolution accuracy, including:
  - Recognizing and resolving lists of toponyms, termed *comma groups*, that share geographic characteristics and can be resolved simultaneously.
  - Automatically establishing and using the geographic knowledge of intended readership, termed *local lexicons*, to improve geotagging accuracy.
  - Considering windows of text of varying size around individual toponyms, termed *adaptive context*, which improve resolution of toponyms in the windows via shared geographic attributes.
- Encoding the above recognition and resolution evidence and algorithms both as sequences of rules and also as features to be used in machine learning–based geotagging frameworks.
- Detailed and comprehensive evaluations of all the above methods, including:
  - Data volume and throughput statistics for NewsStand, as well as database querying performance that characterizes NewsStand’s ability to support large numbers of users.
  - Two new corpora of streaming news articles that are hand-annotated with correct toponyms and their proper interpretations, which can be used to test geotagging performance. These corpora are much larger than typical collections

and are more focused on local news, which is much more representative of the vast majority of news on the Web.

- Comparative evaluations of geotagging performance with state-of-the-art systems such as Thomson Reuters’s OpenCalais, Yahoo! Placemaker, and others, showing large improvements over these systems.
- Investigations of the relative utility of each heuristic used for toponym recognition and resolution, in terms of toponym accuracy and machine learning–based importance scores.

## 1.6 Dissertation Outline

The remainder of this dissertation has the following organization. First, we describe two systems that use geotagging as central components, and thus serve as useful platforms on which to test geotagging methods. Chapter 2 provides an overview of STEWARD, a spatio-textual search engine served as the initial impetus for developing our geotagging methods. Next, Chapter 3 describes various aspects of the NewsStand system for processing streaming news, including detailed descriptions of the system’s many processing modules, its central pipe server and SQL database design, its Web interface, and several experiments designed to test its scalability and querying capabilities. After describing these systems, we continue with techniques used in our geotagging methods. We begin in Chapter 4 with a detailed description of our multifaceted toponym recognition methods. The following chapters expound on our toponym resolution methods, beginning with Chapter 5, which explores and describes our methods for recognizing and resolving comma groups of toponyms, and shows how to incorporate them into our overall geotagging procedure. Next, in Chapter 6, we describe our methods for toponym recognition and toponym resolution, and show how to discover and incorporate local lexicons into the geotagging process. Evaluations on corpora of news shows that understanding local lexicons is crucial to correct geo-

tagging of streaming news from smaller, local newspapers which dominate the news landscape on the Web. Chapter 7 contains a description of our methods for leveraging adaptive context features in a machine learning-based geotagging process. Each of the above chapters also contain descriptions of potential extensions to the work described in each chapter. Finally, Chapter 8 concludes the dissertation with a review of our main contributions, and poses additional open problems for future research.

## Chapter 2

### STEWARD: A Spatio-Textual Document Search Engine

Search technology today is dominated by search engines such as the one provided by Google, where documents are retrieved with the aid of an algorithm that ranks documents related to the query string on the basis of how many other documents link to it [21]. We are interested in developing a search engine where the query string contains a geographical entity, and we wish to find other documents that are related to it by spatial proximity. For example, a document containing “Los Angeles” is deemed relevant to a query string containing “Hollywood”, even though the query string “Hollywood” might not even be mentioned in the document.

In this chapter, we describe the anatomy of STEWARD [78, 128] (denoting “Spatio-Textual Extraction on the Web Aiding the Retrieval of Documents”), a spatio-textual document search engine which uses a geotagger as its primary component. STEWARD was developed as a search engine where the query string contains a geographical entity, and we wish to find documents that are related to the location by spatial proximity, rather than exclusively by keyword containment, as is the case with most search engines, which rank documents based on their relevance to a user’s search string. In particular, strict keyword searches involving location names, such as “College Park, MD”, would involve searching for that exact phrase in document text, being completely ignorant of the underlying geographic information present in such a phrase. On the other hand, STEWARD does not collapse spatial searches into the framework of keyword searches. Instead, it makes use of

the spatial information present in the location search to retrieve documents that are both textually and spatially relevant.

Existing work on spatio-textual search engines generally focuses on finding the geographic scope of websites containing multiple documents, and is usually done by examining their link structure. Instead, STEWARD's focus is on the contents of individual documents. Moreover, STEWARD also tries to identify as many toponyms as possible, rather than simply finding a geographic focus sufficiently general to span the entire document. Browsing through documents in order of proximity to the query string and specified query locations is also a major focus.

Queries to STEWARD can have a purely geographical component, a keyword component, or a combination of both. When the query string is purely a geographical entity, STEWARD finds documents that are related to it by spatial proximity. The documents that are returned are ranked by the extent to which STEWARD determines that the geographic entity in the query is relevant to the document. This is based on many factors, including the number of times that the proximate geographic locations are mentioned in the document, as well as their distribution throughout the document. On the other hand, when the queries consist only of non-geographic keywords, STEWARD ranks result documents according to the frequency and distribution of the keywords. After STEWARD's document standardizer converts each input document to a standard format for processing, STEWARD's geotagger identifies all of the toponyms in each document. This geotagger served as an initial foray into geotagging, which inspired the geotagging techniques described in later chapters.

STEWARD also ranks the locations found in each document in order of importance within the document's content. Rankings are based, in part, on the frequency of their occurrence, and the distribution of their occurrences in the document. When both a geographic location and input keywords are provided to STEWARD in a query, relevant documents (i.e., those containing the input keywords) are ranked in increasing order of distance of their geographic focus from the geographic location component of the query string. The

geographic location component of the input query can be expressed in terms of lat/long, or as a textual reference to a spatial object. For example, the user could search for “housing projects” in the vicinity of “College Park, MD”. The results would only return such documents that qualify both the content and location specifier that was provided to the system by the user.

STEWARD was initially developed for geotagging, browsing, and exploring reports on HUDUSER.ORG [54], a website provided by the Department of Housing and Urban Development (HUD) that distributes reports, periodicals, and other data published by HUD. It was later extended [79] for the geotagging of various other collections of documents, including PubMed abstracts [104] and ProMED-mail reports [55] in a disease monitoring role. It was also briefly used for geotagging collections of news articles, though NewsStand (described in Chapter 3) has since superseded its function in that domain. STEWARD is accessible on the Web at <http://steward.umiacs.umd.edu>.

In the rest of this chapter, we first describe the overall architecture of STEWARD (Section 2.1). Next, we delve into the details of STEWARD’s document standardizer, geotagger, and Web interface (Sections 2.2–2.4). We then discuss potential applications of STEWARD, including an application for disease monitoring (Section 2.5). Finally, we discuss open problems for future research (Section 2.6) and conclude the chapter (Section 2.7).

## 2.1 Architecture

Figure 2.1 shows an overview of STEWARD’s architecture. STEWARD is divided into several processing stages, with each stage having data independence. This architecture has the desirable property that each module in STEWARD can be stopped, resumed, or replaced without affecting the workings of the other modules. STEWARD’s first processing module is its Web crawler, which traverses and downloads all of a website’s public Web pages. Each document  $i$  is then passed to the document standardizer, which converts  $i$  into text and HTML formats, so that later processing stages work with a uniform set of data. Next,



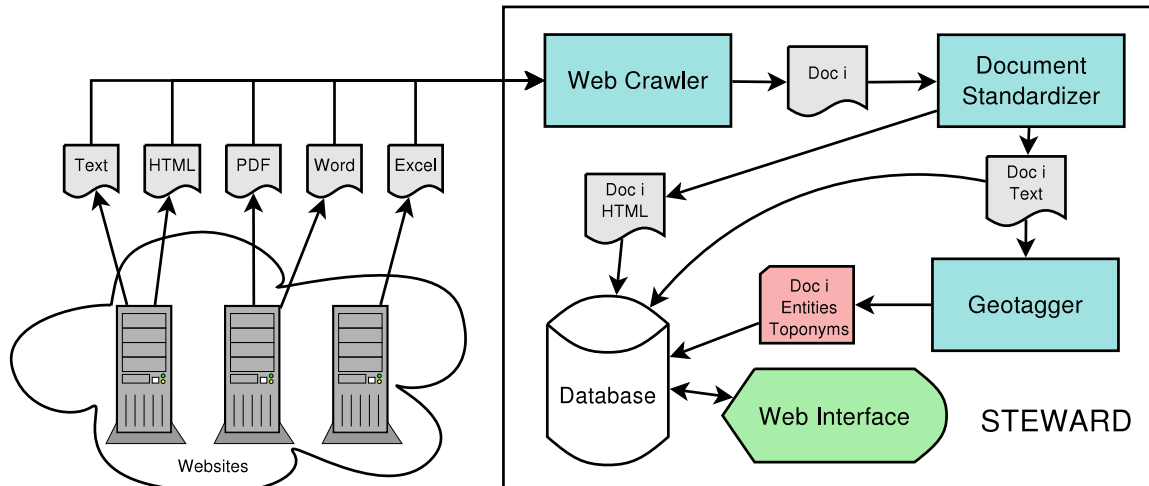


Figure 2.1: STEWARD’s architecture. STEWARD is a pipelined system divided into several data independent modules, so that each stage of processing can be assigned to a different computer.

STEWARD’s geotagger processes the text version of document  $i$ , recognizing and resolving toponyms. All output from these modules is stored in STEWARD’s central PostgreSQL database. Finally, users query STEWARD’s collection of processed documents through its Web interface. In the following sections, we describe these various modules in more detail.

## 2.2 Document Standardizer

Once a document is downloaded by STEWARD’s *Web crawler*, it is first processed through STEWARD’s *document standardizer*, which converts the document from its original format to a text version for geotagging and indexing, and an HTML version for display. The document standardizer ensures that later processing of documents is not dependent on their initial format. For example, further processing of PDF documents should not be different from that of plain text, HTML, Microsoft Word, or Microsoft Excel documents. Whenever possible, additional metadata is extracted from the document, including the title of the document, authors, publication time stamp, and modification date.

After a document has been standardized, it is stored in STEWARD’s database along with its URL, any available metadata, and the text and HTML versions of the document.

## 2.3 Geotagger

Once a document has been converted to text, STEWARD’s *geotagger* process is executed on the document. This process was the precursor to the geotagging methods described in later chapters. STEWARD’s geotagging process consists of toponym recognition, during which toponyms and other entities (e.g., people) in the document are identified. Next, the identified toponyms are resolved by selecting one of the location interpretation corresponding to each toponym. Finally, toponyms are ranked in order of decreasing importance to the document’s content. The resolved and ranked toponyms are stored in STEWARD’s database along with a link to the document in which they were found. The following sections describe STEWARD’s geotagging process in more detail.

### 2.3.1 Feature Vector Extraction

Rather than processing every word in a document, we wish to discard most of the words in the document that most likely are not textual references to geographic locations (e.g., “the”, “and”). Removing such words substantially reduces the amount of work required to process a document. To this end, this stage in the STEWARD pipeline focuses on identifying and extracting only those words and phrases from the document that are most likely toponyms, and are referred to as the *features* of the document. Collectively, the set of features is referred to as the *feature vector* of the document, and the process of extracting the feature vector of a document is termed *feature vector extraction*.

The most popular method for feature vector extraction of a document  $d$  is to compute the *Term Frequency-Inverse Document Frequency* (TF-IDF) [125] measure for each word in the document. The TF-IDF measure emphasizes those words and phrases which are frequent in  $d$ , but appear infrequently in a *document corpus*, which is a set of representative documents from the collection of documents. Given a set of words in  $d$ , the TF-IDF of a word  $w$  can be computed as the ratio of the number of times  $w$  appears in  $d$  to the number

of documents in the corpus containing  $w$ . Only those words that occur frequently in  $d$ , but are infrequent in the corpus, have a large TF-IDF score. The feature vector of a document can be obtained by extracting only those words whose TF-IDF score is greater than a pre-determined minimum threshold. The biggest drawback of this method is that it does not take linguistic cues, such as some of those described in Chapter 4, into account, which can be deduced by parsing the sentence constructs in  $d$ .

To account for this, we use techniques from *Natural Language Processing (NLP)*. One easy way to extract a document’s feature vector is to choose only those words in the document that are proper nouns. To this end, we examined the use of a *Part-Of-Speech (POS)* [61] tagger to aid in feature vector extraction. A POS tagger examines a stream of words and assigns a part-of-speech label (e.g., verb, noun, adjective) to each word in the stream. One typical approach implemented by POS taggers is using a *n-gram Hidden Markov Model (HMM)* [61], and assigns part-of-speech labels based on the most likely path through the HMM, given the input sentence’s word sequence. The states of the HMM correspond to part-of-speech labels, while the assignment of labels to words in a sentence corresponds to the most probable path through the HMM, given the input term sequence. The advantage of this method over TF-IDF-based extraction is that the words or phrases in the feature vector are more likely to be toponyms, although the POS tagger cannot distinguish between names of people, organizations, or other entities, which are also proper nouns. Furthermore, because of the n-gram nature of the HMM, a POS tagger is adept in identifying word phrases—a substantial improvement over TF-IDF based extraction. The HMM POS tagger, similar to TF-IDF, relies on a corpus of documents to build the HMM, and may require extensive training. The POS tagger used in STEWARD was trained on the Brown language corpus [39].

We also examined using a *Named-Entity Recognition (NER)* [61] tagger to aid document feature vector extraction. The NER tagger overcomes some of the POS tagger’s limitations by providing further classification of proper nouns into categories, including

*person, organization*, and most importantly, *location*. As a result, a feature vector extraction algorithm that uses the NER tagger outputs the words or phrases in a document that have been classified as locations by the NER tagger. In spite of the apparent advantages of the NER based feature extraction over the POS-based feature extraction, we point out that POS is generally much faster and more accurate than NER. We used the NER package from the LingPipe toolkit [6], which was trained on the MUC-6 news dataset [26].

In consideration of the above, we adopt a hybrid approach that makes use of both a POS and NER based tagger. All words in a document are first tagged with their corresponding part-of-speech labels. Next, only the proper noun phrases are extracted, along with their context, and passed through a NER tagger. If the proper noun phrase is then tagged as a location, it is added to the feature vector for that document; otherwise, it is ignored. Combining these two methods exploits the strengths of both approaches—the speed of the POS tagger, and the specificity of the NER tagger.

Once the feature vector has been extracted, it is stored in a database as a separate relation. For a document  $d$ , each feature  $f$  in the feature vector of  $d$  is first assigned a unique *feature id*. The feature id is stored with the starting offset of  $f$  in  $d$ , the length of  $f$ , the context of  $f$ , and the *doc id* of  $d$ .

### 2.3.2 Feature Record Assignment

After extracting the document’s feature vector, STEWARD checks to see if any of the features, which may or may not be toponyms, actually are toponyms, by searching in a large gazetteer of toponyms. STEWARD uses a freely-available database, provided by the US Board of Geographic Names, known as the *Geographic Names Information System* (GNIS) [152]. At the time of writing, the GNIS contains approximately 2.06 million locations in the US, including classification labels for locations, such as populated place, landmark, and park. The gazetteer provides the name and lat/long values of each location, along with associated location data, such as a hierarchical categorization of the location by

state and county, as well as population data.

If a feature is found in the gazetteer, then STEWARD designates the feature as a toponym, and extracts all the possible matching interpretations. As noted earlier, a toponym may be associated with any of several interpretations. The problem of determining which interpretation is the correct one is deferred to the next stage in the processing pipeline, described in Section 2.3.3. Those features that do not have a toponym interpretation are dropped, as they are probably not toponyms.

### 2.3.3 Disambiguation via Semantic Analysis

We now present a brief outline of our *disambiguation* algorithm, whose primary objective is to assign the correct location interpretation for each toponym feature  $f$  in a document  $d$ . Note that at this point, most of the features in  $d$  have one or more toponym interpretations from the gazetteer. This can be problematic when using a gazetteer as large as ours, since features in  $d$  may have multiple location interpretations, even when they are not toponyms. Moreover, some features in  $d$  may have a long list of location interpretations, only one of which is correct. As a result, the disambiguation algorithm has the added challenge of identifying those features that are not locations, as well as the added computational costs of identifying the correct interpretation from large sets of potential interpretations.

A key observation, exploited in our algorithm, is that when referring to a relatively unknown geographic location, it is a common practice to provide nearby references to more identifiable geographic locations or hierarchical context. These additional locations provide readers with a familiar geographic context, so they have a notion of the location's general area. For example, when referring to a location "Catonsville", it is common practice to mention that it is near "Baltimore", or is located in "Maryland"—the state containing "Catonsville". Here, the presence of the locations "Baltimore" and "Maryland" give evidence to the location of "Catonsville" and vice versa. Furthermore, it is unlikely to find another pair of "Catonsville" and "Baltimore" in some other part of the world, such that

they are geographically close, and at least one of them is a familiar place. In STEWARD, the population serves as a substitute for the place’s familiarity.

This leads to a simple algorithm which we term the *pair strength* algorithm. Pairs of location interpretations are compared to determine whether or not they give evidence to each other, based on the familiarity of each location, frequency of each location, as well as their *document* and *geodesic* distances. We define document distance as follows: given two features  $f_1$  and  $f_2$  in  $d$ , their document distance is the difference in the offsets of  $f_1$  and  $f_2$  from the start of the document. Given a pair of location interpretations, the algorithm determines the pair’s strength based on the frequency, document distance, geodesic distance and the populations of the pair of locations. The higher the score of a pair, the more likely it is that the interpretations of the pair are correct. The pair strength algorithm generates all possible pairs of location interpretations, which are then sorted in decreasing order of the strength of the pairs and stored in a list  $L$ . At each iteration, the pair with the highest pair strength is chosen and removed from  $L$ . This effectively assigns one or more features to one of its location interpretations. Each assignment may cause some of the pairs in  $L$  to become *infeasible*, in which case they are removed from  $L$ . For example, if “Springfield” is assigned to “Springfield, IL”, all instances of pairs with “Springfield, MA” are removed from  $L$ . Finally, when  $L$  is empty, each feature has been assigned to one of its location interpretations, and the disambiguation phase is complete. The list of assigned interpretations and pair strength scores are then stored in the database with the document’s *doc id*. Note that the pair strength algorithm and its computations bear similarity to the *adaptive context* features, described in Chapter 7, which generalize the notion of pair strength to multiple proximate toponyms.

### 2.3.4 Geographic Focus Determination

The next stage of processing computes the geographic *focus* of a document, as determined from the locations identified in the document. The geographic focus serves as an ordering

of the resolved toponyms in a document, and is presented in decreasing order of their relevance to the document’s content. We compute the *focus score* of a location  $l$  in a document  $d$ , which is the measure of the relevance of  $l$  to  $d$ .

Several methods can be used for determining the focus scores of all locations in  $d$ . A simple measure of  $l$ ’s focus score can be the frequency of occurrence of  $l$  in  $d$ . The problem with this measure is that each location in the document is considered in isolation, so the algorithm does not account for the fact that  $d$  may also contain a number of spatially proximate locations to  $l$ . For example, a document that mentions several locations in “Texas” should probably give more importance to the places in “Texas”, even though each of them may be mentioned only a few times. A more sophisticated algorithm may use a *container based* [7] or hierarchical clustering technique, which groups the locations in  $d$  based on their classification in a container hierarchy. The advantage of this clustering technique is that the locations are grouped according to a natural and logical division method, easily understood by humans. However, if a document contains only a few important locations spread over a large area, then the container object may become too large to be useful.

STEWARD uses an algorithm termed *Context-Aware Relevancy Determination* (CARD). The rationale behind the CARD algorithm is as follows. Two locations  $l_1$  and  $l_2$  are said to be *contextually related* in a document  $d$ , if  $l_1$  and  $l_2$  frequently occur in each other’s context in  $d$ . A location  $l$  is said to be important to  $d$  if  $l$  is *well distributed* throughout  $d$ , as well as *contextually related* to several spatially proximate locations in  $d$ . The CARD algorithm is an improvement over other proposed techniques, as it combines both the geodesic and the document distances between locations in  $d$ , and arrives at a focus score that is more relevant to the content of  $d$ .

## 2.4 Web Interface

Users interact with STEWARD through its *Web interface*. This interface was developed to support document browsing and exploration tasks that leverage STEWARD’s geotagger.

One key idea present in STEWARD's design is to separate spatial and non-spatial aspects of the documents being browsed, to permit parallel spatial and non-spatial exploration of a single document or collection of documents. In addition, several features allow users to traverse relevant portions of documents without leaving STEWARD's interface, which makes for a more cohesive user experience.

When users first connect to STEWARD, they are presented with the interface shown in Figure 2.2. Constructed using HTML and using Ajax, STEWARD's interface is divided into three portions: a top pane used for search inputs, referred to as the *search pane*, and two bottom panes which will serve as the output for the search results. The search pane contains several search options. The primary search filters include specifying search keywords, a search location, or both keywords and a search location. To specify a location, users can enter a textual location in the "Location" text box, and then click "Lookup" to retrieve the location's lat/long values. This functionality is provided by a geocoding service that makes use of Google's API. Alternatively, users can manually enter lat/long values in the "Lat/Long" input fields, or they can click "Capture" and then click a location on the map to save the clicked location as lat/long search parameters. To remove the keyword or location filters for subsequent searches, users can click the "Clear Keywords" or "Clear Location" buttons as appropriate. Users can also select one of several document collections processed by STEWARD to restrict the search to the chosen collection. In Figure 2.2, the user has selected "HUD USER", a collection of reports present on the HUDUSER.ORG [54] website. In addition, the "Search" button executes the search using the specified search parameters, and "Reset Search" affords users a quick way of returning STEWARD to its initial loading state. The four boxes below "Reset Search" serve as a search progress indicator, which will be filled once the search has completed. At the top, the "Advanced" tab gives users access to more advanced spatial query parameters that control the spatial search radius, search shape, and so on.

Figure 2.3 shows the results of a keyword search for "colonias", which are rural hous-



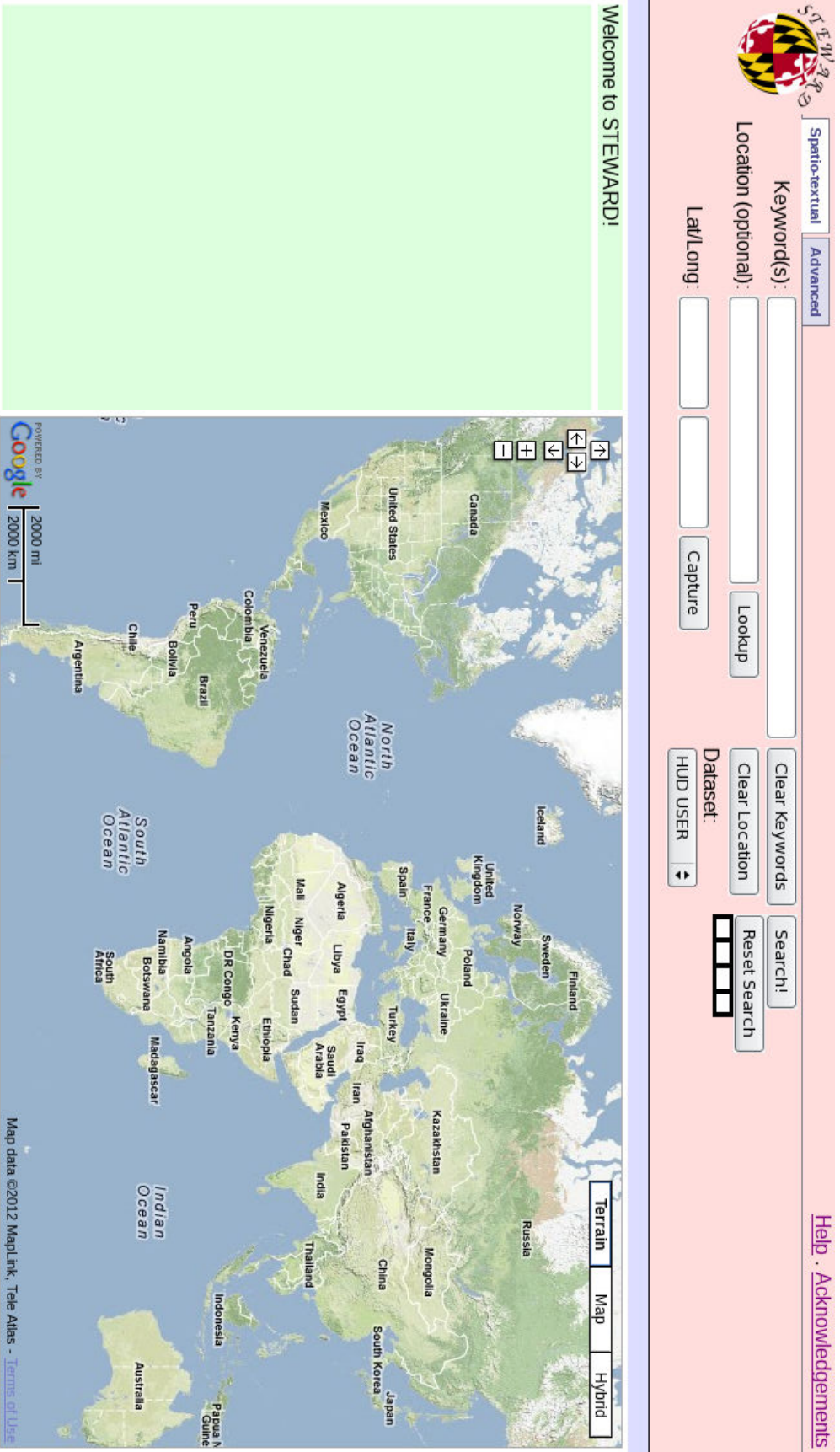




Figure 2.2: STEWARD's primary user interface, shown upon first accessing the system. Search inputs are entered in the top *search pane*, and search results are displayed in the lower left *textual pane* and lower right *spatial pane*.

STEWARDS  [Spatio-textual](#) [Advanced](#) [Help](#) [Acknowledgements](#)

Keyword(s):    

Location (optional):

Lat/Long:

**Results 1-10 of 22** [Previous Results](#) [Next Results](#)


**1** [Capacity Building and Governance in El Cenizo](#)  
**Score: 0.10** [Georefs: 21](#) [Focus](#)  
**Doc: [Original](#), [Highlighted](#)**  
**1 of 34 extracts**

of its local government. Following a description of the unique development challenges of the **colonias**, the article describes the participatory action research model adopted in this project and

**SOUTHWEST HOUSING TRADITIONS**  
**Score: 0.10** [Georefs: 55](#) [Focus](#)  
**Doc: [Original](#), [Highlighted](#)**  
**1 of 19 extracts**

and Urban Development is committed to meeting the unique housing needs of the citizens of the "**colonias**" those rural communities and neighborhoods located close to the U.S.-Mexico border that lack

**PROBLEMS AND SOLUTIONS**



Map data ©2010 AND, Europa Technologies, INEGI - [Terms of Use](#)

Figure 2.3: STEWARD's main interface, after the user has entered a textual search for the keyword "colonias", which are small settlements lying primarily along the US/Mexico border. Notice that most search results are clustered in Texas and Arizona, reflecting the geographic distribution of colonias in the real world.

ing settlements lying along the US/Mexico border, a topic of interest to HUD researchers. The bottom two panes provide query results. The left pane, termed the *textual pane*, contains a list of documents that matched the query parameters, and a variety of information about each match. Below the title of each document result, the document ranking score is given, which is an indication of how well the document matched the query parameters. This score is computed as a combination of the TF-IDF score computed over the document and keywords (described in Section 2.3.1) as well as the distance of the document's geographic locations from the spatial query point, if one was specified. Next to the ranking score, a "Georefs" link allows users to display the collection of locations geotagged in the document. In the first result in Figure 2.3, 21 such locations were found. A "Focus" link also allows users to enter focus mode, where they can explore the locations found in the document in greater detail. Additional links point to the original version of the document ("Original" link) as well as a highlighted copy of the document ("Highlighted"), wherein query keywords are highlighted so users can find their relevance. The focus mode and highlighted copy functionality will be described shortly. Finally, a snippet of text from the document containing the query keywords is also shown with the query keywords highlighted, and arrows above the snippet allow users to navigate through all document snippets that contain the query keywords. These controls enable users to navigate through relevant portions of the document without disruptively forcing them into a different window or application. In Figure 2.3, the lower right pane, termed the *spatial pane*, contains a map display that indicates the geographic foci of documents in the search results. The most prevalent location from each document is displayed as a numbered flag, corresponding to the numbered result in the textual pane. In addition, STEWARD has zoomed in to the smallest bounding box containing all search results. Notice that in the example, most search results have geographic foci that are clustered in Texas and Arizona, even though the query was not specified with a spatial component. This geographic distribution of search results reflects the geographic distribution of colonias in the real world, which serves to affirm the quality

of our geotagging algorithms, and demonstrates one of the knowledge discovery aspects of STEWARD.

Figure 2.4 shows STEWARD’s focus mode, which allows users to select a single document to browse all locations found in the document. In Figure 2.4, the user has entered focus mode for the first search result, “Capacity Building and Governance in El Cenizo” [158], as visually indicated by its highlighted title in the textual pane. This document is a HUD report about El Cenizo, a small city along the Texas/Mexico border. When entering focus mode, the set of locations found in the selected document are displayed on the map as a set of color-coded markers whose color indicates each location’s relative importance within the document, as measured by STEWARD’s focus algorithm (described in Section 2.3.4). Red indicates high importance, while blue denotes low importance. For this report, the highest-ranked location was El Cenizo, as might be expected given the report’s title. Clicking on a marker shows an info bubble containing various information about the clicked location, including its name, lat/long values, and a snippet from the document that contains the clicked location. Two sets of arrows in the upper right and lower left allow users to navigate through the document’s locations. The upper right set of arrows allows browsing of different locations in the document, in order of decreasing importance. In Figure 2.4, 21 such locations were found in the document. The lower left arrows enable users to navigate through all snippets in the document that contain the current location, which in Figure 2.4 is El Cenizo. Like the arrows in the textual pane, these spatial pane arrows allow spatial browsing of the document without interrupting the user’s experience. The info bubble also contains a link to open a highlighted copy of the document being browsed, with all instances of the current location highlighted.

Note that the textual and spatial panes amount to parallel, though different, views of the same search results. The textual pane is a mostly textual view of the documents, containing titles and snippets matching query keywords. On the other hand, the spatial pane is a spatial view of documents, with info bubbles containing location information and snippets

ST. J. 91 99 S

Spatio-textual | **Advanced**

Keyword(s): Colonias

Location (optional):

Lat/Long:

LookUp

Capture

Clear Keywords

Clear Location

Search!

Reset Search

Dataset: HUD USER

Help · Acknowledgements

---

Results 1-10 of 22

[Previous Results](#) [Next Results](#)

**1** **Capacity Building and Governance in El Cenizo**

Score: 0.10 Georefs: 21 Exit Focus Mode

Doc: [Original](#) [Highlighted](#)

1 of 34 extracts

of its local government. Following a description of the unique development challenges of the **colonias**, the article describes the participatory action research model adopted in this project and

**2** **SOUTHWEST HOUSING TRADITIONS**

Score: 0.10 Georefs: 55 Focus

Doc: [Original](#) [Highlighted](#)

1 of 19 extracts

and Urban Development is committed to meeting the unique housing needs of the citizens of the "**colonias**," those rural communities and neighborhoods located close to the U.S.-Mexico border that lack

**3** **PROBLEMS AND SOLUTIONS**

61.00  
El Cenizo  
27,350, -99,490

1 of 21 georefs

the efforts to acquire housing and infrastructure and to enhance governmental performance in **El Cenizo**, a Texas colonia outside Laredo. The efforts in El Cenizo involve a wide range of actors, but

1 of 61 extracts

[Jump to Highlighted Copy](#)

Terrain Map Hybrid

500 mi 500 km

Map data ©2010 AND, Europa Technologies, INEGI - Terms of Use

Figure 2.4: STEWARD's focus mode, where the user is browsing through all locations found in the first result document. Location markers on the map are color-coded by their frequency within the document. Clicking on a marker displays an info bubble containing information about the clicked location.

containing locations. In addition, these parallel views are further reinforced through the numbered flags representing documents, which serve to tie together elements from both panes. Clicking on one of the numbered flags in the textual pane will highlight the corresponding marker in the spatial pane. Similarly, clicking on a numbered flag in the spatial pane will cause the textual pane to scroll to the corresponding search result.

If users click on the “Highlighted” link in the textual pane, or “Jump to Highlighted Copy” in the spatial pane, they are taken to STEWARD’s highlighted copy browser, which is shown in Figure 2.5 after clicking these links for the selected document in Figure 2.4. The “Highlighted” link results in a display of the document with all instances of the query keywords highlighted. Figure 2.5a shows this mode after clicking on the “Highlighted” link for the selected document in the textual pane, where all instances of the query keyword “colonias” are highlighted. Similarly, clicking on the “Jump to Highlighted Copy” link in the info bubble in Figure 2.4 displays a copy of the document with all instances of “El Cenizo” highlighted. This highlighting is done on an HTML version of the document in question, which is generated and stored in STEWARD’s database during the document’s processing. Various arrows in the interface allow for easy navigation between instances of the location. The currently-highlighted location is shown in red, while remaining instances of the location are shown in yellow. Clicking on the arrow to the right of a highlighted location moves the page to the next instance, while clicking on the left arrow moves to the previous instance. The arrows in the upper right corner results in the same navigation actions of moving to the previous or next location instance. However, unlike the arrows surrounding specific instances of highlighted locations, this upper right set of arrows is intended to allow easy and rapid navigation, since they do not move when the page scrolls. Users can thus continue clicking an arrow and navigating to the next location instance without moving the mouse.

Progress has occurred slowly, and the colonia problem is a significant element of the Texas policy agenda (Ward, 1999). But the intergovernmental context for responding to the <colonias> problems and implementing programs is very complex. Compared with other States, Texas has a decentralized and fragmented governmental system. County government is relatively weak, with very limited land-use control powers and very limited responsibility for the types of infrastructure systems needed in the <colonias>. In contrast, the home rule powers of city government, especially annexation powers, are quite strong. There are many types of special districts, including water districts organized to serve agriculture interests. Other than funds for public school systems there is limited State aid for local governments.

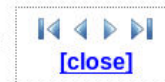


Inadequate infrastructure in <colonias> often results from private developers failing to fulfill their commitments. Although there are a number of alternative arrangements for the provision of water and wastewater, a variety of impediments exist for each. Agriculture-based water supply corporations have not provided services to <colonias>, although rapid urbanization, shifts in the economy, and difficult climatic conditions (freezes and droughts) have displaced some agricultural production. In these areas, water corporations have been more interested in alternative supply arrangements (Wilson, 1997). Another potential solution to infrastructure provision for <colonias> adjacent or close to cities is annexation. The expense of annexation is significant, but State and Federal authorities have tried to facilitate this solution.

These solutions, however, are not applicable to El Cenizo, and the community's choice was to incorporate. Once incorporated, the city commission became responsible for responding to community needs. The local governance process in El Cenizo will be discussed below, after a number of conceptual points concerning governance are considered.

#### (a) Query keywords

learning activities were undertaken between spring 1996 and summer 1998 and included the placement of UT students on internships in <El Cenizo>, a policy research initiative on <El Cenizo> and other colonias (Chapa, 1997a, 1997b), and a semester-long community development project that will be discussed in some detail in this article. This last activity was initiated as a response to the great frustration in the community concerning the performance of its local government and the other governmental agencies involved in infrastructure provision. After a series of discussions in the community, the initiative decided to focus on capacity building and improvement in local government through youth development.



Three principal questions are explored in this article. First, how can the performance of government be enhanced in small, low-income communities like <El Cenizo>? Second, how can community involvement in government in low-income communities be enhanced? Finally, how can university service-learning projects and other outreach initiatives be framed to facilitate capacity building and government performance in such communities?

The article is composed of four sections. The first briefly describes the colonias of South Texas for those readers who may be unfamiliar with the setting. The second presents the conceptual frame of reference for the university-community initiative, which draws on a concept of governance defined as the relationship between civil society and government. The effectiveness of governmental action depends on the extent to which citizens are organized and politically active as well as the capacity of government itself. The third section examines <El Cenizo> using this framework. A brief history of the community is presented, emphasizing its various efforts to acquire local public services. Then the capacity of local government in <El Cenizo>, particularly in terms of municipal finance and infrastructure provision, is assessed. The final section of the article focuses on the COPC-

#### (b) Locations

Figure 2.5: STEWARD's highlighted copy browsing mode. This mode shows a version of the processed document with either (a) query keywords or (b) locations highlighted. The report on colonias in El Cenizo selected in Figure 2.4 has all instances of "colonias" and "El Cenizo" highlighted. Arrows allow navigation between instances.

## 2.5 Application: Disease Monitoring

The STEWARD system can be leveraged in a number of new application scenarios. For example, it was briefly used as a tool for reading news articles. Instead of organizing articles solely on topics, as done in, e.g., Google News [45] and MSN Newsbot, STEWARD can embed news articles on a map, representing each article by its principal location as determined by its toponym ranking algorithm. However, this functionality was somewhat superseded by the development of the NewsStand system, described in Chapter 3.

In this section, we focus on one such application of STEWARD to create an infectious disease monitoring system [79] that automatically classifies and organizes disease incidence reports, based on geographic location and type, for analysis by domain experts. The system searches documents on the Web for references to infectious disease names, as well as references to geographic locations. In particular, the documents searched include PubMed [104], which are papers published in biomedical journals, and ProMED-mail [55], a mailing list for doctors and other medical professionals around the world to report disease outbreaks. If a document mentions “cases of avian influenza in Indonesia”, our system is able to identify “avian influenza” as an infectious disease and “Indonesia” as a geographic location. The system then associates that document with the appropriate disease type, as well as the set of lat/long values of the geographic locations found in the document, after which the document is displayed on the map interface.

This disease monitoring system identifies textual references to infectious diseases by using an *ontology* of infectious diseases. An ontology is a hierarchical database of the important concepts and relationships in some knowledge domain, which in this case is infectious diseases. For a particular infectious disease, the ontology includes the disease’s medical name, common name, scientific classification of the disease-causing pathogen (in terms of class, order, and genus), common symptoms, and relationships to other diseases. Note that identifying references to diseases and selecting the corresponding ontology concepts suffers from similar ambiguity problems as those seen in geotagging. For a more



detailed description of the disease monitoring system, refer to Lieberman et al. [79].

One example query to STEWARD in its disease monitoring role is presented in Figure 2.6. The figure shows query results after a search for “avian influenza” in ProMED-mail [55] disease reports. The geographic distribution of locations found in result documents indicates contemporaneous (May 2007) patterns of outbreaks of avian influenza. In addition, the marker colors indicate the importance of each location within the disease report, with Thailand and China prominent among geotagged locations, as shown by the marker colors.

## 2.6 Open Problems

Developing STEWARD has highlighted a number of directions for future research. First, STEWARD’s geotagging and focus determination algorithms (described in Sections 2.3.3 and 2.3.4 respectively) could be empirically validated using annotated datasets of documents, as well as compared with other systems that perform similar functions, such as MetaCarta [121] and Web-a-Where [7]. Such validations would study the tradeoff between geographical locations in documents that have been missed versus the keywords in documents that have been incorrectly identified as locations.

One potential improvement for STEWARD’s geotagging algorithm is the identification of publisher addresses and citations within documents. We found that oftentimes in the scholarly articles processed by STEWARD, lists of references are often provided, and publishers (e.g., Prentice-Hall, Springer) have their addresses mentioned in citations. Our geotagger found these locations, and if there are enough of them, finding the geographic focus can be more difficult because of these noisy locations, which are not truly relevant to the document’s content. Given that STEWARD was initially designed to process documents in the hidden Web, it is reasonable to account for these locations and addresses in citations by detecting and discounting them. For example, some aspects of these locations can be exploited, such as their appearing in a separate “References” section, or more generally at

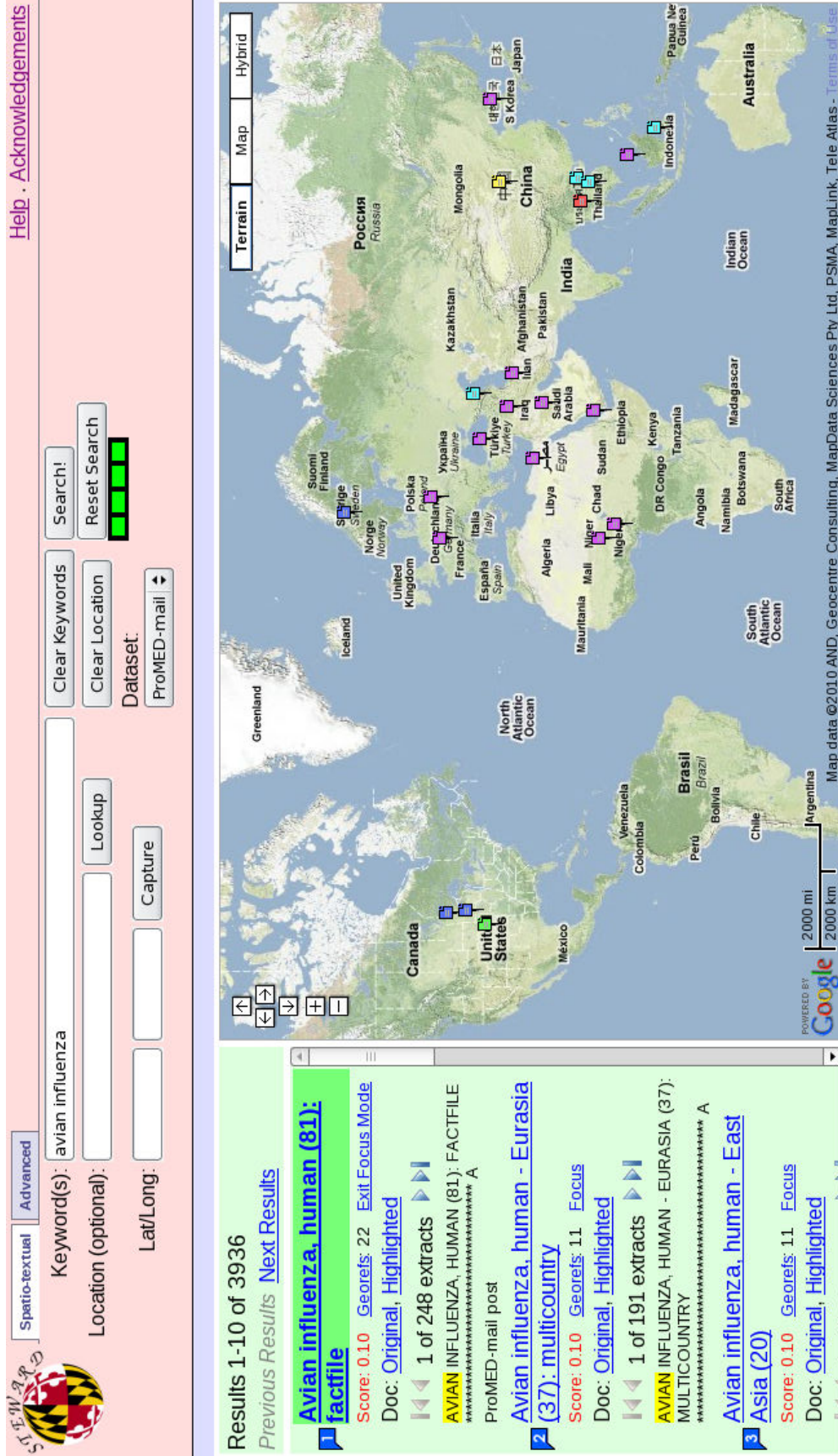


Figure 2.6: STEWARD’s Web interface after a search for “avian influenza” in ProMED-mail [55]. The geographic distribution of locations found in result documents indicates patterns of outbreaks of avian influenza in May 2007.

the tail end of a document. Citation detection in documents is an active area of research (e.g., [18, 32, 51, 63, 166]) and given our data domain, some of these approaches, or novel ones, could be applied within STEWARD. Similarly, STEWARD could be leveraged in other domains if combined with suitable ontologies. For example, STEWARD could be used to collect and organize tourist, historical, or recreational information about a city or a region.

STEWARD also presents several new challenges in the database field, in that it enables spatial and spatio-textual queries on document collections. These queries can be understood as variants of top- $k$  queries that return the  $k$  most relevant documents, based on the textual and spatial query parameters. While spatial queries have been researched extensively, methods of efficiently executing spatio-textual queries in a database needs investigation. Currently, when undertaking a spatio-textual query, STEWARD initially applies a keyword search, followed by the spatial search on the resulting document collection. This method slows the search considerably, as an index can only speed up the keyword search. If STEWARD had a hybrid index designed for spatio-textual queries, executing queries would see substantial speed improvements. The proper way to build such an index, as well as its properties, need further investigation. Furthermore, an important aspect of STEWARD's querying involves understanding the mechanics of spatio-textual query processing, in order to optimize retrievals for speed, efficiency, and a smaller memory footprint. This is usually a question of determining which component of the search (i.e., textual or spatial) should be performed first. A good search strategy should also take into consideration the size of  $k$  when performing top- $k$  queries, the average cost of performing a textual search, the cost of performing a spatial search, and the cost of parallel vs. sequential algorithms. STEWARD provides an ideal platform to test various such indexing and querying algorithms.

## 2.7 Summary

In this chapter, we discussed the architecture of the STEWARD system, which is a spatio-textual search engine for documents on the Web that served as the original impetus for developing NewsStand, described in Chapter 3, as well as the geotagging algorithms described in Chapters 4–7. STEWARD opens up exciting new possibilities for GIS researchers by providing the ability to extract and query geographical information from unstructured text documents, which is a cumbersome and difficult medium with which to work. Some of the challenges in designing a system like STEWARD include being able to correctly identify most georeferences in a document, and to reduce the occurrences of false positives that occur when a word is incorrectly identified as a location. This problem is further complicated by correctly identified georeferences which are not relevant to the document’s content; such georeferences should be assigned a low focus score. For example, locations in the bibliography of a document are not relevant to the content of the document. There is also a problem of dealing with names of organizations and persons, as well as addresses, which contain geographical locations (e.g., “University of [Maryland]”, “Mayor of [El Cenizo]”). On another level, and an important contribution of this work, is that we have highlighted the need for Web-based publication standards that would facilitate and enhance spatio-textual querying and browsing capabilities. Adoption of such standards would enable more up-front rather than backend processing approaches, which would resolve some of the ambiguities mentioned above, and hence greatly improve text mining capabilities on the Web.

## Chapter 3

### NewsStand: Map-Based Exploration of Streaming News

With the explosion of digitization in the Internet age has come a deluge of user-consumed and user-generated data. One major domain that has been affected considerably is that of the news media. Newspapers that were previously published on a weekly or daily basis due to distribution limitations can now be accessed around the clock on the Web, and hence have moved toward a continuous publishing model. Furthermore, non-traditional news sources such as blogs and Twitter have grown in importance so that they now compete with the traditional news media for audience attention. Web-capable mobile devices, recently coming into their own right, provide an additional, powerful means of accessing and generating news on the Web. This new, collective, and collaborative news media generates a constant supply of *streaming news* for information consumers. Our goal is to automatically collect, organize, understand, and index these many sources of streaming news so as to facilitate news exploration and retrieval by end users, specially with the aid of a map query interface. Furthermore, since streaming news appears and fades from relevance so rapidly, we likewise wish to make this streaming news available to end users as fast as possible. Given the amount of streaming news on the Web, and the considerable recent attention given by researchers to continuous queries over streaming data (e.g., [50, 53, 59, 74, 82, 102, 108]) and the many systems exploring streaming news as well as other media such as images, audio and video (e.g., PersoNews [14], Newsjunkie [41], Europe Media Monitor [64], HealthMap [40], and Perseus [65]), this is clearly a significant challenge.

Importantly, in the news domain, the so-called *Five Ws* (and H)—who, what, when, where, why, and how—are key to a well-written and comprehensible news article. In particular, a news article usually emphasizes the “Where”, reporting events in a certain geographic region. However, popular news aggregators such as Google News, Yahoo! News, and Microsoft Live News have only a rudimentary understanding of the implicit geographic content of news articles, usually based on the address of the publishing newspaper. Furthermore, these systems present articles grouped by keyword or topic, rather than by geography. Given that much of the interest in news is motivated by location-related attributes of readers (e.g., where readers are situated, hail from, aspire to be), it is somewhat surprising that they cannot deal easily with the two most common types of spatially-related queries:

1. *Feature-based*: “Where is story *X* happening?”
2. *Location-based*: “What is happening at location *Y*?”

We focus on enabling readers to answer these queries and we do so by presenting the responses using a map query interface, rather than the conventional linear interface that mimics a traditional newspaper, where the articles are presented in order of their importance as deemed by an editor with no attention to location. This layout forces readers to perform a brute force sequential search (i.e., read the various articles while looking for mentions of the locations which interest them). It is also noteworthy that this interface is linear and static, whereas the map query interface is dynamic, in that the articles associated with a particular location can vary over time without disturbing the positioning of other articles.

To answer the above and related queries, we present an automated system called NewsStand [143] (denoting “Spatio-Textual Aggregation of News and Display”) that uses transactional database technology and is available on the Web<sup>1</sup>. NewsStand automatically associates news articles with the geographic references mentioned in them (i.e., performs geotagging on them), and groups articles into story clusters based on their textual and geographic content. It then places markers representing story clusters on an interactive map

---

<sup>1</sup><http://newsstand.umiacs.umd.edu>

query interface, thereby allowing meaningful, visual exploration of the news. For example, stories mentioning “College Park, MD” are represented by suitably-placed markers on the map at the location corresponding to College Park in Maryland. NewsStand originated as an outgrowth of STEWARD (described in Chapter 2), and its success led to the creation of a sister system, TwitterStand [130], which gleans news events from tweets.

Note that readers may not initially see stories on the map due to several factors, such as their relative significance to other stories, and the current pan position or zoom level. The interplay between significance and zoom level is an important feature of NewsStand, and differentiates it greatly from existing spatially-referenced news reading systems (e.g., MetaCarta GeoSearch News [100], which maps locations in stories using MetaCarta [121]). The absence of dynamic zooming in these systems means that the set of stories presented to readers is static, rather than dynamic as in NewsStand. NewsStand’s use of the map as the medium for spatial news aggregation also differentiates it from Google News [45], Microsoft Bing News [101], and Yahoo! News [163], all of which feature limited local news coverage, usually accessible by entering a city or postal code. However, the presented list of articles appears to be based primarily on the publication location of the newspaper, rather than story content. The AP Mobile News Network [13] exemplifies an even coarser determination of geography, based on where the story was filed. For example, a story submitted to the Maryland news wire would be associated with all postal codes in Maryland.

NewsStand is designed to be scalable, responsive, and modular, with article processing divided among many independent modules (Section 3.2). The system collects and preprocesses news articles from over 10,000 news sources on the Web. Articles are grouped into story clusters using an online clustering algorithm. NewsStand’s geotagger then assigns geographic locations to each article. Articles are also geographically aggregated and ranked by story significance, as measured by the number of distinct news sources mentioning the story and several other factors. In addition, news stories are spatially aggregated, ranked, and displayed based on the current position and zoom level in the map query interface. For

example, when viewing the entire world in the map, users only see markers corresponding to stories that are significant to an international audience, thus imparting a sense of the major news events happening around the globe. As users zoom and pan to different geographic areas, NewsStand continuously updates the map to keep the display full of relevant story markers. Users can zoom in to a country, state, or city level to see increasingly local stories. Just by extracting geographic content from news stories, this relatively sparse set of controls gives users the power to better understand current events in terms of geography.

We also dedicate space here for explanation of the framework that enables the smooth and rapid operation of these modules, and how our design goals of scalability, reliability, and interactivity shaped its construction. In particular, central to NewsStand’s operation is its *pipe server*, described in Section 3.3, which acts to coordinate NewsStand’s many backend processing modules by assigning batches of processing work via a communication protocol, and also monitors the system’s health by verifying that documents are being processed smoothly. Where the pipe server directs modules to perform work, NewsStand’s *SQL database*, based on PostgreSQL and described in Section 3.4, stores information about documents present in the system, as well as the results of their processing. This database design evolved from that used in STEWARD [78] (described in Chapter 2), a spatio-textual search engine with similar processing and querying capabilities. After processing the input news documents, NewsStand’s user interface retrieves location-associated news clusters to display in its map user interface by executing variants of what are known as *top- $k$  window queries*. These queries, described in Section 3.6, retrieve the  $k$  highest scoring news clusters in NewsStand’s database that are located in the map’s current viewing window. Experiments in Section 3.7 illustrate the voluminous nature of streaming news processed by NewsStand, and demonstrate its ability to rapidly process and retrieve streaming news.

In the rest of this chapter, we first provide a high level overview of NewsStand’s architecture (Section 3.1). Next, we delve into the details of NewsStand’s processing modules (Section 3.2). We then describe NewsStand’s pipe server and its communication protocol



(Section 3.3) as well as NewsStand’s SQL database, including its data schema (Section 3.4). We introduce and describe NewsStand’s Web interface (Section 3.5), which is supported by SQL queries for streaming news retrieval (Section 3.6). Next, we present the results of several experiments that characterize NewsStand’s processing and querying time of streaming news (Section 3.7). Finally, we discuss open problems (Section 3.8) and conclude the chapter (Section 3.9).

### 3.1 Architecture

In this section we present a high level overview of NewsStand’s current architecture. NewsStand captures the latest news from thousands of individual news sources, and processes about 50,000 new articles every day (as of January 2012). Therefore, the most important criteria in designing NewsStand’s architecture were scalability of the system and the fast processing of individual articles. Additional goals included presenting the latest news as quickly as possible, within minutes of online publication, and being robust to failure.

Figure 3.1 shows a graphical overview of NewsStand’s architecture. NewsStand’s back-end processing is organized as a pipeline, with documents entering and streaming through the pipeline and being processed as they flow through the pipeline’s various stages. To enable efficient distributed processing of articles, NewsStand’s collection and processing of news is divided into several modules, shown in Figure 3.1 as rounded rectangles in blue. Each module performs a different type of processing on documents added to the system, with later modules in the pipeline often depending on results of earlier modules. Arrows indicate data flow through NewsStand through these modules, each of which process their data independently. This independence allows many instances of modules to be run on separate computing nodes in a distributed computing cluster. Because each module might execute on a different node, a given article might be processed by several different computing nodes in the system. In addition, we designed the modules in a way that allows for multiple instances of any module to run simultaneously on one or more nodes. As a result,

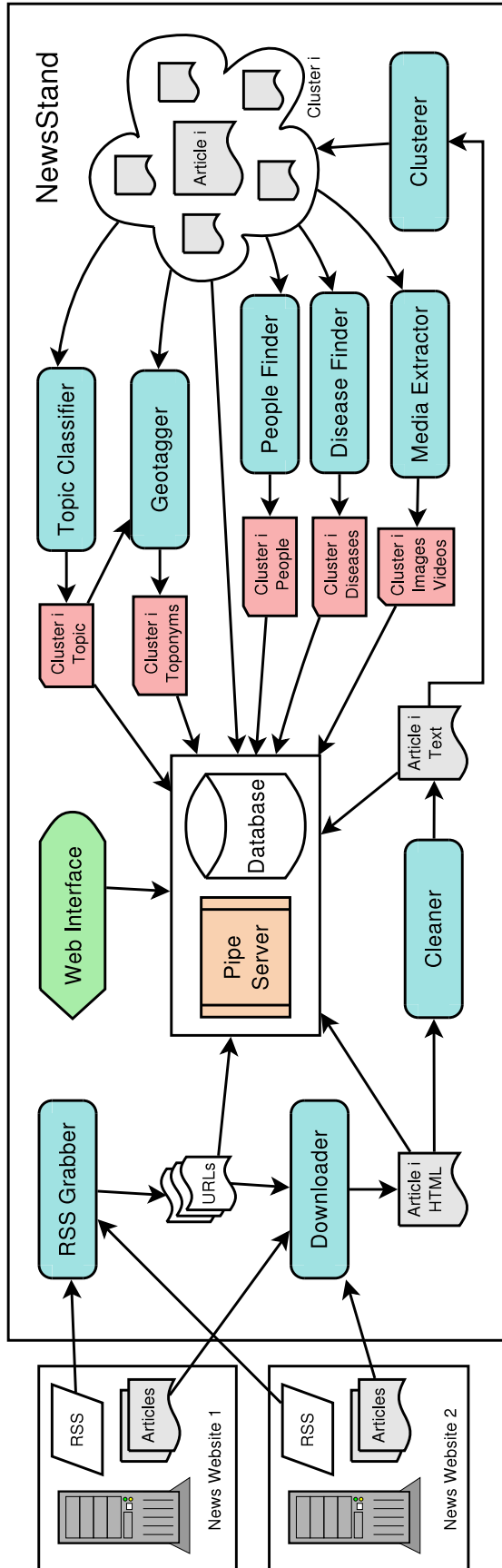


Figure 3.1: An overview of NewsStand's architecture, including processing modules (blue) and intermediate output (red), with arrows indicating data flow. Modules are orchestrated by a central pipe server, which is responsible for allocating work. A central database provides data synchronization.

NewsStand is able to execute as many instances of modules as are required to handle the volume of news received.

A significant challenge in designing NewsStand’s architecture was coordinating its processing modules so that they execute simultaneously without excessive idle time. To address this challenge, NewsStand features two centralized sources of control and synchronization:

1. A *pipe server* tracks documents as they flow through the processing pipeline and assigns documents to processing modules.
2. An *SQL database* stores information about documents and results of each processing stage.

These are shown in the center of Figure 3.1. Each module receives assignments of work from the pipe server, and receives data input and sends data output to the SQL database. Notice that this arrangement decouples NewsStand’s control channel (pipe server) from its data channel (database). Both the pipe server and database have specific communication protocols to which processing modules must adhere. Also, using transactions, the database ensures that the overall system state changes atomically and is never internally inconsistent. Furthermore, the database system can be replicated across multiple nodes as necessary to handle increased system load. For our database, we use the open source PostgreSQL package. The pipe server and its communication protocol are described in more detail in Section 3.3, while the database layout is presented in Section 3.4.

Document processing proceeds as follows. When a processing module instance starts, it connects to both the pipe server and the database. The pipe server then sends a batch of document identifiers, or `docids`, to the module instance. For each document, the module instance retrieves relevant information about that document from the database, performs its processing on the document, and stores the results back in the database. When all documents in the batch have been processed, the module reports back to the pipe server that

the batch is finished, and receives another batch of documents to work on. `docids` are added to the pipe server by the first module in the pipeline, namely the RSS grabber. Note that individual instances of modules are directed by the pipe server to work on independent batches of documents. In this way, processing bottlenecks are avoided by starting additional instances of processing modules that require more processing time, allowing greater scalability. Finally, NewsStand's Web interface accesses the central database to retrieve data for display. Each retrieval action is posed in terms of a corresponding SQL query within NewsStand's database. These queries are detailed in Section 3.6.

Notice that NewsStand's architecture bears some similarity to that of STEWARD (described in Chapter 2), which is not surprising given that it evolved from STEWARD. One key difference from STEWARD is that in addition to a central database, NewsStand has a central pipe server that serves to coordinate all modules by passing `docids` between modules, thus controlling the work flow. In addition, NewsStand has many more modules than STEWARD, and in particular an online *clustering* module, which all contribute more evidence to improve geotagging. Also, note that NewsStand's architecture can be modeled as a directed graph, where nodes are modules and links describe data flows between modules. This model of NewsStand's operation is similar to that of other distributed data flow architectures, such as that of Dryad [57]. NewsStand differs in that it has a central pipe server and database, while Dryad and other distributed systems tend to avoid central control points, which are essentially single points of failure. However, the central synchronization database makes the data processed by NewsStand much easier to access with more traditional applications, at the expense of some scalability. In addition, one potential drawback in NewsStand's architecture is that as a work pipeline, the system's continued functioning relies on all modules in the pipeline working properly and reliably, since later modules necessarily depend on output from earlier modules. However, this is mitigated somewhat by the high degree of automation in monitoring NewsStand's system status through its pipe server protocol (described in Section 3.3).

In subsequent sections, we present NewsStand’s architecture in more detail by describing its processing modules, pipe server, and database layout. Knowing the details of these components will clarify the evidence available to NewsStand’s geotagger, and aid in understanding our toponym recognition and resolution algorithms (described later in Chapters 4–7).

## 3.2 Processing Modules

In this section, we provide more detail about the workings of NewsStand’s many processing modules, illustrated in Figure 3.1. Knowing these details will aid the understanding of NewsStand’s database design (described in Section 3.4), as well as our toponym recognition methods (Chapter 4) and toponym resolution algorithms (Chapters 5–7). NewsStand’s processing modules include:

1. *RSS Grabber*: Polls RSS feeds and retrieves URLs to news articles.
2. *Downloader*: Downloads HTML news articles from URLs.
3. *Cleaner*: Extracts article content from source HTML.
4. *Clusterer*: Groups together articles about the same story.
5. *Topic Classifier*: Assigns general topic types to articles (e.g., “Business”, “Sports”).
6. *Geotagger*: Finds textual mentions of geographic locations and assigns lat/long values to each.
7. *Disease/People Finder*: Finds textual mentions of people, diseases, and other entities.
8. *Media Extractor*: Extracts images and videos, and captions associated with them.
9. *Image Clusterer*: Finds clusters of images and also detects duplicate images within news clusters.

Below, we provide more detail about each module type.

### 3.2.1 RSS Grabber

Myriad reputable newspapers, news organizations, and blogs make their news and commentary publicly available on the Web. However, automating the collection and standardization of large volumes of news articles from such a diverse array of sources can be challenging. While the Web is certainly an abundant source of news, the various sources of news are by no means uniform. Articles from major newspapers are generally well-formatted and internally consistent, while the quality of news from blog websites may be suspect. At a lower level, news articles may be written in a variety of languages, and may be stored in different character encodings. Also, with few exceptions, the majority of newspapers tend to be local in scope, and thus mostly publish stories about a limited geographic area. Thus, we must be concerned with collecting stories from news sources geographically situated all over the world, and not just from the largest or most-circulated news sources.

To address these issues, NewsStand uses a large set of *Really Simple Syndication (RSS)* feeds as its primary source of data. RSS is a widely-used XML protocol for online publication and is ideal for NewsStand, as it requires at least a title, short description, and web link for each published news item. RSS 2.0 also allows an optional publication date, which helps determine the age and freshness of stories. By using RSS, we need not extract story metadata from news articles themselves, which may be difficult due to inconsistent webpage formatting among different news sources.

Retrieving data from RSS feeds is the task of NewsStand's first module, its *RSS grabber*, which connects to news source websites' RSS feeds and retrieves a list of URLs pointing to news articles to be downloaded. To collect the set of RSS feeds, NewsStand's RSS feeds were bootstrapped by crawling several aggregation websites that contained lists of newspapers, along with links to their websites and the newspaper names. Then, each newspaper website was crawled in turn to search for links to the newspaper's RSS feeds. Each found RSS feed was added to NewsStand's database, along with the newspaper name and feed name, if applicable. Many newspapers had multiple RSS feeds corresponding to indi-

vidual sections of the newspaper (e.g., top stories, international, business, sports...), and each was added as a separate feed to NewsStand's database for use by the RSS grabber. Oftentimes the same article appears in multiple feeds from the newspaper, a form of redundant information. This procedure resulted in a set of over 10,000 active RSS feeds from online news sources from all over the world.

Several design criteria went into NewsStand's RSS grabber. The first and most important criterion for the RSS grabber is to retrieve news in a timely fashion, as soon as possible after the time of publication. This is necessary because stories may continually change and be updated, even after they have been published in an RSS feed, which may result in mismatched story metadata if they are not processed quickly. However, quickly retrieving articles using RSS is problematic due to the nature of RSS as a "pull" protocol, since it requires the retrieving client to initiate the data transfer. As a result, an RSS client must constantly poll the RSS feed to determine whether any new data is available. On the other hand, NewsStand's RSS grabber must poll thousands of feeds, which takes time and bandwidth, so the naive solution of a tight polling loop is not suitable. Ideally, the RSS grabber would adapt to each RSS feed, polling more responsive and faster-moving RSS feeds more quickly, while polling slower feeds less frequently, and dropping bad RSS feeds altogether. However, the poll time cannot be too infrequent, since RSS feeds usually limit the number of articles that can be retrieved at a time to 10 or 15 articles, to prevent resource exhaustion.

NewsStand's RSS grabber was designed using these guidelines. For each RSS feed  $f$  in NewsStand's database, the RSS grabber records the last time that  $f$  was polled, as well as a poll interval for  $f$  which indicates how long to wait before polling  $f$  again. The RSS grabber proceeds by retrieving a list of feeds whose poll time and poll interval indicate that the feed is ready to be polled. When the RSS grabber polls a feed  $f$ , it checks to see whether any new article URLs were retrieved by comparing the URLs found in the poll against existing entries in NewsStand's database. New URLs are added to the database, assigned doc ids, and reported to the pipe server (described in Section 3.3) for further

processing. The RSS grabber then updates the poll interval for  $f$  via a simple *exponential backoff* feedback scheme, inspired by those used in networking protocols [112]:  $f$ 's poll interval is multiplied by a random value selected uniformly from  $[0.5,0.6]$  if new data was found, and  $[1.0,1.2]$  otherwise. Note that currently, the RSS grabber makes no provision for articles that were previously published on the Web and hence added to NewsStand's database, but were since updated.

This adaptive polling scheme captures many desirable qualities for NewsStand. Over time, the polling interval for a feed  $f$  will tend to converge on and remain near  $f$ 's update rate, since it is decreased when new data is found, and increased otherwise. This has the effect of polling each RSS feed at an appropriate rate that minimizes wasted bandwidth, while still ensuring that news is retrieved in a timely manner. Figure 3.2 shows article counts for RSS feeds in NewsStand compared to the RSS grabber's poll interval for each feed. Each data point corresponds to an RSS feed. These counts were collected over one month of news articles in September 2010. As the figure shows, the RSS grabber's poll interval is generally adapted to the article rate of each RSS feed, with poll intervals decreasing with article count, and vice versa.

The random values used for increasing and decreasing poll intervals introduces variations in polling times, so that poll times are spread out, which avoids bandwidth spikes. Furthermore, notice that the multiplicative factor used in lowering a feed's poll interval (i.e., in the range  $[0.5,0.6]$ ) is relatively stronger than the factor used for raising it (i.e., in the range  $[1.0,1.2]$ ). In other words, these value ranges are tuned so that RSS feeds are rewarded faster for having new data than punished for not having new data—that is, they reflect a “forgiving” rather than “vengeful” philosophy. Because RSS polls are limited in the number of articles that are returned from a single poll, this forgiving outlook ensures that as little data from RSS feeds are missed as possible.

As an additional benefit, this scheme allows for the detection and suppression of duplicate or redundant RSS feeds, since on each poll, NewsStand's database is consulted to



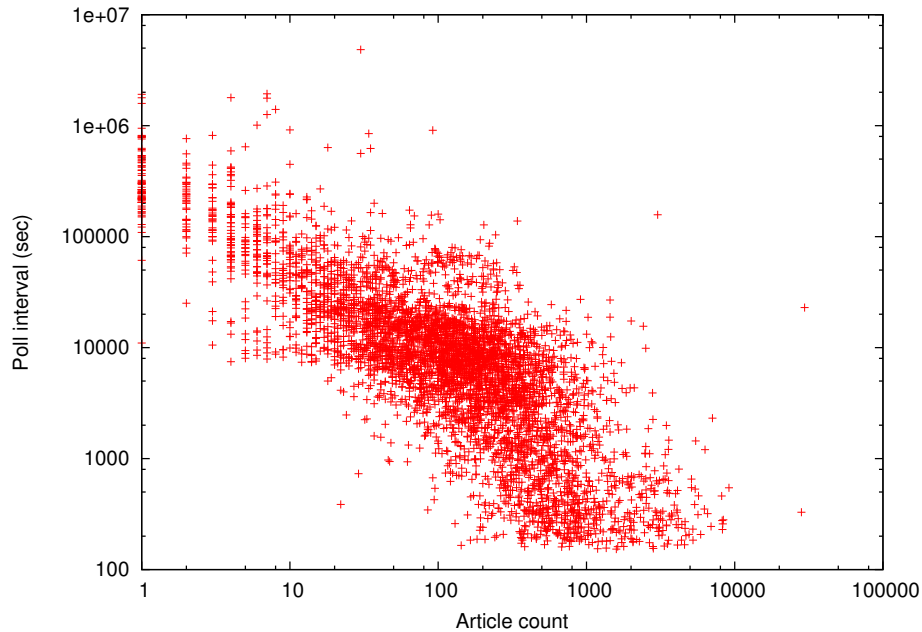


Figure 3.2: Number of new articles posted by RSS feeds versus the RSS grabber’s poll interval, measured over one month of news. Poll intervals are adapted to each feed’s update rate.

determine whether the feed contained any new data. In addition, because poll intervals will tend to match RSS feeds’ update rates over time, these intervals provide a means of quickly ranking feeds based on responsiveness or post speed. For example, users may be more interested in news from a very fast or voluminous source than from a slower source, since a rapidly updating news source may be in closer contact with the “pulse” of the news. As a result, NewsStand can give a higher rank to stories from these fast news sources.

### 3.2.2 Downloader

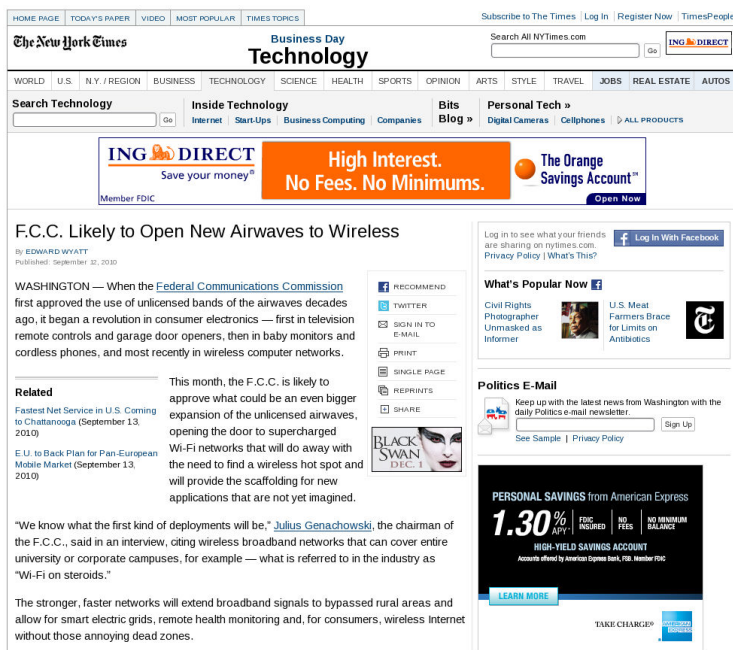
After URLs have been collected and added to the database, NewsStand’s *downloader* is responsible for downloading the HTML pages corresponding to these URLs. The downloader retrieves these pages and stores them in NewsStand’s database. All downloaded pages are reported to the pipe server, which forwards them to later processing modules.

Though the downloader is a relatively simple module, several issues were addressed in its design. In particular, at times the URLs gleaned from RSS feeds are actually redirects,

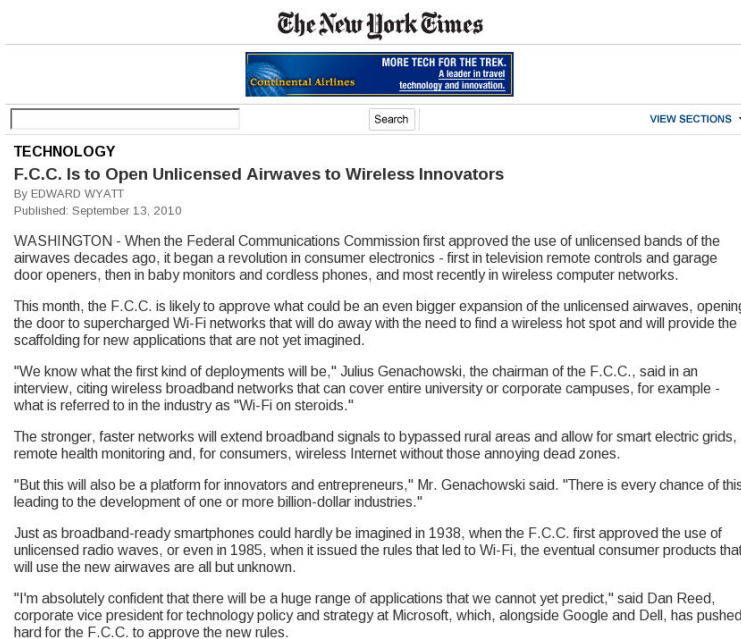
pages that serve only to redirect the accessing browser to the true URL of the news article. Redirects are a common occurrence in RSS news aggregators, since they collect URLs from many websites and serve links to them, and furthermore allow aggregators to track the popularity of links served by them. If a redirect is found, the downloader retrieves the URL at which the redirect points, and stores it explicitly in the database.

Another issue with downloading articles from many thousands of news websites is that websites serve data in different data encodings (e.g., ISO-8859-1, CP-1252, UTF-8). Furthermore, many websites report different data encodings via HTML meta tags than they actually use. For example, one commonly reported encoding is ISO-8859-1, while the actual content served is encoded with CP-1252. These encoding mismatches cause problems when attempting to standardize document data for later processing stages. The downloader attempts to determine each page's encoding, and convert the data to UTF-8, so that it can be stored in NewsStand's database. If the conversion fails, the document is dropped from NewsStand.

Finally, note that websites often cater to different users by serving different versions of pages depending on the particular browser and platform. For example, a news article in the New York Times may be accessed by a multitude of mobile devices (e.g., BlackBerry, iPhone, Android, numerous others) in addition to traditional desktop users. In particular, versions of webpages intended for mobile devices tend to be simpler in structure and contain less extra content, such as advertising and links to related pages, due to mobile devices' premium on screen space. Figure 3.3 illustrates the differences between desktop and mobile versions of a news article [161] published by the New York Times in September 2010. Notice that the large advertisements present in the desktop version are missing from the mobile version, which has only a single, small advertisement. In addition, links to other articles, pages, and websites are not present in the mobile version. Clearly, the mobile version has a much simpler structure and little extra content other than the article itself, which makes automated processing of these articles much easier. NewsStand's downloader takes



(a) Desktop



(b) Mobile

Figure 3.3: Two versions of an article [161] from the New York Times published in September 2010, demonstrating the different pages served to (a) Mozilla Firefox on a Linux desktop and (b) the same browser, but masquerading as a BlackBerry device. The large advertisements and multitude of links present in the desktop version are absent in the mobile version, which eases the cleaner's task.

advantage of this by masquerading as a mobile device, so that downloaded pages will have a simpler structure, and will be easier to process by later modules (in particular, the cleaner module, described in Section 3.2.3). Note that such masquerading does not limit the number of news articles that we retrieve, and only affects the method of retrieval and resulting content delivery. We found that for most news sites, the major difference was that advertising and distracting links were significantly reduced, while article content did not change significantly between the versions.

### 3.2.3 Cleaner

After an article is downloaded, it is next processed by NewsStand’s *cleaner* module, which serves to extract the content from the article, thereby “cleaning” it for later stages of processing. This cleaning process allows later modules to not be led astray by the many types of content present in HTML pages, such as advertising, links to irrelevant content, and reader comments. The latter are especially problematic due to comments often mentioning elements from the story itself, though in a much noisier manner (e.g., with misspellings) which can mislead document processing algorithms. Note that humans have little trouble picking out content from HTML using visual characteristics, since HTML is essentially a visual medium. However, for an automated algorithm, finding content in a raw string of HTML is much more difficult. In addition, due to the large and ever-growing number of news sources on the Web, and since each individual website places content within the page differently, it is infeasible to create custom extraction rules for each website, so the content extraction algorithm must work regardless of the source website.

Many algorithms have been developed for extracting content from HTML documents (for an overview, refer to Gupta et al. [47]). One common approach (e.g., [38, 83, 90]) is to search for long sequences of unbroken text, or sequences with relatively few HTML tags, and some success has been reported for a variety of websites. However, these algorithms have little sense of context, in that they do not understand or make use of the article’s textual

content itself. That is, they are overly general, rather than tuned for cleaning news articles. For example, news articles are generally written in the *inverted pyramid* style [131], with the first several sentences containing most details needed to understand the article, as well as features such as the article's dateline, if present. Therefore, from a utility standpoint, it is more important to capture the first few sentences of the news article correctly than the remaining portions of the article.

NewsStand's cleaner takes advantage of knowledge about the article's content obtained with metadata about the article. In particular, the cleaner uses information contained in the RSS feed from which the article was discovered. Initially, strings of text are chosen using high-confidence evidence from the RSS feed, and uses successively lower-confidence evidence based on what was previously found. In this way, the cleaner's content extraction algorithm is analogous to seed-filling, with initial seeds of content chosen with high confidence, and the seeds used to find lower-confidence strings of text.

To begin, the cleaner generates a parse tree of the article's HTML. Each node in the tree corresponds to an HTML element. In addition, strings of text in the article correspond to a subset of leaf nodes in the parse tree. The cleaner's task is then to select the nodes of the parse tree that correspond to the article's content, and to ignore the nodes that are not. One complication is that article sentences do not correspond directly with individual leaf nodes. That is, a single text node in the parse tree may contain a single word, a part of a sentence, a single sentence, or many sentences, depending on the website's page layout. For example, if the article contains a sentence that contains one or multiple links to other HTML pages, the sentence's text will be broken across multiple text nodes in the parse tree.

After generating a parse tree from the article's HTML, the cleaner algorithm proceeds by collecting a set of *anchor nodes*, which serve as an initial, high-confidence set of text nodes that are likely part of the article's content. Put another way, the anchor nodes serve as the initial seeds for our flood-filling content extraction algorithm. The cleaner finds anchor nodes by searching for keywords gleaned from article metadata present in the RSS feed

entry for the article. In particular, the article's title and description fields are used for this purpose. As a result, these keywords are article-specific, and since they are of limited size, they are highly tuned to the important content of the article. Any text nodes in the parse tree that contain a keyword and are of sufficient length are likely highly related to the article's content, and are added to the set of anchor nodes.

Note that the RSS metadata is very concise and is not likely to contain enough keywords to capture all the article's content. This process might find text nodes near the article's beginning, due to the inverted pyramid structure, but later details will be missed. To find more detailed text, the cleaner next draws upon the anchor nodes to find a larger set of lower-confidence *adjunct nodes*. These adjunct nodes are found by collecting additional keywords from the anchor nodes, and searching for these keywords among the remaining nodes. Text nodes containing enough keywords are added to the set of adjunct nodes.

Finally, to gather the remaining article content, the spatial characteristics of the anchor and adjunct nodes are used. In particular, the cleaner leverages the fact that most text in the article's content will be nearby within the structure of the parse tree. Accordingly, nodes that are nearby the anchor or adjunct nodes in terms of the parse tree, termed *proximate nodes*, are collected to be part of the content. Similarly, nodes that are "sandwiched" between anchor or adjunct nodes, termed *sandwich nodes*, are likewise collected. To produce a final cleaned text, the cleaner gathers all anchor nodes, adjunct nodes, proximate nodes, and sandwich nodes, and sorts them by their starting offsets in the document. The text is then extracted from these sorted nodes, and stored in NewsStand's database.

Note that to be effective, the cleaner relies on accurate RSS metadata. As a result, it depends on the RSS grabber (described in Section 3.2.1) and downloader (Section 3.2.2) having been executed on the processed articles in a timely fashion. If an article is downloaded long after it was entered into NewsStand via the RSS grabber, a later, updated version of the article with different title and content may be retrieved, which could limit the cleaner's effectiveness. For example, title changes to articles are common as the story evolves and

more details are revealed, which could be problematic if the new article’s title (retrieved with the downloader) does not match the old (retrieved by the RSS grabber).

In addition to the above algorithm, NewsStand’s cleaner is also enhanced by several means. Websites often have text that is part of the site layout, and is common to a large number of pages. To avoid considering this text as potentially part of article content, part of NewsStand’s cleaner collects and incorporates website-specific *stop words*, which are sequences of text that appear in many articles retrieved from the website. Any text nodes containing these stop words are removed as potential candidates for the content extraction. In addition, to provide for better content extraction, NewsStand executes the cleaner twice in the article processing sequence: once before the clusterer (see Section 3.2.4), and again afterward. This multiple execution allows the cleaner to make use of *cluster terms*, keywords which appear in many articles in the cluster. In particular, in addition to the RSS metadata, cluster terms are used for the initial collection of anchor nodes. As a result, the set of anchor nodes is more complete, which provides a more complete extracted content. Finally, as mentioned earlier, the downloader (see Section 3.2.2) masquerades as a mobile device, since some websites will serve simpler pages, with less advertising and other irrelevant items, to mobile devices. This simplifies the cleaner algorithm’s task, since there are fewer potentially false-positive text nodes to consider.

### 3.2.4 Clusterer

After being passed through the cleaner, the resulting content text is next used by NewsStand’s *clusterer* [144] to group it with other articles containing the same story. Broadly, a news story is defined in terms of both *story content* and *story lifetime*—that is, articles in the same cluster should share important keywords, and should have temporally proximate dates of publication. Time is an essential part of grouping news articles, since two articles may contain similar keywords but describe vastly different news events. For example, two stories about separate attempted assassinations in Iraq may share many keywords,

but should be placed in separate clusters if one story was breaking news and the other was several days old. In addition, we want new or breaking articles to be clustered quickly, so that breaking stories can be presented immediately to users.

This speed requirement precludes the use of traditional, offline approaches to clustering. For every new article downloaded, the entire news collection would have to be clustered again, incurring unacceptable performance penalties for voluminous news days. Instead, NewsStand’s clusterer takes an *incremental* or *online* approach to clustering that reuses existing clusters, and requires significantly less computation time. Furthermore, the clusterer uses the above temporal constraint and several optimizations to effect fast processing of thousands of articles per day. The potential drawback of the online approach is that since it uses imperfect information, the resulting clustering may not be of the same quality as an offline counterpart. However, we found that NewsStand’s output clusters were of generally good quality.

Upon receiving a new article to be clustered, the clusterer first normalizes the article’s content by *stemming* [115] input terms and removing punctuation and other extraneous characters. It then generates a *feature vector* from the document’s text via the well-known *TF-IDF* score [125] for each term in the article. This score emphasizes those terms that are frequent in a particular document and infrequent in a large corpus  $D$  of documents. For the corpus, the clusterer simply uses the collection of news articles present in all current clusters in NewsStand. Note that even though this corpus constantly evolves with each new article processed, the clusterer computes the feature vector for a particular article only once, upon its addition to the system, for performance reasons. In practice, this optimization does not affect clustering noticeably.

The clustering algorithm is a variant of *leader-follower clustering* [36] that permits online clustering in both the term vector space and the temporal dimension. For each cluster, we maintain a *term centroid* and *time centroid*, corresponding to the means of all feature vectors and publication times of articles in the cluster, respectively. To cluster a new arti-



cle  $a$ , we check whether there exists a cluster where the distance from its term and time centroids to  $a$  is less than a fixed cutoff distance  $\epsilon$ . If one or more candidate clusters exist,  $a$  is added to the closest such cluster, and the cluster's centroids are updated. Otherwise, a new cluster containing only  $a$  is created. The clusterer uses the *cosine similarity measure* [139] for computing term and time distances between the new article and candidate clusters. In addition, the clusterer computes *cluster terms* for each cluster, which amount to the most heavily weighted keywords in the cluster. These cluster terms are used in several later modules as a summary of the important content in the cluster.

To improve performance, cluster centroids are stored in an inverted index that contains, for every term  $t$ , pointers to all clusters that have non-zero values for  $t$ . The clusterer uses this index to reduce the number of distance computations required for clustering. When a new article  $a$  is clustered, we compute the distances only to those clusters that have non-zero values in the non-zero terms of  $a$ . As a further optimization, we maintain an *active list* of clusters whose centroids are less than a few days old. Only those clusters in the active list are eligible to receive a new article. We remove clusters from the active list after several days, since the values for the distance function will be negligible. Together, these optimizations allow the clustering algorithm to minimize the number of distance computations necessary for clustering articles.

One interesting observation is that the act of online clustering is strongly analogous to a newspaper editor's job of deciding whether a story is new enough to deserve its own headline, or not. In the paper publishing model, editors make the decision whether to run a story with even a small amount of detail. This makes breaking news difficult to report accurately since details could change in the course of printing (hence, "stop press!"). On the other hand, in the case of online news publishing, updates are essentially free, and the change can be seen by the news audience immediately. When a story is first reported, it tends to be reported with a smaller amount of detail, especially in the case of breaking news. As more details are revealed and the story evolves, editors must decide whether to

simply update the already-run story with the newer details and post a notice that the story has been updated, or whether the new details that have been revealed deserve their own headline, in their own right.

Our online clustering algorithm makes the same decisions, but on a global scale, across all news feeds. In NewsStand, each cluster can be thought of as a single story. Individual articles, part of a cluster, might be thought of as sentences or paragraphs, part of a news story. When a new article is added to NewsStand, the clusterer decides whether there is enough overlap with an existing cluster (i.e., an existing story), and if so, simply adds the article to that cluster (i.e., adding more sentences to the story). On the other hand, if the article is different enough in content and detail from NewsStand’s current set of clusters, a new cluster is created (i.e., publishing a new story). In essence, the online clusterer acts as an editor for a single global newspaper consisting of all newspaper articles aggregated by NewsStand.

### 3.2.5 Topic Classifier

One module that makes use of clustering information is NewsStand’s *topic classifier*, which takes each article and assigns a general topic according to the type of article (e.g., business, health, sports). These article types generally correspond to the newspaper section in which the article would appear. Assigning topics to articles allows filtered browsing of articles based on a particular topic. Topics are also useful for further automated processing of articles, since they provide a preliminary indication of the type of content that might be present in the article. For example, a mention of “Toyota” in a business article is more likely to be a car company, rather than a city east of Nagoya, Japan. Similarly, in a sports article, a mention of “Barcelona” might be the name of a soccer team, rather than a city in Venezuela. The topics used in the topic classifier include business, science/technology, entertainment, health, sports, and general (the default topic, in case none of the others matched well enough).

To compute topics for articles, the topic classifier uses a *naive Bayes classifier* [61], well-known in the text classification literature. The naive Bayes classifier chooses the most likely class for a document using words in the document as features, and furthermore, assumes independence among the words. That is, the classifier does not consider dependencies between words in natural language, which considerably simplifies the classification model, and speeds up training and classification. Despite these rather strong independence assumptions, it has been shown to work well for text classification. This speed is useful in the case of NewsStand, since we wish to process news and make it available as quickly as possible. For more details of the naive Bayes classifier, refer to Jurafsky and Martin [61]. NewsStand’s topic classifier was trained on a manually-created corpus of articles. The classifier generally works well, so these topics can be used as a high-confidence source of evidence in geotagging and other modules.

In addition, after selecting a topic for an individual article, an overall topic for the article’s cluster is also selected as the most common topic present in articles in the cluster. In this way, an overall cluster topic is found, which is used by later modules as well as the user interface when querying for news by topic.

### 3.2.6 Geotagger

After clustering, NewsStand’s *geotagger* module executes toponym recognition and resolution on the article to find and resolve locations in the document. In addition, since each article is associated with a cluster, the geotagger selects the most prominent locations present in articles in the cluster and associates them with the cluster. Since geotagging comprises a significant portion of this dissertation, we defer descriptions of the geotagger’s toponym recognition and toponym resolution methods to Chapter 4 and Chapters 5–7, respectively.

### 3.2.7 Disease Finder / People Finder

Two other of NewsStand's modules are its *disease finder* and *people finder*, which search for mentions of diseases and person names in articles, respectively. These modules allow for querying of diseases and people, and, in combination with NewsStand's geotagger, finding their association with particular locations via the news. In addition, they allow for querying trajectories of particular diseases or people. For diseases, trajectory querying corresponds to tracking the incidence, outbreak and spread of various diseases, by observing the locations in which a disease is found over time. For people, we might be interested in tracking politicians as they travel and visit various areas. In addition, finding mentions of people and diseases can be used to benefit geotagging as well, by filtering out location interpretations. As with several other of NewsStand's modules, the disease and people finder evolved from corresponding functionality in STEWARD, in particular STEWARD's disease monitoring capabilities (described in Section 2.5). In its disease monitoring role, NewsStand bears some similarity to systems such as BioCaster [29], HealthMap [40], and EpiSPIDER [149, 150], though these latter systems do not offer the same levels of ontological detail and dynamic querying capabilities as those in NewsStand.

To find diseases, the disease finder searches the article's text for entries in a disease lexicon. This lexicon was created by collecting lists of diseases from Wikipedia [157] as well as the Centers for Disease Control and Prevention (CDC) diseases and conditions website [25]. In total, the disease lexicon contains about 150 diseases and conditions, which generally correspond to large families of diseases. Of course, the number of individual diseases and conditions is ever-growing and numbers far more than 150. For example, the human disease ontology available from the Open Biological and Biomedical Ontologies Foundry [107] contains over 14,000 diseases. However, many of these diseases are never reported in the news, since the news is intended for a more general readership, rather than medical specialists. In addition, the disease lexicon could be augmented with additional, more specific names for more specialized use. When searching in article text for names

from the disease lexicon, the search is case-sensitive for acronyms (e.g., “HIV”), and case-insensitive otherwise. Output from the disease finder is displayed in NewsStand’s disease layer (shown later in Figure 3.10a).

Note that diseases, like many other entities, exhibit various forms of ambiguity. In particular, certain disease specifications might refer to any of a family of diseases. For example, “cancer” is frequently mentioned in the news even though there are many types and subtypes of cancer. At times, this ambiguity is intended, as when a person specifically refers to the entire family of diseases, but other times it may not be. The disease finder sidesteps these general ambiguity problems by dealing with a small, high-level disease lexicon. That is, considering a hierarchical ontology of diseases, only nodes that are near to the root are included in the lexicon, i.e., “cancer” is recognized without regard to its particular type. In a geotagging context, this would be equivalent to using a small, unambiguous gazetteer.

Similar to the disease finder, the people finder’s task is to search for person names in news articles. This module finds prominent people such as national politicians, sports figures, and celebrities, but also searches for less-prominent or relatively unknown people that often appear in news. As a result, the people finder’s task is more complex than that of the disease finder, since people vastly outnumber diseases in the news and in general. That is, there are a much greater variety of person names, in many different combinations, than disease names. Thus, a small, limited lexicon approach is not applicable here.

However, it is not enough to simply find mentions of people; to be useful, the people finder needs to merge repeated mentions of the same person into the same entity, a task known as *coreferencing*. As with other problems in entity recognition, finding people in text is hampered by various forms of ambiguity. A given person can be referred to in several forms, even within the same article. For example, an initial mention of President “Barack Obama” may be followed by mentions of “Obama” only, even though both instances refer to the same person. In addition, if the article’s text contains both “Barack Obama” and “Michelle Obama”, it may not be clear to which person “Obama” refers. For the people

finder, we assume that a mention of a lone surname refers to the first person mentioned in the article with that surname, i.e., the first person’s name with matching suffix. This assumption is consistent with the inverted pyramid style of news articles, which recommends introducing important details early, and proscribes redundant information.

The people finder’s basic algorithm is to search for particular cue words that signal person entities. Once an initial set of person entities are found, surnames are derived by collecting the last word of each entity. This step is necessary because news articles often refer to individuals by surname only. Finally, to collect person entities which may have been missed by the initial cue word search, a search is performed for these surnames throughout the article. This final search collects person entities that are not flagged by a cue word. Table 3.1 contains a set of cue word classes used by the people finder, along with some illustrative examples of each class. Notice that cue words can appear as both prefixes (e.g., honorifics, job titles) and suffixes (e.g., generational suffixes, some declaratory words). In addition, both full and abbreviated forms of cue words are included in the search (e.g., “Senator” and “Sen.”). Spelling and usage differences also factor in the search. For example, in US news sources, the abbreviation “Mr. X” is commonly used, while British sources often refer to “Mr X” without a terminating period. Finally, another consideration is that some cue words are considered part of the person entity (e.g., generational suffixes, given names), while others are not (e.g., honorifics, job titles). The list of given names was obtained from the Social Security Administration’s popular baby names website [137]. Output

Table 3.1: Cue words used by the people finder to discover person entities.

Cue Type	Examples
Honorifics	Mr. X; Ms Y; Dr. Z
Generational suffixes	X, Jr.; Y III
Postnominals	X, KBE; Y, M.D.
Job titles	Sen. X; President Y; Sgt. Z; Det. W
Declaratory words	X said; added Y
Common given names	John X; Michael Y; Jennifer Z; Lisa W

from the people finder is shown in NewsStand people layer, which will be described later (shown in Figure 3.10b).

After finding disease and people mentions in an article, the cluster associated with the article is reprocessed. The most frequent disease and person occurrences in articles in the cluster are collected and associated with the cluster. An additional optimization is that the cluster is not reprocessed in this manner if its size has not changed since it was last processed. This can happen if the disease finder or people finder is given multiple documents from the same cluster in the same work batch. In this case, reprocessing the cluster would be redundant work, which would substantially retard article processing speed for large clusters. Finally, note that the disease and people finders are currently limited to the English language due to the language-specific disease lexicon and cue words. However, lists might be gathered and used for other languages.

### 3.2.8 Media Extractor

Another module that executes after clustering is NewsStand's *media extractor*, which processes the HTML versions of articles in the cluster by searching for various types of embedded media that are related to the story, including images, video, and audio clips such as sound bites. While seemingly trivial, websites contain many types of media that are unrelated to the news story. For example, the vast majority of images on a typical news article's webpage are not images related to the news story, but instead are related to advertising, page layout, and visitor tracking (i.e., Web bugs [94]). Thus, the media extractor must determine which images, video, and audio clips are associated with the news story, and filter out the rest. In addition, media found in news articles tend to have associated metadata in the form of image and video captions that provide more information to readers about what is contained in the media. The media extractor must find these captions and associate them with the media in question. Doing so enables the retrieval of media independently of the article from which it was obtained. Finally, webpage structure varies significantly among

websites, and different websites embed media in different ways. For example, video embedded in news articles may be using any of several video plugins and services, each with their own structure and parameters. In addition, image captions may be associated directly with an image via an HTML attribute, or may be present in a nearby paragraph. Ordinarily, for humans, this structure variation does not cause a problem for finding media captions, since humans find these captions by relying on visual and contextual characteristics (i.e., knowing that the caption appears nearby to the image, and knowing the entities that appear in the image and finding them in the caption’s text). However, an automated algorithm such as the media extractor must account for this varied structure.

Finding images is relatively simple, in that the media extractor simply searches for HTML `<img>` tags, each of which corresponds to an image. To find image captions, the media extractor checks for a `title` or `alt` attribute (indicating text explicitly associated with the image), or a parent container element (e.g., `<div>`, `<p>`, `<a>`) containing a moderate amount of text. The text associated with the image using the above procedure will serve as the image’s caption. However, this process results in many spurious images that are unrelated to the news article’s content. To filter out these unrelated images, the media extractor uses the cluster terms computed by the clusterer (described in Section 3.2.4). Only images with associated text containing at least one of the cluster terms are kept, and the remainder are dropped. Figure 3.4 is an example of an image found by the media extractor, along with its associated HTML code. Notice that the image caption is present in the `alt` attribute of the image’s HTML code. This image was selected due to its caption having several cluster terms, namely “Hurricane Igor”.

Extracting video and audio clips are more involved, because methods for embedding these types of media are much more varied than for images. Different websites use different tags and software plugins to display video and audio, each customized for the particular website’s visitors, and also with captions embedded in a variety of ways. As a result, the media extractor currently recognizes video on a per-site basis, with each website for which



# Hurricane Igor rapidly strengthens

Sunday, 12 September 2010

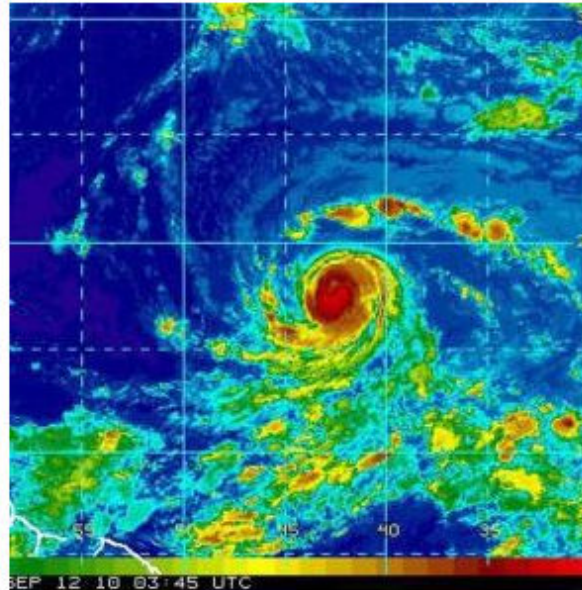
SHARE | PRINT | EMAIL | TEXT SIZE

Forecasters in the US have warned that Hurricane Igor has rapidly strengthened to a category four storm in the open Atlantic, though it doesn't immediately threaten land. Igor has maximum sustained winds of 135mph and is moving west at 14mph. Some additional strengthening is expected in the next two days.

The centre of the storm is located about 1,120 miles east of the Northern Leeward Islands.

Meanwhile, a tropical depression off the coast of Africa has prompted officials to issue a tropical storm warning for parts of the Cape Verde Islands.

Forecasters warned the newly formed depression could strengthen into a tropical storm as early as Sunday night. The storm warning was issued for the southern Cape Verde Islands, including Maio, Sao Tiago, Fogo and Brava.



Hurricane Igor has become the fourth hurricane of the Atlantic season with maximum winds near 75mph (AP)

ENLARGE

(a) Story image

```

```

(b) HTML code

Figure 3.4: An image extracted from a news article [16] about Hurricane Igor in September 2010. Notice that the caption is present in the image's alt attribute.

video are recognized requiring its own recognition pattern. This arrangement is not scalable for large numbers of websites due to the manual effort involved. However, we can still capture a good number of video and audio by noting that many news websites are local affiliates of large television and news networks (e.g., in the US, ABC, NBC, CBS), and use the same infrastructure to serve video and audio. That is, these local affiliates use the same software plugins, with the same, consistent parameters, so the recognition pattern used for, say, WUSA9, a CBS affiliate in Washington, DC, can also be used for the CBS 2 station in New York. In the future, greater acceptance of current HTML standards may allow for easier recognition of embedded media (e.g., `<video>` and `<audio>` tags). For captions, video and audio captions are generally present as parameters in the applet tag, and like the image extraction, one or more cluster terms must be present in the caption for the video to be accepted.

Because news websites often update articles, links to media sometimes become stale because the corresponding media is removed or changed. To account for these potentially stale and broken links, NewsStand also has a media cache module, which downloads and stores media locally on NewsStand's servers. In addition, recall that NewsStand's downloader module masquerades as a mobile device when downloading articles, so that NewsStand receives simpler versions of webpages that are easier for automated processing (see Section 3.2.2). While this is useful in processing the article's text, one potential consequence is that there may be fewer instances of media present in pages intended for mobile devices, due to mobile platforms' limitations in screen size and power, and hence the amount of media served by NewsStand could be likewise limited. However, we found that over a month's worth of news in September 2010, consisting of about 1.6 million articles, over 280,000 images were downloaded, quite a sizable number.

### 3.3 Pipe Server

As noted earlier, NewsStand's modules are orchestrated by a central, master *pipe server* that serves as the control system and is responsible for delegating work to NewsStand's processing modules. It maintains a collection of work queues called *pipes*, with one pipe per module type. Each pipe contains a number of document identifiers, referred to as `docids`, which correspond to documents moving through stages of processing. Each processing module connects to the pipe server to receive work batches of `docids` that are intended for an instance of that module type.

Several factors influenced the pipe server's design. First and foremost, being the central controlling software in NewsStand, it must be highly reliable, and never go down or crash. It should have reasonable memory usage and not have significant external dependencies, such as requiring a full-fledged database to function properly. Furthermore, it should be resilient to unreliable processing modules, which could disconnect at any time, either explicitly, or due to software bugs or networking problems. All of these cases must be handled gracefully. In addition to reliability, speed of processing is a key factor in the design. Because new articles are constantly streaming in to NewsStand, the pipe server must not be a bottleneck in articles' processing time.

The pipe server and its communication protocol (described in Section 3.3.1) have several features that address these goals. First, the pipes and the `docids` contained in them are stored in a disk-based hash, which does not depend on NewsStand's database. As a result the `docids` move very quickly through the pipes so that they can keep up with the very rapidly incoming new data. Also, when communicating with processing modules, rather than waiting for immediate responses from each module, which could slow processing, the pipe server employs non-blocking input/output and buffering. Each connected module is tracked individually with regard to the work batch sent to it, and this work does not move to the next pipe until the module sends back a valid work complete response. Note that this design assumes that modules are not malicious (e.g., reporting that work was finished

when it was not). The main drawback of this design is that the pipe server amounts to a single point of failure. If the pipe server does halt, NewsStand's processing will also cease. However, over months of measurement, we found that the pipe server's stopping was due to rebooting the server on which it runs, rather than reliability issues introduced by its design.

Note that the `docids` traveling through these pipes correspond to documents in NewsStand's database. However, it is interesting that from the perspective of the pipe server, these `docids` are simply numbers to be tracked, because the pipe server does not connect to the database directly for reliability reasons. That these numbers correspond to database documents is incidental to its operation.

Upon creation, processing modules connect to the pipe server and initiate a handshake that announces an instance of the module's presence and in what role the instance will function. The pipe server then pushes a block of `docids` to be processed to the module, and waits for a return message indicating the work has been finished. If no such response is received after a set time limit, the pipe server assumes that the module instance somehow failed. The pipe server then requires the failed module to resend the handshake before it will delegate additional work to that module. However, if the return message is received in time, the pipe server forwards the completed `docids` to the next set of work pipes, as determined by NewsStand's data flow graph. In addition, along with the return message, the pipe server protocol allows sending a list of `docids` for which processing somehow failed. If such a list is present in the return message, the pipe server drops the failed `docids` from the system, effectively ending processing for these `docids`, the failed `doc ids`, and forwards the remaining `docids` to the appropriate work pipes.

In addition to handling the distribution and flow of work, the pipe server serves a number of other useful functions within NewsStand. For each module type, the pipe server tracks statistics such as the total number of documents processed by the module and the number of documents that failed processing (i.e., which were dropped as a result of that module), and allows for individual modules to track custom statistics by reporting them

along with work completion messages. These statistics are useful for diagnosing problems with individual modules (e.g., determining that a particular module is dropping too many documents), or for learning about the nature of the data flowing through the system and how algorithms interact with the data (e.g., for the geotagger, learning which types of evidence are most useful in toponym recognition and resolution). In addition, the pipe server’s “info” function within the protocol exposes the list of modules that are connected, as well as their work status and system-specific information such as process id and username that executed the module instance. The info function allows for a high level of automation in monitoring NewsStand’s system status, such as checking for crashed modules or backed up work pipes. For example, an automated script runs hourly which connects to the pipe server and uses the info function to get NewsStand’s system status. If a work pipe backup is detected, emails are dispatched to alert the system maintainers of this condition. These features allow NewsStand to be easily kept in a consistently running state.

We continue with a description of the pipe server’s communication protocol (Section 3.3.1) as well as the techniques used to check connected clients’ status (Section 3.3.2).

### 3.3.1 Communication Protocol

The pipe server communicates with slaves using a simple protocol that allows for a variety of tracking and maintenance features. The protocol messages themselves are presented in Figure 3.5, formatted in BNF notation, with messages corresponding to  $\langle *msg \rangle$  nonterminals. This section describes the protocol in greater detail and lays out the reasoning behind each message. Slaves that connect to the pipe server must adhere strictly to this protocol, or they are promptly disconnected, along with a message explaining the reason for the disconnection. All slave messages undergo data validation (e.g., ensure that `docids` are integers) for extra caution, and acknowledgment messages are required after each message as well. This policy lies in keeping with our goal of maximum reliability.

```

<module-msg> ::= 'MODULE' slavetype processid username

<insert-msg> ::= 'INSERT' pipename docid...

<work-msg> ::= 'WORK' workid docid...

<response-msg> ::= 'RESPONSE' workid <drop> <stats>
<drop> ::=  $\emptyset$  | 'DROP' docid...
<stats> ::=  $\emptyset$  | 'STATS' key=val...

<info-msg> ::= 'INFO'

<clear-pipe-msg> ::= 'CLEARPIPE' pipename

<reset-stats-msg> ::= 'RESETSTATS' pipename

```

Figure 3.5: The pipe server’s protocol messages, specified in BNF notation, with  $\langle * \text{-msg} \rangle$  nonterminal symbols corresponding to protocol messages.

**Module identification** When slave modules first connect to the pipe server, they identify themselves so the pipe server can assign appropriate work to them (“ $\langle \text{module-msg} \rangle$ ” in Figure 3.5). This information is presented in the `slavetype` parameter. In addition, each slave is required to report the `processid` and `username` under which it is running. Crucially, this extra information enables the quick and easy stopping and starting of slave modules that may be misbehaving or unresponsive, because it is a simply a matter of logging in to the machine from which it is running and killing the reported process id.

**Insert work to pipe** This message (“ $\langle \text{insert-msg} \rangle$ ” in Figure 3.5) instructs the pipe server to add a list of `docids` to a given `pipename`. This is used by the *RSS grabber* module, which retrieves article links from RSS feeds, creates initial entries for these articles in the database (assigning a `docid` in the process), and directs the pipe server to insert the `docid` in the *download* pipe.

**Assign work** Should a pipe have unassigned `docids`, and a slave of that pipe type is not working, then the pipe server sends a batch of work to the idle slave using this message

(“*<work-msg>*” in Figure 3.5). A work batch consists of a `workid`, an identifier for this work batch, as well as a list of `docids` to be processed. The `workid` will be required to be returned along with the work response message (described in the next section).

Since work is sent in batches, one question to consider is how many documents to send out in each batch. Clearly, we want a `blocksize` that maximizes the processing throughput for the system. Since the pipe server protocol’s overhead is minimal compared to the actual document processing, the `blocksize` value will not make much difference when processing individual documents. However, many of NewsStand’s processing modules operate on clusters of documents, rather than on individual documents. As a result, with a larger `blocksize`, there are more opportunities for documents that appear in the same cluster together to be processed at the same time, thus enhancing the system’s throughput. One downside with a large `blocksize` is that some documents and clusters take much longer to process than others (e.g., longer documents with more entities, large clusters), and if too many of these documents are sent in the same work batch, then the slave may time out, causing wasted work and potentially stale data. Currently, NewsStand’s pipe server is configured with a `blocksize` of 100 documents, arrived at through trial and error.

**Work response** When a slave is finished with a batch of work, it sends a work response message (“*<response-msg>*” in Figure 3.5). The message contains at least the `workid` that was originally sent with the work batch. This requirement acts as a sanity check that the slave had finished the same work that was handed out by the pipe server, which may not always happen due to slaves becoming unsynchronized from the pipe server as a result of network instability or other problems. In addition, the work response message can contain optional clauses that expose additional functionality. The first clause is a *<drop>* clause, where slaves can inform the pipe server of documents whose processing resulted in errors of some kind, and should not be processed by later modules in the processing pipeline. If a drop clause is present, then `docids` listed in the clause are dropped from the pipe,

rather than being forwarded to the next pipe. The second optional clause is a *<stats>* clause, through which slaves can report module-specific statistics in the form of key-value pairs, which are aggregated by the pipe server. For example, the geotagging module might track separately the number of toponyms recognized or resolved using each type of evidence. In addition to slave-specified statistics, the pipe server automatically tracks the number of documents that were sent to a specific type of slave, and how many of them were dropped. In this way, slaves that drop too many documents are easily found and investigated.

**Pipe information** The pipe server contains an *info* facility for reporting the current status of pipes, as well as modules connected to the pipe server (“*<info-msg>*” in Figure 3.5). This facility allows for easy monitoring of the system’s status. Figure 3.6 shows the information returned by the *info* command. The pipe server reports for each pipe (Figure 3.6a) the number of documents yet to be processed, the last time a work batch was completed for the pipe, and the number of documents processed versus the number of those passed (i.e., not dropped via a drop clause). In this way, if a pipe is getting clogged (i.e., a large amount of work is waiting to be processed), it can be readily observed and dealt with appropriately (e.g., by starting additional module instances for that pipe). In addition, for each connected module (Figure 3.6b), several data are reported: the module type, hostname from where the module connects, current work batch size, timestamp of when the work batch was assigned (“Gave Work At”), timestamp of when the previous work batch was completed (“Heard At”), and the module’s process id.

In combination with pipe information, these statistics allow for quick diagnosis of any backups in the document flow. Also, since the pipe server reports information in machine-readable format, the pipes and modules are monitored by a script that emails alerts about such backups in the data flow. Because the hostname and process id are reported with each connected module, should the module type need to be restarted, running modules can be stopped cleanly by terminating using the process id, thus preventing resource exhaustion.



Pipe	Size	Last Work Done At	Processed	Passed
download	22	Thu May 26 16:18:44 2011	20531341	15549184
clean	3	Thu May 26 16:18:41 2011	17832420	15349084
cluster	85	Thu May 26 16:17:37 2011	15314622	14522858
cleanafterclust	0	Thu May 26 16:18:31 2011	13482590	13466247
ner	0	Thu May 26 16:18:42 2011	14917601	14917601
topic	54	Thu May 26 16:17:02 2011	14540838	14540838
tag	0	Thu May 26 16:18:40 2011	14918046	14918046
people	45	Thu May 26 16:17:49 2011	14917501	14917501
misc	0	Thu May 26 16:18:42 2011	14255621	14219634
media	45	Thu May 26 16:17:27 2011	14540793	14540793
events	135	Thu May 26 16:18:42 2011	14540703	14389460
diseases	45	Thu May 26 16:17:37 2011	14540793	14540793
aggregator	45	Thu May 26 16:18:01 2011	14540793	14540793
mediacache	0	Thu May 26 16:17:30 2011	10800581	10800581
imageclust	0	Thu May 26 16:17:57 2011	10800581	10800581

(a) Pipe status

Pipe	Hostname	Work Size	Gave Work At	Heard At	Pid
aggregator	frigg	45	Thu May 26 16:18:40 2011	Thu May 26 16:18:01 2011	19155
cache_update	sametsrv02	0	0	Mon May 23 21:46:41 2011	21642
clean	hod	0	0	Thu May 26 16:15:37 2011	28456
clean	hod	3	Thu May 26 16:18:44 2011	Thu May 26 16:18:41 2011	28411
cleanafterclust	odin	0	0	Thu May 26 16:18:31 2011	12963
cleanafterclust	odin	0	0	Thu May 26 16:11:12 2011	13007
cluster	volla	48	Thu May 26 16:17:37 2011	Thu May 26 16:17:37 2011	17041
diseases	odin	0	0	Thu May 26 16:17:02 2011	13146
download	volla	22	Thu May 26 16:18:34 2011	Thu May 26 16:18:29 2011	15680
download	volla	0	0	Thu May 26 16:17:21 2011	15716
events	mimir	45	Thu May 26 16:18:42 2011	Thu May 26 16:18:42 2011	23803
events	mimir	90	Thu May 26 16:17:56 2011	Thu May 26 16:17:56 2011	25622
imageclust	frigg	0	0	Thu May 26 16:17:57 2011	19200
media	mimir	45	Thu May 26 16:18:40 2011	Thu May 26 16:17:27 2011	21591
mediacache	volla	0	0	Thu May 26 16:17:30 2011	16463
misc	frigg	0	0	Thu May 26 14:56:24 2011	20138
ner	mimir	0	0	Thu May 26 16:11:24 2011	7751
ner	mimir	0	0	Thu May 26 16:18:42 2011	7715
people	volla	0	0	Thu May 26 16:17:49 2011	16389
rssgrabber	sametsrv02	0	0	Thu May 26 16:18:43 2011	17949
tag	nott	0	0	Thu May 26 16:18:39 2011	19527
tag	nott	0	0	Thu May 26 16:16:21 2011	19560

(b) Slave status

Figure 3.6: NewsStand's (a) pipe status and (b) module status screen, as reported by the pipe server through its info facility.

**Pipe maintenance** The final pipe server protocol commands are those used for pipe maintenance (“*<clear-pipe-msg>*” and “*<reset-stats-msg>*” in Figure 3.5). These commands allow for clearing a pipe of work, and for resetting the statistics associated with the pipe.

### 3.3.2 Checking Module Status

Every so often, the pipe server checks on the connected module processes to determine their status. For modules that have completed their work batches or are otherwise currently idle, the pipe server allocates a new work batch if any documents are waiting in the module’s pipe. In addition, if a module is taking too long in processing a work batch, the module is considered to have failed, and is disconnected and the work batch assigned to it returned to the pipe. Each module receives a fixed amount of time per document, *doctime*, to complete a work batch. As with *blocksize*, choosing an appropriate *doctime* involves a tradeoff. A small *doctime* ensures that buggy modules are disconnected quickly, but may not allow enough time for legitimate modules to finish their work batches, while a large *doctime* would allow buggy modules to clog their pipes, but also allow legitimate modules to finish their work batches. Currently, *doctime* is set to 30 seconds per document.

## 3.4 Database Design

This section presents NewsStand’s database design. There are two, somewhat disparate main goals behind this design:

1. Managing the large amount of data streaming through NewsStand and derived from its processing.
2. Serving NewsStand’s map query interface quickly to facilitate interactive browsing and exploration of news.

To address the first goal, NewsStand employs a central PostgreSQL [116] relational database that holds all data downloaded and generated from processing. Due to our domain

of streaming news, data is constantly being added to and deleted from this database, unlike typical SQL databases where the data is mostly static. This data churn can wreak havoc on performance and must be managed carefully. For example, database query planners track statistics about data value distributions within these tables to determine efficient query execution plans given a particular query. Heavy data churn means that these statistics will quickly go out of date and must be updated frequently. In PostgreSQL, this operation is known as *vacuuming*, and we perform vacuuming regularly in the database. To address the second goal of serving NewsStand’s map query interface quickly, we maintain a separate *cache database* containing only the most recent data, which improves querying performance (see Section 3.6 for a description of such queries).

Figure 3.7 provides an overview of NewsStand’s database schema, showing the tables used for NewsStand’s core data processing. Tables are color coded by purpose: Red relates to individual documents (*feeds*, *docs*, *doc\_text*, *media*), blue for clusters of documents (*clusters*, *cluster\_locs*), and because locations play such an important role in all of NewsStand’s querying, we give special attention to geotagging tables, shown in green (*entities*, *doc\_locs*). Also, indexed attributes are indicated by filled circles next to the attribute names. Each table’s tuples have one identifier as primary key (shown as filled diamonds next to attribute names), and these identifiers are used in numerous foreign key constraints, shown as arrows, which enforce data integrity. Note that Figure 3.7 is a simplified depiction of NewsStand’s database, which currently contains over 100 tables, but these few serve as a useful illustration of the core data and functionality present in NewsStand, as well as the general design strategy.

In subsequent sections, we describe the tables for documents (Section 3.4.1), geotagging (Section 3.4.2), and clusters (Section 3.4.3), as well as how they are populated by NewsStand’s processing modules (see Section 3.2 for descriptions of these modules). Also, we describe NewsStand’s cache database in Section 3.4.4.

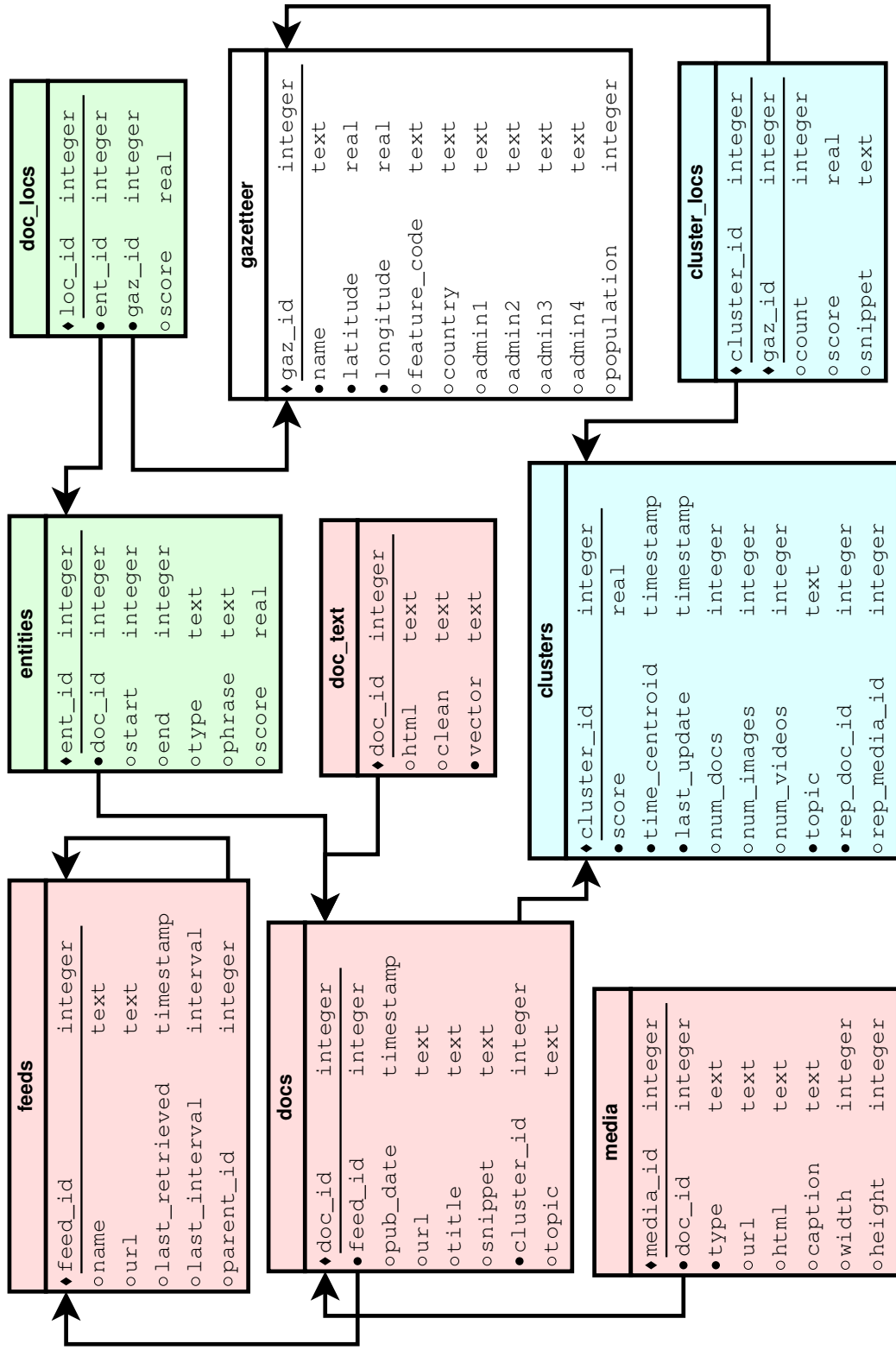


Figure 3.7: NewsStand’s database schema, featuring tables used in core processing and querying. Tables related to individual documents (red), geotagging (green), and clustering (blue) are shown. Arrows indicate foreign key constraints, and indexed attributes are shown by filled shapes adjacent to attribute names (filled diamonds for primary keys, filled circles for other attributes).

### 3.4.1 Documents

Tables related to document processing are shown in red in Figure 3.7. Document processing begins with the `feeds` table, which holds information about the RSS feeds that NewsStand’s *RSS grabber* polls to inject new documents into the system. Along with a feed name and `url`, a timestamp `last_retrieved` and time interval `last_interval` are stored with each feed. The latter two attributes are used by the RSS grabber to adjust the polling intervals of RSS feeds. If the RSS grabber polls a given feed and finds new data, then the polling interval is decreased; otherwise, it is increased. In this way, a feed’s polling interval gradually moves toward its data generation rate. Finally, an additional `parent_id` field stores an optional pointer to a “parent” RSS feed. This field is filled for multiple RSS feeds that belong to the same news source (e.g., a single newspaper with multiple feeds corresponding to different newspaper sections), and allows quick retrieval of all feeds from a single news source.

While polling RSS feeds, the RSS grabber populates the `docs` table, which holds information about individual documents. For each document, the `feed_id` value is set to that of the feed from which it came, and the document’s publication date, source URL, title, and snippet of text, all of which come from the RSS feed, are stored in the `pub_date`, `url`, `title`, and `snippet` attributes, respectively. The `cluster_id` and `topic` fields are populated by NewsStand’s clusterer and topic classifier, to be described shortly.

The next phases of processing for a document involve the `doc_text` table, which stores text versions of the document. After the `docs` entry for a given document  $d$  has been created,  $d$  is downloaded by NewsStand’s *downloader* module, and this HTML version of  $d$  is added to the `html` field for  $d$ . Next, NewsStand’s *cleaner* module takes the downloaded HTML version, and produces a cleaned version, storing the result in the `clean` field. Finally, the document’s clean text is mapped to its vector space representation [126] and stored in the `vector` field, which allows full text indexing and keyword searching. Queries involving keywords that leverage this vector are described in Section 3.6.

After retrieving the document’s text, images, videos, and other media are found by NewsStand’s *media extractor*, and stored in the `media` table. Several relevant attributes are stored along with each media entry, including the media’s `type` (e.g., “image”, “video”), `url`, `embedding html`, `caption` as determined from the embedding structure or other methods (e.g., `image alt tags`), and `width` and `height`.

### 3.4.2 Geotagging

Several tables in NewsStand’s database are populated by the *geotagger* module. The most important of these are the `entities` and `doc_locs` tables, shown in green in Figure 3.7, which contain entities and locations found during toponym recognition and resolution, respectively. `entities` is populated with entities found in the toponym recognition process, such as person names, organizations, and locations. Each entity is associated with the `doc_id` of the document from whence it came, as well as the `start` and `end` offsets within the document, the `type` of entity, the `text phrase` of the entity, and the entity’s `score` which is determined during the entity recognition process. This `score` can reflect, for example, the confidence that the given entity is correct. Of course, toponym recognition is nominally only concerned with location entities; however, in the course of toponym recognition, knowledge that a given term or pattern often signals an entity of some other type can aid in distinguishing locations from non-locations (e.g., knowing that “Mr.” often precedes a person’s name), and these entities are stored as well. Other modules, such as the *people finder* and *disease finder*, also store output entities in the `entities` table.

After finding entities during toponym recognition and populating the `entities` table, the *geotagger* proceeds with toponym resolution to resolve any geo/geo ambiguities and assign final lat/long values to each location entity in `entities` (i.e., the document’s *toponyms*). Lat/long values are obtained from a *gazetteer*, a list of locations and associated metadata, which is stored in the `gazetteer` table. Our gazetteer is based on GeoNames [43], a crowdsourced gazetteer containing over 8 million locations. The output from

the toponym resolution process is stored in the `doc_locs` table, which serves to tie together a toponym entity (`ent_id`) and a gazetteer entry (`gaz_id`). Also, a toponym `score` reflects the importance of the toponym within the document, which is based on frequency and distribution, and is used to determine the document's geographic focus (i.e., central location associated with it). Note that the `doc_locs` table does not store location information explicitly (e.g., lat/long values), which is instead stored in the `gazetteer` table. This design serves to centralize location information so that it can be easily updated, which occurs nightly when the `gazetteer` table is synchronized with GeoNames.

### 3.4.3 Clusters

NewsStand's *clusterer* plays a central role in processing streaming news. Tables involved with clusters of documents are shown in blue in Figure 3.7. The `clusters` table aggregates information about clusters. It contains the cluster's importance `score`, computed as a combination of several factors such as freshness of the documents in the cluster, cluster size, rates of growth such as velocity and acceleration, and diversity of news sources within the cluster. The `score` is used extensively when retrieving clusters to display in NewsStand's user interface (described in Section 3.6). Additional information stored with each cluster includes the time centroid of documents in the cluster (`time_centroid`), last time the cluster was updated (`last_update`), and number of documents, images, and videos in the cluster (`num_docs`, `num_images`, `num_videos`). Also, the cluster's `topic` is computed based on the topics of documents contained within it, and a representative document and image are chosen to display along with the cluster (`rep_doc_id`, `rep_media_id`).

After geotagging the documents in the cluster, the cluster itself is associated with locations found in the documents. These locations are stored in the `cluster_locs` table, which, analogously to `doc_locs`, ties together a cluster (`cluster_id`) and a location (`gaz_id`). Also stored are the number of times the location appears in documents in the

cluster (`count`), how relevant the location is to the cluster as a whole, determined by the scores of instances of that location in the cluster (`score`), and a text `snippet` from the cluster's representative document that contains the location. This snippet is shown in NewsStand's interface, to give users an understanding of the location's relevance to the story.

#### 3.4.4 Cache Database

NewsStand's main database serves as a central data repository, where the main goals are internal consistency, reliability, and enough space to hold the large amount of news flowing through NewsStand. NewsStand downloads approximately 50,000 new documents each day, and stores the most recent four months' of news, including the documents' actual text rather than simply a URL, at any given time. In addition, documents are processed extensively, generating much additional data to be stored in the database. NewsStand's processing modules constantly communicate with the database, resulting in heavy query traffic involving many tuple modifications (i.e., inserts, updates, and deletes). These modifications could cause select statements to block while waiting for transactions to complete. As a result, this database is not suitable for serving NewsStand's user interface, where the key goal is interactive query speed.

Instead, for serving the user interface, we maintain a separate, smaller *cache database* containing only the most recent news data (several days' worth). The cache database has mostly the same schema as the main database, except with less data. Smaller tables result in more table rows residing in the database's cache buffers, rather than having to go to disk to respond to queries. In addition, since NewsStand's user interface does not modify the main database, queries will not block on waiting for modifications. Of course, the cache database needs to be updated from the main database in a timely manner so that the latest news is always being served. A special cache updating module continually polls the main database for clusters whose `last_update` value is newer than the previous update time, and copies the cluster and its associated information to the cache. In this way, updates to



the cache database are minimized and more database query processing resources are used for serving the interface.

### 3.5 Web Interface

In this section, we describe the capabilities and design criteria that went into creating NewsStand's Web interface. Our main goal in designing NewsStand's user interface was to convey as much geographic and non-geographic information about current news as possible. The interface consists of a large map on which stories are placed, and the viewing window serves as a *spatial region query* on the geotagged news stories. Users interact with NewsStand using *pan* and *zoom* capabilities to retrieve additional news stories. As users pan and zoom on the map, the map is constantly updated to retrieve new stories for the viewing window, thus keeping the window filled with stories, regardless of position or zoom level. A given view of the map attempts to produce a summary of the news stories in the view, providing a mixture of story significance and geographic spread of the stories. Users interested in a smaller or larger geographic region than the map shows can zoom in or out to retrieve more stories about that region.

Recall that there are two basic types of spatial queries:

1. *Feature-based*: "Where is story *X* happening?"
2. *Location-based*: "What is happening at location *Y*?"

Corresponding to these query types, there are two basic modes of using NewsStand, termed *top stories mode* and *map mode*. The distinction between top stories mode and map mode can be understood in terms of the map's purpose in answering queries in each mode. In top stories mode, when hovering on a topic in the left pane, the map is populated with the locations associated with that topic. In other words, in this mode, when answering the query "Where is story *X* happening?", the map is used for output. On the other hand, in map mode, the map is used for both input (specifying a query window) and output (showing the

clusters associated with locations in the window). Thus, the map is used both to pose the query “What is happening at location *Y*?” and to display the results.

Figure 3.8a shows NewsStand’s main Web interface in top stories mode. In this mode, a list of news clusters is presented in the text (left) pane, while the locations associated with a news cluster are shown in the map (right) pane. Other information in the cluster list includes the news source for the article, time the cluster was last updated with a new article, and links to the other articles, images, and videos in the cluster. In addition, a number of controls in NewsStand’s top pane allow for controlling various other aspects of querying. A set of topic links at top left allow for filtering of the displayed clusters by general topic (e.g., business, entertainment, sports), and a search box at top right allows for filtering clusters by keyword. Also, a drop down list allows one of various mapping APIs to be selected, and an “Options” link lets users filter stories by newspaper source. Notice that the top stories mode bears similarity to STEWARD’s user interface (described in Section 2.4), though in STEWARD, each entry in the left list is a single document, while in NewsStand each entry corresponds to a cluster of news articles. NewsStand also offers a much richer set of querying capabilities due to the greater variety of processing performed by NewsStand’s processing modules.

Hovering on a cluster in the text pane, as is done in Figure 3.8a for a news cluster about the US Republican Party’s 2012 primary elections, results in the locations associated with that cluster being displayed as location markers on the map. In other words, the feature-based query “Where is story *X* happening?” is executed, where *X* is the cluster that the mouse hovers on. This action also opens an info window for the most prominent location in the cluster, which shows a mention of the location within the context of a news article in the cluster. This allows users to understand not simply that a location was associated with the cluster (e.g., in Figure 3.8a, “South Carolina”), but also *why* it was associated with the cluster by displaying the context. Also, links allow users to navigate through different instances of the location that are present in the article, to show other contexts in which



(a) Main user interface



(b) Images in a news cluster

Figure 3.8: NewsStand's Web interface in top stories mode, showing (a) the main user interface, and (b) browsing images for a single news cluster.

the location appeared. Hovering on a different map marker opens an info window for the corresponding location. Figure 3.8b shows the set of images extracted from articles in the elections cluster, which were found using the media extractor (described in Section 3.2.8). Each image is associated with a caption which can be accessed by hovering the mouse over the image.

Figure 3.9a shows NewsStand when browsing in map mode. In this mode, the map expands to the entire screen and is used for executing location-based queries—that is, “What is happening at location *Y*?”, where *Y* is the geographic region displayed in the map. Results of one such query covering the United States is shown in Figure 3.9a. Marker icons corresponding to news clusters are displayed at the principal locations associated with each cluster. When several clusters are associated with a single location, a small “+” is displayed above the marker icon at that location. Also, several different marker icons are used, which indicate the general topic of news stories (e.g., business, entertainment, sports). Hovering on a marker icon opens an info window showing the information associated with the cluster tagged to that location. Several additional controls allow other querying and navigation capabilities. At top right, “Local” and “World” links and a “Locate” box allow users to automatically pan and zoom the map to their present location, a world view, or a user-specified location, respectively. Also, a slider allows the number of icons displayed on the map to be changed dynamically to suit users’ needs.

While in map mode, different layers of data can be displayed on the map, with each layer having been generated by a different type of processing in NewsStand. While Figure 3.9a shows the map mode’s icon layer, where cluster locations are represented by story icons, Figure 3.9b shows the map mode’s location layer, where locations are represented textually. The location layer allows faster discovery of specific locations mentioned in the news, rather than the general areas of interest shown by the icon layer. However, because the keywords take up more screen space than the markers, it is difficult to place many stories on the map without introducing clutter.



(a) Icon layer



(b) Location layer

Figure 3.9: NewsStand's Web interface in map mode, showing (a) its icon layer, where cluster locations are represented by icons, and (b) its location layer, where cluster locations are represented textually.

Interestingly, rather than only being useful for end users, the textual representation also allows for rapid “eyeballing” of the geotagger’s quality of output by leveraging humans’ advanced visual understanding. For example, in Figure 3.9b, one article mentioned “Chad”, which was tagged to the African country. However, as the snippet shows, “Chad” actually is used as a person’s name in the article, so this is a geotagging error—in particular, an error in toponym recognition, which involves resolving geo/non-geo ambiguity (see Chapter 4). To aid in the development of our geotagging algorithms, NewsStand allows human users to provide feedback via the error feedback menu in each info window. Users can specify whether the tagged location is in reality “not a location” (geo/non-geo error) or “wrong location” (geo/geo error), and can enter a textual comment as well. Our toponym recognition and toponym resolution algorithms (described in Chapters 4–7) use this feedback to retrain and hopefully improve its accuracy in the future.

Figure 3.10 shows NewsStand’s people and disease layers, which are generated from the output of NewsStand’s people finder and disease finder modules (described in Section 3.2.7). As with the location layer, people and diseases are shown textually in these layers, and in some cases, errors can be found. For example, nicknames and titles are sometimes mistakenly tagged as person names, such as “America”, corresponding to “Mr. America”, in Figure 3.10b. As before, the slider dynamically changes the number of entities displayed on the map.

### 3.6 Database Queries

Here, we describe some of the database queries used in NewsStand’s Web interface, described in Section 3.5. With NewsStand’s database schema (described in Section 3.4), each interface action is easy to cast in terms of an SQL query. Recall that the two main modes of using NewsStand correspond to the two main types of SQL queries that it supports: *top stories mode*, used to answer feature-based queries (“Where is story *X* happening?”), and *map mode*, which answers location-based queries (“What is happening at location *Y*?”).



(a) Disease layer



(b) People layer

Figure 3.10: NewsStand's (a) disease and (b) people layers.

Note that NewsStand’s database contains far more information than is feasible to send over a network connection and display in a client’s user interface. Instead, we are only interested in the top few results for each query. To this end, as we will see in the following sections, the queries used within these two modes are all variants of what is known in database parlance as *top- $k$  queries*, i.e., queries that return the first  $k$  ranked results according to some ranking function, which varies depending on the particular query. In addition, many more queries are used throughout NewsStand’s interface than are presented here, due to lack of space. Instead, we present only the queries associated with story and location retrieval, though these queries will serve to illustrate the principle of top- $k$  retrieval that applies to all of NewsStand’s interface queries.

Additionally, the queries introduced below feature *joins* among multiple tables in the database, which often hinder query performance. However, these joins are presented for conceptual understanding rather than reflecting the actual implementation of these queries. As noted in Section 3.4.4, in order to improve the speed of querying and hence improve interactivity, the queries that serve the user interface are executed in the cache database rather than in NewsStand’s main database. When executing the queries below, we use what are called *materialized views* of the query results, which amount to precomputation and storage of the query results (without any filtering conditions) prior to runtime. For example, consider the query “SELECT \* FROM a, b WHERE a.id = b.id AND a.val1 = 'X' AND b.val2 = 'Y'”. Join queries in NewsStand have a similar form, though they are much longer, and so for the sake of brevity we omit the full text of these queries. Creating a materialized view of this query would involve executing the join portion of the query, namely “SELECT \* FROM a, b WHERE a.id = b.id”, and storing the result. Henceforth, when executing the original query, the filter conditions (“a.val1 = 'X' AND b.val2 = 'Y'”) would only need to be applied to the materialized view result, without any joins at runtime. This additional optimization further improves cache query performance.



Below, we continue with descriptions of NewsStand’s interfaces and queries that return clusters in top stories mode (Section 3.6.1) and map mode (Section 3.6.2), as well as queries involving a single cluster that are used in both modes (Section 3.6.3).

### 3.6.1 Top Stories Mode

Answering queries of the form “Where is story  $X$  happening?” is the purview of NewsStand’s first mode, referred to as “top stories mode” or “text mode”. This query is also known as a feature-based query. Figure 3.8a shows NewsStand’s interface in top stories mode. In this case, assuming a landscape display, the left pane shows the top- $k$  story clusters, ranked in importance from top to bottom of the visible part of the display screen. This pane is populated by one of several queries presented in Figure 3.11, depending on filtering parameters. Figure 3.11a is the basic query to populate the pane, which retrieves clusters in order of cluster score. Additionally, several links at the top left allow for filtering of the top- $k$  clusters according to their topic types (e.g., “Business”, “Sports”), while at top right, a menu allows selecting clusters with articles from particular news feeds, and a keyword search allows the selection of clusters relevant to particular keywords. These operations are implemented using the SQL presented in Figures 3.11b, 3.11c, and 3.11d, respectively, which are all variants of the basic top- $k$  query of Figure 3.11a except with additional constraints. For the keyword search query of Figure 3.11d, the cluster rankings are modified to include a keyword relevance measure in addition to the cluster’s score.

As the mouse is hovered over the clusters in the left pane, the most relevant locations in the selected cluster are displayed on the map in the right pane of the display using what we term “markers”, which are icons corresponding to the most dominant topic type of the elements of the cluster. This action corresponds to the feature-based query of “Where is story  $X$  happening?”, in that its input is one of the news clusters, and its output is the set of locations that are relevant to the cluster. Also, note that a slider is present at the top of the right pane, whose movement to the right (left) allows the maximum number of locations

```
SELECT c.cluster_id FROM clusters c
ORDER BY c.score LIMIT k
```

(a) Get top clusters

```
SELECT c.cluster_id FROM clusters c
WHERE c.topic = t
ORDER BY c.score LIMIT k
```

(b) Get top clusters of a given topic *t*

```
SELECT DISTINCT c.cluster_id
FROM clusters c, docs d
WHERE c.cluster_id = d.cluster_id
      AND d.feed_id = fid
ORDER BY c.score LIMIT k
```

(c) Get top clusters with articles from feed *fid*

```
SELECT DISTINCT c.cluster_id
FROM clusters c, docs d, doc_text dt
WHERE c.cluster_id = d.cluster_id
      AND d.doc_id = dt.doc_id
      AND match(dt.vector, kw)
ORDER BY kwrnk(c.score, dt.vector, kw) LIMIT k
```

(d) Get top clusters with keyword *kw*

Figure 3.11: SQL queries used to populate the left pane of top stories mode.

for which icons are displayed for the highlighted cluster to be increased (decreased). The identity of the locations for which icons are present depends on the number of times the location is mentioned in the articles that make up the cluster, with priority given to those that are mentioned most frequently. The presence of this slider is precisely the novel aspect of NewsStand that enables it to answer the top- $k$  version of the feature-based query of “Where is story  $X$  happening?” The algorithm that performs the display ensures that all of the desired locations can be seen, and thus the area displayed is the minimum bounding box of the locations. In database parlance, what we have here is an instance of a top- $k$  query where  $k$  corresponds to the number of visible locations for a particular cluster or a cluster that contains a particular keyword. That is, we have a “top- $k$  locations” query. This query is implemented in SQL as shown in Figure 3.13a, to be described shortly.

### 3.6.2 Map Mode

NewsStand’s second mode, used for answering queries of the form “What is happening at location  $Y$ ?”, is termed *map mode* and is shown in Figure 3.9a. This query is also known as a location-based query. In this case, NewsStand provides the capability of reading over 8,000 newspapers (RSS feeds) by using a map. As mentioned earlier, the result of processing the RSS feeds is a set of clusters of articles by topic. Initially, the map contains topic icons at locations corresponding to those in the  $k$  most representative clusters, where “representative” takes into account factors such as importance measured by currency, size and rate of growth of the cluster in terms of velocity and acceleration, as well as a desire to have a good spatial distribution in the area being displayed. A variant of the feature-based query can also be executed in map mode. This is done by entering a keyword in the “search” box at upper right. The result is a set of  $k$  clusters relevant to the keyword, and displayed using topic icons at the clusters’ locations. Once a search has been activated, all later searches will be restricted to the keyword. However, the searches are restricted to the displayed part of the map—that is, they are spatially restricted and are analogous to a spatial join operation.

Again, while in map mode, a slider is present at the upper right corner of the map whose movement to the right (left) effectively allows the maximum number of different clusters for which topic icons are displayed at their representative locations to increase (decrease). Although this feature seems very similar to its analog in top stories mode, its semantics are actually very different. In particular, each additional location corresponds to potentially increasing the number of viewable clusters although this is achieved by increasing the number of locations for which icons are displayed. The presence of this slider represents a second novel aspect of NewsStand as it enables it to answer the location-based query of the form “What is happening at location  $Y$ ?” where, contrary to conventional assumptions,  $Y$  is not a location, but is actually a region corresponding to the part of the world that is viewable. Thus, moving the slider to the right increases the number of clusters that could be viewed, as these clusters are associated with the various locations, although the clusters associated with the additional location could be the same as the clusters associated with the existing viewable locations. Thus, we see that the number of viewable clusters resulting from moving the slider to the right is non-decreasing. In database parlance, what we have here is an instance of a top- $k$  query where  $k$  corresponds to the number of viewable clusters. In other words, we have a “top- $k$  clusters” query. Again, we could say that the resolution (also referred to as the zoom level, but measured here in terms of the number of clusters that are visible, again in contrast to the conventional definition which is in terms of visible area) also increases.

Another database analog of the top- $k$  clusters query is a ranked spatial range query or a ranked spatial join query. The challenge in implementing this query lies in deciding on the order in which the locations corresponding to clusters are delivered to the user, which is a function of their importance. This need not necessarily be the number of times they are mentioned. For example, it could be based on the number of clusters in which they appear at least once. Another factor could be their currency in terms of the time at which they arrive, and the velocity and acceleration of the cluster’s rate of growth. Again, these issues

arise because our data is dynamic on account of our streaming environment. Given the above analogies of top stories mode and map mode with the top- $k$  query, it is also natural to let  $k$  vary, where  $k$  denotes the number of markers (topic type icons), which is achieved by using the slider.

Figure 3.12 presents examples of top- $k$  cluster queries used in map mode. Figure 3.12a illustrates the default mode, where a set of clusters are used to populate the visible map window. The `clusters` table is joined with `cluster_locs` to retrieve the cluster locations, as well as with the `gazetteer` table to retrieve lat/long information, which in turn is used to filter the clusters to only those that lie within the query window  $qw$ . Results are ordered by the cluster’s score. Notice that this query is related to that of Figure 3.11a, except with an additional query window constraint on the cluster’s locations. Also, as before, we use variants of this query to incorporate additional query constraints involving topics, source feeds, and keywords, which are shown in Figures 3.12b, 3.12c, and 3.12d.

### 3.6.3 Single-Cluster Queries

In this section, we present queries that retrieve information about a single cluster. These queries are used in both top stories mode and map mode, since both modes involve the retrieval and display of news clusters and information associated with them. As with previous Web interface operations, these actions are easy to cast in terms of SQL using NewsStand’s database schema. These queries are presented in Figure 3.13, and are described in detail below.

Recall from Section 3.6.1 that the primary form of queries in top stories mode is “Where is story  $X$  happening?”, where  $X$  corresponds to a cluster of news. This query corresponds to Figure 3.13a, which retrieves the top- $k$  locations associated with a given cluster with identifier  $cid$ . The locations are retrieved in order of their relevance score to cluster  $cid$ .

In addition, several queries are used to render cluster information in NewsStand’s user interface. In both top stories mode and map mode, summary information about each cluster

```

SELECT DISTINCT c.cluster_id
FROM clusters c, cluster_locs cl, gazetteer g
WHERE c.cluster_id = cl.cluster_id
      AND cl.gaz_id = g.gaz_id
      AND contains(qw, g.latitude, g.longitude)
ORDER BY c.score LIMIT k

```

(a) Get top clusters in query window *qw*

```

SELECT DISTINCT c.cluster_id
FROM clusters c, cluster_locs cl, gazetteer g
WHERE c.cluster_id = cl.cluster_id
      AND cl.gaz_id = g.gaz_id
      AND contains(qw, g.latitude, g.longitude)
      AND c.topic = t
ORDER BY c.score LIMIT k

```

(b) Get top clusters in query window *qw* with topic *t*

```

SELECT DISTINCT c.cluster_id
FROM clusters c, cluster_locs cl, gazetteer g
WHERE c.cluster_id = cl.cluster_id
      AND cl.gaz_id = g.gaz_id
      AND c.cluster_id IN (
        SELECT d.cluster_id FROM docs d
        WHERE d.feed_id = fid
      )
      AND contains(qw, g.latitude, g.longitude)
ORDER BY c.score LIMIT k

```

(c) Get top clusters in query window *qw* with articles from feed *fid*

```

SELECT DISTINCT c.cluster_id
FROM clusters c, cluster_locs cl,
      doc_text dt, gazetteer g
WHERE c.cluster_id = cl.cluster_id
      AND cl.gaz_id = g.gaz_id
      AND c.rep_doc_id = dt.doc_id
      AND contains(qw, g.latitude, g.longitude)
      AND match(dt.vector, kw)
ORDER BY kwrnk(c.score, dt.vector, kw) LIMIT k

```

(d) Get top clusters in query window *qw* with keyword *kw*

Figure 3.12: SQL queries used in map mode.

```

SELECT * FROM clusters c, cluster_locs cl, gazetteer g
WHERE c.cluster_id = cl.cluster_id
      AND cl.gaz_id = g.gaz_id
      AND c.cluster_id = cid
ORDER BY cl.score LIMIT k

```

(a) Get top locations for cluster *cid*

```

SELECT * FROM clusters c, docs d,
         feeds f, media m
WHERE c.rep_doc_id = d.doc_id
      AND d.feed_id = f.feed_id
      AND c.rep_media_id = m.media_id
      AND c.cluster_id = cid

```

(b) Get cluster summary for cluster *cid*

```

SELECT * FROM docs d
WHERE d.cluster_id = cid
ORDER BY d.pub_date LIMIT k

```

(c) Get all articles in cluster *cid*

```

SELECT * FROM docs d, media m
WHERE d.media_id = m.media_id
      AND d.cluster_id = cid
      AND m.type = t
ORDER BY d.pub_date LIMIT k

```

(d) Get media of type *t* for cluster *cid*

Figure 3.13: Queries to retrieve single-cluster information.

is displayed, in different ways. In top stories mode, each cluster is shown with the title of its representative article, along with a snippet of text from the article, when it was last updated, total number of documents in the cluster, and so on. This information is retrieved with the query presented in Figure 3.13b. Additional links in the left pane allow users to retrieve all documents, images, or videos associated with a single cluster. Documents are retrieved using the query in Figure 3.13c which takes data from the `docs` table, while the retrieval of images and videos is accomplished with the query of Figure 3.13d, which retrieves media of the appropriate type from the `media` table.

## 3.7 Experiments

This section describes the results of several experiments designed to characterize the performance of various parts of the NewsStand system. We investigate the amount of streaming news collected over time (Section 3.7.1), the numbers of geotagged toponyms found by NewsStand’s geotagger and images found by its media extractor (Section 3.7.2), and the size of components in NewsStand’s database (Section 3.7.3). We also investigate the time required for processing articles (Section 3.7.4) as well as the execution speed of database queries to retrieve information for NewsStand’s Web interface (Section 3.7.5). Note that unlike typical system evaluations on news data, which use small, static corpora of news from a single generally prominent source (e.g., Reuters, New York Times), these experiments were conducted on the live NewsStand database system over several months’ worth of streaming news data. As a result, they better characterize NewsStand’s long term performance on streaming news.

### 3.7.1 Data Collection

First, we measured how frequently articles are collected and entered into the NewsStand database through the many RSS feeds that it polls. Figures 3.14a and 3.14b present statistics



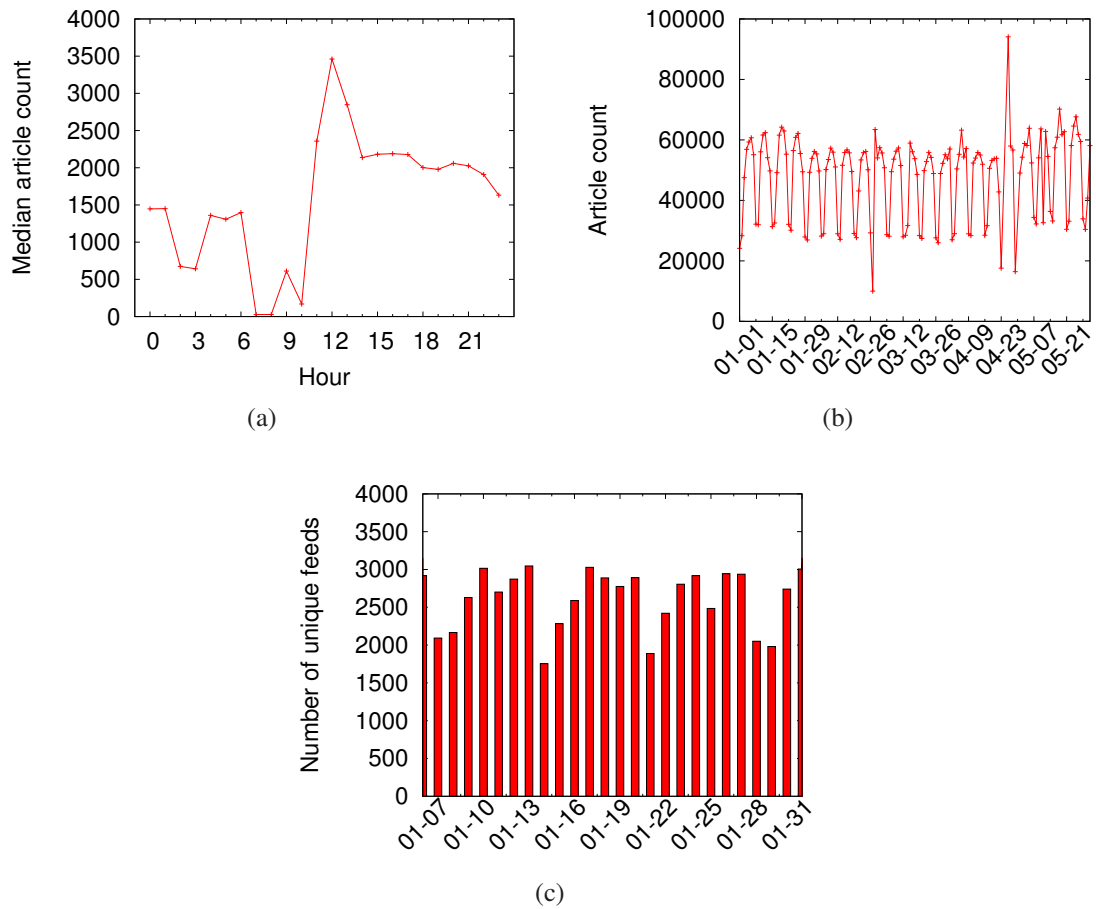


Figure 3.14: Number of articles retrieved by NewsStand (a) per hour and (b) per day, as well as (c) the number of unique feeds supplying these articles.

about the number of articles retrieved by NewsStand from RSS feeds per hour (measured using Eastern Standard Time) and per day, respectively, as measured over a five month period from January to May, 2011. Both figures show the large volume of news processed by the NewsStand database every day, which dwarfs typical corpus sizes (i.e., hundreds of documents) used in information retrieval and geotagging research (e.g., [7, 42, 68, 97]). They also show the generally cyclical publishing rate of sources polled by NewsStand, with most articles being published during daytime hours and during the week, rather than on weekends. The clustering of article publishing times between the hours 11–13 in Figure 3.14a is also a byproduct of NewsStand’s current focus on US-based news sources. Figure 3.14b shows that the NewsStand database ingests on the order of 50,000–60,000 articles on weekdays, and about 30,000 articles on weekends. Also, Figure 3.14b demonstrates dips in article counts followed by peaks near 26 Feb and 25 Apr, which are due to system downtime at those times, and subsequent catching up of the system. We also measured the number of unique feeds that supply these articles over time, and these feed counts are shown in Figure 3.14c. Over all measured days, NewsStand processed and displayed articles of around 2,000–3,000 distinct news sources, indicating the breadth and variety of news sources and data available in NewsStand.

### 3.7.2 Data Content

While a primary goal of NewsStand is to deliver as much news as possible to its users, the actual content of such news is important as well. Here, we present measurements of the extracted content found by NewsStand in the articles that it retrieves from news sources. For this experiment, we measure content in terms of the number and types of locations found by NewsStand’s geotagger, as well as the amount of multimedia items found by NewsStand’s media extractor. These measurements were made over a month’s time, and are shown in Figures 3.15a and 3.15b. The number of locations and media follow the characteristic week-versus-weekend pattern seen earlier, with around 130,000 extracted locations

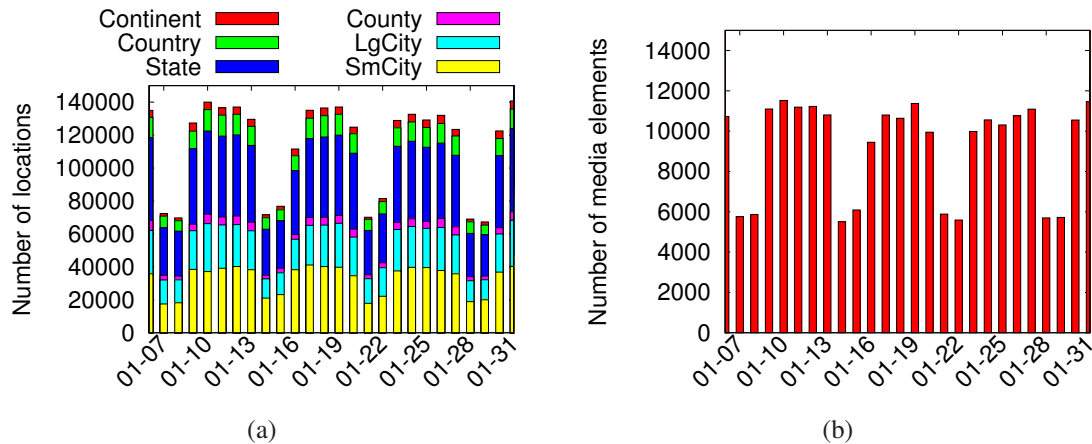


Figure 3.15: Different types of content found by NewsStand in articles, including (a) number and types of locations, and (b) amount of multimedia items (images and videos).

and 11,000 extracted media during the week. With about 40,000 articles downloaded daily, this corresponds to 3–4 locations per article on average, which indicates the usefulness of NewsStand’s map-based interface, since it will be filled with a large number locations and thus there is much data to query. Further, examining the breakdown of location types, smaller places including states and small cities (under 100,000 population) dominate the location type counts, followed by larger places including large cities (over 100,000 population) and countries. These type counts show NewsStand’s focus on highly local streaming news which is more difficult to geotag automatically, but results in a much richer, local news experience. On the other hand, the relatively low number of images and videos, about 1 image per 3–4 articles, shows the difficulty of extracting relevant images and image captions while also filtering for advertising and other spurious media.

### 3.7.3 Data Size

Next, we examined the sizes of tables and indexes present in NewsStand’s database. The sizes of the tables shown in Figure 3.7 are listed in Table 3.2. For each table, we include the number of rows, the total disk space consumed by the data in the rows, and the total disk space of data plus the indexes present in the table (“D + I”). We see that the `doc_text`

Table 3.2: Database object sizes.

	Rows	Data	D + I
<code>cluster_locs</code>	4.0M	8GB	15GB
<code>clusters</code>	2.8M	1.9GB	4.5GB
<code>doc_locs</code>	25.8M	5GB	12GB
<code>docs</code>	6.3M	20GB	81GB
<code>doc_text</code>	5.5M	83GB	87GB
<code>entities</code>	207M	25GB	38GB
<code>feeds</code>	10k	200MB	350MB
<code>gazetteer</code>	8.0M	1.5GB	4.2GB
<code>media</code>	1.4M	4GB	9GB

table is by far the largest in terms of raw data amount, 83GB, which is not surprising given that it contains the full text of articles in the database. However, the `docs` table rivals `doc_text` in size when accounting for the index space as well, since the `docs` table has many more columns and indexes on these columns, including a full-text index on the `snippet` attribute for keyword searches. In terms of number of rows, the `doc_locs` and `entities` tables (containing data for locations and entities in documents, respectively) have many rows, reflecting the many locations and other entities found by NewsStand’s geotagger and other processing modules.

### 3.7.4 Processing Time

Next, we examined the times required for a new article ingested into the NewsStand database to be completely processed by all its modules, as well as the time for the article to become available in the Web interface. For the backend processing time, we summed the time taken by each module to process the document. On the other hand, the times to become available in the Web interface include the total module processing time, as well as the time necessary for pipe server communication and cache database updates. As a result, we studied both times to provide both backend and frontend performance evaluations. Figure 3.16a presents the total module processing times as measured over one month’s worth of news.

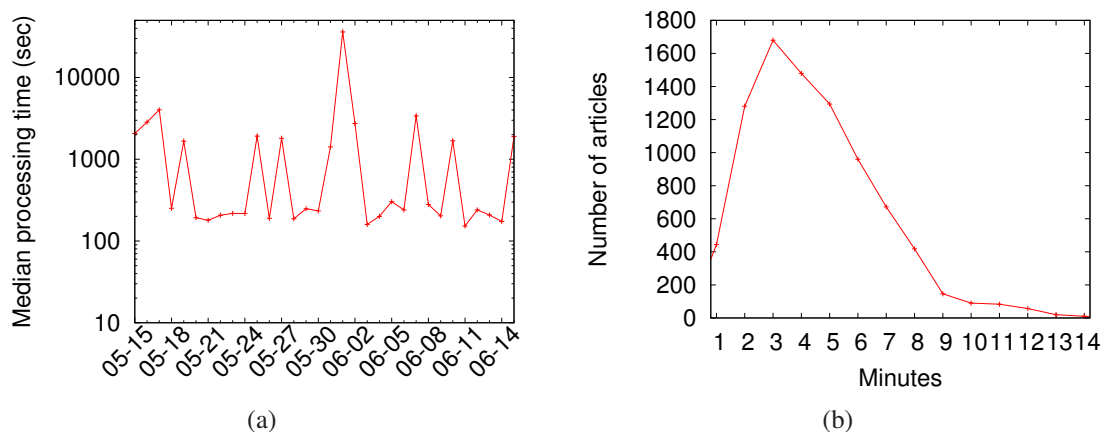


Figure 3.16: Backend processing time in terms of (a) median processing time per document, measured over a month's worth of news, and (b) time for articles to appear in NewsStand's Web interface, measured over a day's worth of news.

On most days, articles were fully processed within a few minutes, demonstrating that NewsStand's database architecture is well adapted for processing streaming news. However, several spikes in the processing time appear as well, with a large spike on 1 Jun. These spikes are due to downtime and maintenance for the machines executing NewsStand's database modules, which due to being a research system are somewhat unavoidable.

We also directly measured the amount of time it took for articles to appear in NewsStand's Web interface. Over a day's time, we executed a query through the Web interface every 5 minutes to retrieve the 30 most recent articles, and measured how long it took for those articles to be processed. We took these measurements during a day where NewsStand was processing documents under typical daily loads—i.e., not during processing spikes due to downtime or other factors. Figure 3.16b shows the results, with most retrieved articles taking between 3–5 minutes to appear on the Web interface. This result corroborates our previous finding that NewsStand processes streaming news quickly.

We further characterized article processing times in terms of how much time is spent in each of NewsStand's processing modules. These times are listed in Table 3.3. Processing times per document are rather low, with the total processing time for a single document on average being just over 6 seconds. As work is batched in groups of at most 100, a given

Table 3.3: Time required by each module to process single documents, in seconds.

Downloader	1.024	Geotagger	2.961
Cleaner	0.098	People finder	0.535
Clusterer	1.648	Disease finder	0.125
Topic classifier	0.047	Media extractor	0.166
Total		6.604	

work batch would travel through the system and be fully processed by NewsStand’s modules in at most 10 minutes, at full load. Of course, if the system is not saturated with work, then articles can be fully processed in a shorter time, and as demonstrated by Figure 3.16b, we typically observe articles in NewsStand that are only a few minutes old. Examining individual processing times in detail, the geotagger, clusterer, and downloader modules are bottlenecks in processing time, accounting for fully 85% of the total processing time for a document. Reasons for the slowness of these modules vary, but can generally be accounted for by the number of database queries that they make. The geotagger and clusterer modules execute a large number of database queries to draw in additional evidence for use in geotagging and clustering. On the other hand, the downloader module’s primary bottleneck is the time required to download articles from websites. Note that despite their slow processing times, we avoid slowing the system as a whole by starting more instances of these modules, thus increasing the system’s processing throughput. We also start more instances when a large amount of work is waiting in pipes, due to modules having been stopped for some time.

### 3.7.5 Query Performance

Our final set of experiments were designed to test the interactivity of NewsStand’s Web interface, as measured by the time required to execute database queries generated by the interface, especially in NewsStand’s map mode. As outlined in Section 3.6.2, these queries are all variants of top- $k$  window queries. Recall that Figure 3.12 shows examples of this

type of query, which all contain a query window  $qw$  expressed in lat/long values, and return the  $k$  clusters with highest scores that fall within  $qw$ . We generated queries by randomly creating query windows over land masses with a variety of sizes. Each window consists of lower left lat/long values, width, and height. We executed these queries in NewsStand's cache database (described in Section 3.4.4), and measured query performance over time. In this way, we tracked live performance fluctuations that arise from the NewsStand database operating on streaming news.

Our first set of query performance experiments tested the number of top- $k$  window queries per second that could be sustained by NewsStand's cache database, which should be roughly proportional to the number of users that could be simultaneously browsing news in NewsStand's interface. For each experiment, we created several test processes, each of which would connect to the database and execute queries as quickly as possible. By increasing the number of processes and hence database connections, we saturated the database's query capacity to determine an upper limit on the number of queries per second that can be handled simultaneously. For this experiment,  $k$  was fixed at 200, matching the value used in NewsStand's user interface. Though this value may seem small, especially compared to the database's size, it is actually reasonable given limited bandwidth and screen space.

Figure 3.17a presents the queries per second (qps) rate delivered by NewsStand's cache database, as measured over one minute of processing. As the number of test processes increased, the qps rate likewise increased, to a ceiling of about 100, as shown by the flattening of the qps curve at around 8–9 test processes. If we consider that a user of NewsStand could generate at most two queries per second, via continuous actions such as scrolling, panning, and zooming, the system as-is can support up to 50 users at a time. These performance results are respectable, especially given that they were executed on a live system, constantly being updated with additional streaming news. To gain additional performance when scaling the system up to support larger numbers of users, further options such as database replication and round-robin scheduling could be considered.

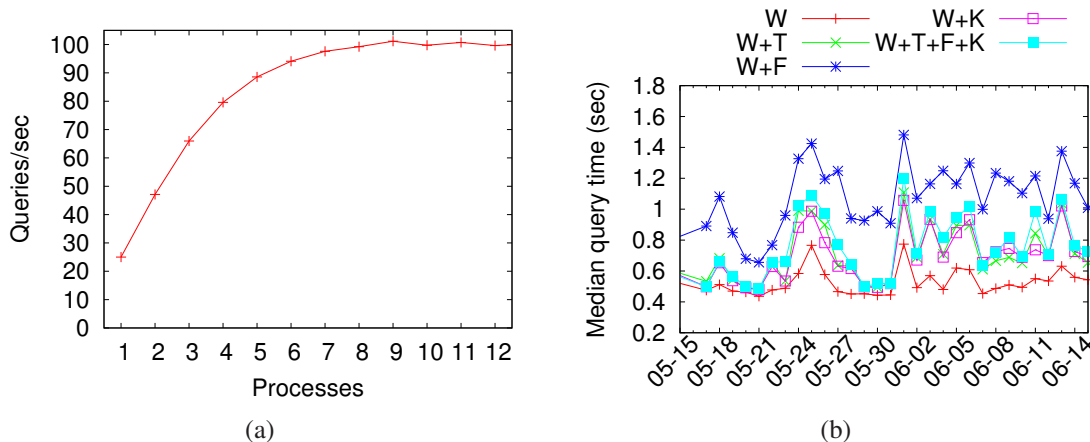


Figure 3.17: Query performance as measured by (a) the sustained number of queries executed by NewsStand’s database, and (b) median query times for top- $k$  window queries with various additional constraints.

Our next query time experiment combined the top- $k$  window queries described above with additional constraints that are added by certain queries available in NewsStand’s interface. We tested the following additional constraints:

1. *Topic*: Retrieve stories relevant to a given topic (e.g., “Business”, “Sports”).
2. *Feeds*: Retrieve stories with articles from a given set of news feeds.
3. *Keyword*: Retrieve stories relevant to a keyword.

To select topics, feeds, and keywords to use in these queries, we randomly sampled query values present in NewsStand’s query log files. Furthermore, in keeping with our goal of benchmarking streaming news, we executed a set of these queries every five minutes over a one month period, to track performance of the live NewsStand system over time. Figure 3.17b contains performance results for these queries, shown as the median query time per day for each query type, with constraints specified as letters: “W” for window, “T” for topic, “F” for feeds, and “K” for keywords, and combinations such as “W+F” referring to a window query with a feed constraint. Examining the results, we see that query times fluctuate from day to day, but tend to fall under 1 sec. The plain top- $k$  window query was faster



than the other query types, with additional constraints generally slowing the query time. Further, the feeds query was consistently slower than the other query types, most likely due to the requirement that query results be from a set of feeds, rather than a single value. Nonetheless, query performance was relatively consistent across all dates and times tested, reflecting NewsStand’s stability as a live system.

For our final query time experiments, we examined the execution time for queries placed in a particular region or zoom level. Figure 3.18a presents query times for random window sizes and positions categorized by the location of the query region, as a box and whisker plot. For all regions, query times were again respectable, with all medians being under 0.2 sec. Furthermore, notice that the query times tended to have skewed distributions, with most query times being low, but with high outliers. We also observed that query performance for query windows in the US region is significantly higher than in other areas. This is likely due to NewsStand’s source bias for newspapers in the US rather than other areas. Figure 3.18b explores query performance differences by varying the zoom level where again both query window size and positioning were randomized and no distinction was made for different regions of the world. As might be expected, as the zoom level increases (thus decreasing the query window size and hence the number of markers present in the window), query times decreased. As before, query times are generally low, with some larger outliers.

### 3.8 Open Problems

Several aspects of NewsStand could benefit from further improvement. NewsStand tends to exhibit a geographic bias toward the areas about which news stories are usually written, so a more uniform coverage of the news is needed. Also, the system currently only processes articles written in English, so it could be improved by adding articles and news sources in other languages. Some of NewsStand’s modules could be further improved, such as its cleaner module which would incorporate some kind of advertising detection and removal,

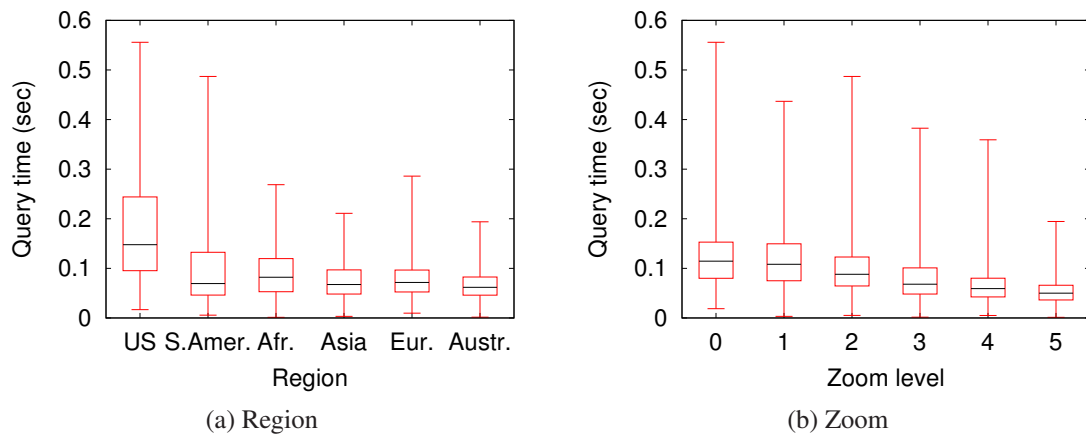


Figure 3.18: Region and zoom level versus query time.

as well as a more generic media extractor which searches for specific plugin controls (e.g., videos embedded in Flash) and retrieves information from them. Video captions could be recognized using cluster terms. Additionally, as mentioned previously, the RSS grabber adds documents to NewsStand’s database only once, as they are published, rather than re-processing articles which were previously published but since updated. Provisions could be made to update and reprocess these articles by, for example, observing a different publication date, even for articles with the same URL. Improvement of these modules would improve in turn the input, and hence performance, of our geotagging methods (described in Chapters 4–7). We also plan to place other media on the map itself, such as representative pictures, videos, and audio clips. We must therefore examine methods for determining the best representative picture for a cluster of news articles. We could also develop methods for finding and searching for quotes in news articles, and associate them with their speakers.

In addition to module improvements, NewsStand’s architecture could be improved to increase its reliability and stability. NewsStand’s modules currently run on a collection of workstations and are started and restarted manually with the aid of administrative scripts. However, it could be made more robust by designing a system to automatically restart crashed modules. In addition, NewsStand’s architecture makes it suitable for processing documents using *cloud computing*, since multiple instances of processing modules can

execute simultaneously. As a result, a high reliability system, perhaps involving MapReduce [34] could decentralize processing and ensure work completion in a timely manner. Also, there are several problems using NewsStand's PostgreSQL database, in that the database, and especially the cache database, involve a large amount of data churn. This SQL-style database, along with others such as MySQL, were not designed to handle large volumes of streaming data, such as streaming news, and thus require constant maintenance in the form of vacuum cycles, which in turn use up the database server's resources and impact query performance. The problem is exacerbated for domains such as Twitter which involves a much larger data stream. To accommodate these data streams, we may be able to leverage the emerging trend of *NoSQL* databases [67], such as MongoDB [1], CouchDB [10], HBase [11], or one of many others, to reduce such maintenance penalties on query performance. Also, rather than having a single cache database, several such databases on separate computers could allow greater scalability and a larger eventual user base. With a larger user base, additional analytics can be used to improve querying performance. For example, the geographic location of users (as determined by IP addresses), or their most frequent window queries, can be used to inform caching strategies.

Another interesting aspect of the news is its temporal nature. We could explore use NewsStand's archive of news through time to examine which locations and other entities were in the news in given windows of time. Also, note that news stories evolve over time, and newspapers update news articles as new details of the story emerge. Ideally, NewsStand would be updated with the latest version of articles as they are updated. Handling news article updates by downloading later versions of pages and incorporating them into the system. The basic problem is again the "pull" nature of webpages in that we don't want to poll too often because it will waste bandwidth. For example, we might check for duplicate URLs in RSS feeds, and check whether the title, description, or publication date has been updated since we last saw it, which can indicate whether new data is present in the article.

### 3.9 Summary

NewsStand, along with STEWARD, described in Chapter 2, provides an important motivation for the development of our geotagging algorithms presented in Chapters 4–7. NewsStand demonstrates that extracting geographic content from news articles exposes a previously unseen dimension of information that aid in understanding the news. Indeed, “NEWS” can be succinctly described as an acronym of “North, East, West, South”. We have also shown that NewsStand’s architecture and database design support rapid processing and querying of streaming news. We believe that the increasing prevalence of geotagged content on the Web will enable compelling applications for systems like NewsStand in other knowledge domains. Also, it is clear that as the prevalence of streaming data on the Web increases, systems such as NewsStand that are capable of quickly processing this streaming data will have ever increasing importance.

## Chapter 4

### Multifaceted Toponym Recognition

Now that we have introduced frameworks for using geotagging in the form of the STEWARD (Chapter 2) and NewsStand (Chapter 3) systems, we proceed with performing the geotagging itself. As noted earlier, the first step toward successful geotagging is finding toponyms within the document to be geotagged, referred to as *toponym recognition*, which is the subject of this chapter. Toponym recognition is difficult because many names of places are also common names of people and other entities. For example, “Paris” can refer to the French capital and many other places in the USA, but can also be a person’s given name (e.g., “Paris Hilton”).

Some variants of the toponym recognition problem consider other ambiguities occurring in natural language, such as the use of different words to refer to the same toponym in the same text, and require that these ambiguities be resolved. This situation can arise in historical texts concerning places whose names have changed over time. Another possibility is considering such phrases as “my house” versus “a house”—the first might be considered a toponym, since it is definite, referring to an actual place with geographic coordinates, while the second is indefinite. In the following discussion, we restrict ourselves to considering only toponyms consisting of proper names. Our techniques further make the simplifying assumption that within a single text, toponyms with the same name refer to the same place, sometimes called *one sense per discourse* [164]. Also, note that while geotagging relies on toponym recognition, it is only one stage in the process and should be regarded as a

means to an end. Therefore, we do not overly concern ourselves with toponym recognition performance in isolation, but rather with the performance of our geotagging system as a whole. This observation informs us against overly relying on algorithms specifically tuned for the toponym recognition problem.

Toponym recognition can be considered as a subset of a more general problem studied in natural language processing, called *named-entity recognition* (NER). Whereas toponym recognition involves finding entities in text that correspond to geographic location names, named-entity recognition involves finding locations, as well as entities of other types (e.g., names of people and organizations). In our example sentence “Jordan visited London last Friday”, the output from a toponym recognizer would include the location “London”, while correct output from a named-entity recognizer would also include “Jordan” as a person, and possibly “Friday” as a day of week. Sometimes evidence is stronger for a particular entity interpretation versus another interpretation. For example, in the pattern “X visited Y”, the “visited” verb lends credence to X being a person and Y being a location, since locations are visited by people. Machine learning–based NER systems will often discover patterns like these from corpora of documents that are annotated with entities, and use these patterns to build a language model by which entities and entity types can be predicted, given the linguistic context.

Given toponym recognition’s status as a subproblem of NER, tools developed for the more general problem of NER can be used for toponym recognition. In this case, the general strategy is to take an input document, execute an off-the-shelf NER system on the document (e.g., LingPipe [6], Stanford NER [37], ANNIE [31]), and take the location entities. State-of-the-art NER systems typically use statistical machine learning methods to train a language model from an annotated language corpus. These systems usually employ generative models such as hidden Markov models (HMMs) or discriminative models such as conditional random fields (CRFs). Once trained, the model is used to determine the most likely sequence of parts of speech, or most likely set of named entities. Of course, these

models will be inherently limited by the size, contents, and availability of suitable training data, which in many cases is quite limited.

Once location entities are found, location interpretations are assigned from a gazetteer, and in the toponym resolution step, one of the interpretations is chosen for each toponym. However, this simple strategy is problematic. Because NER is a more general problem, systems developed for NER tend to be tuned for this more general problem, rather than specifically for locations, so they may be less accurate in detecting locations. Second, this strategy is inflexible in that the toponym recognition and toponym resolution procedures are completely independent, and thus cannot share evidence. For example, it may happen that a supposed toponym  $t$  found by the toponym recognition procedure is incorrect, i.e.,  $t$  should not have been selected as a toponym. However, the toponym resolution procedure is then forced to consider  $t$  as a toponym and select one of the incorrect location interpretations of  $t$ , even if none of these interpretations are evidenced by  $t$ 's context. Returning to our example, in the sentence “Jordan visited London last Friday”, if “Friday” were incorrectly recognized as a toponym, the toponym resolution procedure would necessarily select the interpretation in Texas. A better option would be to allow the toponym resolution procedure to drop toponyms discovered by the toponym recognition method that are not evidenced.

Also, when evaluating NER systems on our domain of news articles (described in Section 4.3), we found that they were biased toward precision at the expense of recall. In other words, they miss many toponyms so that the ones that they report are valid. Also, statistical NER systems are usually trained on corpora of tagged news wire text that contain few less-prominent toponyms. As a result, the toponyms in an NER training corpus essentially serve as a very limited gazetteer, which in turn limits the breadth of a toponym recognizer using models trained on the corpus. This limitation drastically reduces their performance on articles from local newspapers, as also noted by Stokes et al. [140]. Also, the generally small size and homogeneity of corpora used in training NER systems do not capture the fast moving and ever changing nature of the news cycle. While this bias is not unacceptable

for NER, it is problematic when used in a geotagging system, since toponym recognition imposes an upper bound on recall for the entire geotagging process (i.e., toponym recognition and resolution). In other words, the geotagging procedure only has a chance to geotag a toponym correctly if it was recognized during toponym recognition, and any missed toponyms in the initial toponym recognition phase will negatively impact the recall for the entire process. As a result, the low recall of typical NER techniques severely limits the entire geotagging process's recall, and thus we saw the need for more comprehensive techniques. Of course, it is worth noting that information about entities other than toponyms is useful for toponym recognition, since it may offer a means of resolving type ambiguities.

Bearing these considerations in mind, the toponym recognition process we designed for processing streaming news has a considerably more flexible architecture. Our multifaceted toponym recognition process uses standalone NER software as only one of many recognition methods, of potentially varying quality. We include rule-based recognition in the form of entity dictionary tables, cue word matching (e.g., “X County”), and toponym refactoring. In addition, we leverage statistical NLP tools in the form of NER software with postprocessing filters, and part-of-speech (POS) tagging with additional recognition rules. These methods are described in detail in later sections, but we provide a brief overview here. After an initial tokenization step, our method proceeds by performing lookups into various tables of entity names, including location names and abbreviations (e.g., “Maryland”, “Md.”), business names (e.g., “Apple”, “Toyota”), common person names (e.g., “Chad”, “Victoria”), as well as cue words for the above types of entities (e.g., “X County”, “Mr. X”, “X Inc.”). We also refactor geographic names by shifting particular cue words (e.g., “X Lake” to “Lake X”).

In addition to the above rule-based methods, we leverage statistical NLP tools. We use an NER package to recognize toponyms and other entities, and perform extensive postprocessing on its output to ensure higher quality. We also perform part-of-speech (POS) tagging to find phrases of proper nouns, since names of locations (and other types of en-



tities) tend to be composed of proper nouns. The POS tagging also provides a means of recognizing additional grammatical forms that hint at entities' types, including active verbs and noun adjuncts, which we use as signals to adjust entity types. Furthermore, we incorporate evidence from other documents in the document's news cluster. At the end of the entire procedure, we attempt to reconcile entity types, and establish groups of entities to be resolved concurrently, by grouping textually similar entities together. Essentially, we designed this multifaceted toponym recognition procedure in keeping with our goals to be flexible enough to capture variations that occur in streaming news, as well as to be as all-inclusive as possible when recognizing toponyms, in order to maximize the procedure's recall (i.e., to miss as few toponyms as possible). Our toponym resolution procedures, described in later chapters, serve to restore precision to the process by dropping supposed toponyms with no supporting evidence for any of their possible interpretations.

Furthermore, during and after the recognition procedure, we allow entities to overlap, and even to share boundaries but with different types. For example, consider a document containing "Chad". We may have evidence that "Chad" is a person entity, as well as different evidence that "Chad" is actually a location entity. Rather than keeping only one type, we create two entities with the same boundaries but different types. In other words, we wait to resolve entity and interpretation conflicts as late as possible, so that we leverage as much evidence as possible to resolve these conflicts. Of course, recall is not the only factor in designing a robust toponym recognition system, and precision also plays a role. In our example document, if we came across a sentence containing "Mr. Chad Johnson", the "Mr." and "Johnson" provide strong evidence that "Chad" is a person entity and can be safely disregarded. However, in general, recall must be emphasized over precision due to toponym recognition's status as the initial stage of a combined geotagging process.

Our recognition procedure can be broken into two stages, around which this chapter is organized. First, we generate an initial set of possible entities using many sources of evidence (Section 4.1). Second, we execute a variety of postprocessing filters that attempt

to resolve entity types using additional forms of evidence (Section 4.2). Next, we describe how we incorporated our toponym recognition method into NewsStand, and evaluated it by comparing it against two state-of-the-art competing systems (Section 4.3). Finally, we discuss several open problems with regard to our methods (Section 4.4) and conclude the chapter (Section 4.5).

## 4.1 Finding Toponyms

In this section, we describe several methods of finding an initial set of potential toponyms, including both rule-based and statistics-based methods. At this point we are concerned mainly with finding as many potential toponyms as possible. Later, we apply additional filtering rules to remove some erroneous patterns while still maintaining overall high recall (described in Section 4.2). Note that in recognizing toponyms, we only consider exact, case-sensitive matches for groups of tokens. This strategy is acceptable for text domains such as news articles and the hidden Web, because documents in these domains tend to follow linguistic and grammatical rules for writing, but exact matching would be less suitable for other domains where these rules are followed less closely (e.g., blogs, tweets).

We begin with a description of tokenization (Section 4.1.1), followed by our methods for lookup into entity tables (Section 4.1.2) and an entity dictionary (Section 4.1.3). We then describe methods based on natural language processing, namely finding proper nouns (Section 4.1.4 and named-entity recognition (Section 4.1.5).

### 4.1.1 Tokenization

The first step in recognizing toponyms is to tokenize [61] the input document’s text. Tokenization involves breaking the text into meaningful parts, referred to as tokens, and a useful tokenization is more than simply splitting on whitespace. Consider the following dateline from a news article:

ALBANY, N.Y. (BP) — In a lengthy debate . . .

A useful tokenization of this text would result in tokens such as “ALBANY”, “(comma)”, “N.Y.”, “(open parenthesis)”, and so on, which is markedly different from a simple tokenizer based on whitespace. We use the regular expression–based tokenizer provided as part of the Stanford NLP package [37], which contains a grammar with a large number of rules for English natural language tokenization. After tokenization, we determine sentence boundaries in the text so that we avoid constructing comma groups across sentence boundaries. Like tokenization, finding these boundaries is ordinarily not simply a matter of finding periods. As our example above shows, periods and other punctuation sometimes appear in acronyms, abbreviations, and other linguistic forms. However, the tokenizer distinguishes in-token punctuation such as those present in acronyms from lone punctuation which makes sentence splitting trivial. Once we have tokens and sentence boundaries, toponym recognition becomes a matter of grouping adjacent tokens into toponyms.

#### 4.1.2 Entity Tables

After tokenization, we proceed with toponym recognition by looking for a curated, small set of well-known locations and other entities appearing in the document’s text, which serves as a convenient baseline for toponym recognition. This set of entities is gathered from several tables in our gazetteer, which is based on GeoNames [43], and is updated and kept current on a daily basis. In particular, we collect lists of continents, countries, and top-level administrative divisions (e.g., states, provinces), and search for them among the document’s tokens. In addition, we search for common abbreviations for all of the above (e.g., “California” can be abbreviated as “Calif.” or “CA”). We also search for *demonyms*, which are words used to refer to people from a particular place (e.g., “German”, “Marylander”). Demonyms, while not locations proper, have some aspect of location that can be useful in recognizing and resolving toponyms, in that the location they represent can contribute to an overall sense of locality for the document. We iterate over the document’s

tokens, looking for groups of tokens that match an entry in an entity table, and if we find such a match, we create an entity of the corresponding type. For location entities, we also associate each entity with the proper location interpretation from the gazetteer.

### 4.1.3 Entity Dictionary

Next, we recognize additional entities of many types by using an entity dictionary, containing names of entities that commonly appear in the news. We use this dictionary to recognize both toponyms and non-toponyms, because knowing that a particular entity strongly refers to a non-toponym is useful in resolving geo/non-geo ambiguities. For example, knowing that “Apple” is a famous brand name allows us to discount the possibility that “Apple” refers to a small city in Ohio, in the absence of strong evidence. In addition to particular instances of entities, the entity dictionary also contains many *cue word* patterns which serve as keywords to identify entities of various types. For example, the phrase “County of” strongly indicates that one or more following tokens corresponds to a location. We search for entities and cue words among the document’s tokens, and collect matches as entities. For cue words in the dictionary, we search for adjacent capitalized tokens as the corresponding entity. Our entity dictionary was constructed by observing the output from our toponym recognition and resolution processes and checking for recognition errors, to discover which geo/non-geo ambiguities proved most problematic in our domain. The entity dictionary is by no means complete, but it serves as a useful starting point for a toponym recognition process in the news domain. In addition, as we discover new sources of ambiguity, the dictionary is updated with new classes of entities, so it is always evolving.

Table 4.1 contains a set of entity types, examples of entities, and entity cue words present in the entity dictionary. All examples shown in the table are also names of various locations around the world, indicating the high degree of geo/non-geo ambiguity present in toponyms. In addition, we added many different forms of spatial cues to account for minor variations in how the cue words are used. For example, both “X Lake” and “Lake X”

Table 4.1: Sample entity patterns and types from our entity dictionary. In cue word patterns, *X* and *Y* refer to variables that will match words. Each non-cue example in the table is also the name of multiple locations present in our gazetteer, indicating the high level of geo/non-geo ambiguity in location names.

---

General entities:	Religions	Christian, Islam, Hindu
	Seasons	Spring, Fall
	Directions	South, Northeast, Midwest
	Days	Monday, Friday
	Months	March, August
	Timezones	EST, WEST
	Colors	Gray, Navy, Lime
Organization entities:	Brand names	Apple, Coke, Toyota
	News agencies	AP, UPI
	Terror groups	Hamas, Taliban
	Unions	NEA, PETA
	Government orgs	Congress, Army
	Postnominals	<i>X</i> Corp., <i>Y</i> Inc.
Spatial cues:	Populated regions	State of <i>X</i>
	Populated places	Town of <i>X</i> , <i>Y</i> City
	Comma groups	<i>X</i> and <i>Y</i> counties
	Water features	Gulf of <i>X</i> , <i>Y</i> Lake
	Spot features	<i>X</i> School, Mt. <i>Y</i>
	Universities	University of <i>X</i> at <i>Y</i>
	General	<i>X</i> -based, <i>Y</i> -area
Person entities:	Honorifics	Mr. <i>X</i> ; Ms <i>Y</i> ; Dr. <i>Z</i>
	Generational suffixes	<i>X</i> , Jr.; <i>Y</i> III
	Postnominals	<i>X</i> , KBE; <i>Y</i> , M.D.
	Job titles	Sen. <i>X</i> ; President <i>Y</i> ; Sgt. <i>Z</i>
	Declaratory words	<i>X</i> said; added <i>Y</i>
	Common given names	John <i>X</i> ; Jennifer <i>Y</i>

---

are common variants of the “Lake” cue. Universities are another special case because of the many ways in which they are specified in text, especially with multi-campus university systems. For example, “University of Maryland at College Park” might be written “University of Maryland, College Park”, “University of Maryland in College Park”, “University of Maryland—College Park”, or other similar ways. Each of these variants are encoded into the entity dictionary’s matching rules.

#### 4.1.4 Proper Nouns

Next, we use a POS tagger to find proper noun phrases, which are useful in recognizing locations because locations tend to consist of proper nouns. We search for sequences of proper noun tokens, and consider them as locations. In addition, because our tokenizer considers possessive forms (i.e., “’s”) and hyphens as distinct tokens, we include these elements in location names if they connect sequences of proper nouns as well. These are useful for capturing locations such as “Prince George’s County”, in which “’s” separates the proper noun sequences “Prince George” and “County”. In addition, we also consider simple prepositional modifiers as proper noun separators, which will capture phrases such as “University of Texas at Arlington”. For each proper noun phrase we find, we add an entity of type “proper noun phrase” to the entity pool for this document, since we cannot determine a more specific type using POS tags alone. Typical state-of-the-art POS taggers generally train and use statistical language models, such as hidden Markov models (HMMs), decision trees, and other techniques. We use TreeTagger [133], a decision tree-based POS tagger, trained on the Penn TreeBank corpus [91].

Obviously, not all proper noun phrases are locations, so this technique will be under-precise for toponym recognition in that it will capture many noun phrases that are not locations, such as names of people, organizations, and other entities. However, despite its lack of precision, finding proper noun phrases is consistent with our goal of high recall—that is, not missing any locations present in the document. At this stage of processing, we

are not overly concerned with precision in location recognition, since that will be restored in the toponym resolution step, where erroneous location interpretations will be filtered.

#### 4.1.5 Named-Entity Recognition

As a final toponym recognition method, we leverage tools developed to address the problem of *named-entity recognition* (NER). NER seeks to discover typed entities present in an input text, which usually includes at a minimum entities such as people, organizations, and importantly, locations. As noted earlier, NER methods have their limitations when used for toponym recognition, due to NER being a more general problem. However, in keeping with our philosophy of multifaceted toponym recognition, we include NER in our toponym recognition procedure. As an NER package, we use the Stanford NLP Group’s NER and IE package [37], which is built around a conditional random field (CRF) classifier [66]. We used the language model included with the Stanford NER distribution, a three-class classifier to find persons, organizations, and locations, which was obtained by training on a mixture of CoNLL, MUC-6 and MUC-7, and ACE corpora.

We feed the article text to the NER system, and save the person, organization, and location entities into our entity pool. To avoid frequently noisy output entities, we only keep the entities that have a minimum score of 0.95. One observation is that this NER method captured similar entities as found by collecting proper noun phrases (described in Section 4.1.4), a result which is not overly surprising as named entities tend to consist of proper nouns. However, using the NER system offers the benefit of determining entity types, in addition to simply finding entities. Knowing entity types helps to avoid geo/non-geo errors, as non-location entities can generally be disregarded.

Rather than simply using the output entities from the NER system directly, we perform a number of postprocessing steps that serve to avoid some common pitfalls with which the Stanford NER system has trouble. These postprocessing steps are executed sequentially and act as entity filters. For example, we found that some output entities were fragmented,

in that the boundaries were chosen incorrectly, erroneously including or excluding nearby tokens, and we created filters to address this and other problems. Each filter is described below. Note that scores and score thresholds mentioned in each filter’s description correspond to scores assigned by the Stanford NER package.

The following sections contain examples of entities presented within their textual context. For ease of presentation, we visually distinguish these entities using brackets. For example, in the text “In [College Park], the mayor...”, “[College Park]” refers to the entity under consideration, while the surrounding text serves as context. Capturing the distinction between entity and context will be important for several filters described below.

#### 4.1.5.1 Boundary Expansion

Oftentimes, the NER system will find an entity in the proper context, but select the entity boundaries incorrectly. For example, it may select “Equatorial [Guinea]” rather than the correct “[Equatorial Guinea]”. In this example, the selected entity was correct, but the boundaries were not correct. Furthermore, the specific context in which an entity was found can effect how the NER system selects boundaries for the entity. In other words, the NER system may extract  $e_1$  “[Equatorial Guinea]” in one part of the document, and  $e_2$  “Equatorial [Guinea]” in another, simply due to the linguistic context in which  $e_1$  and  $e_2$  were found. This filter attempts to correct these fragmentation errors by expanding entity boundaries using other entities found in the text. In particular, we try to expand entities that are substrings of other entities. In our example, we expand  $e_2$  (“Guinea”) to “Equatorial Guinea” because  $e_2$ ’s preceding token, “Equatorial”, matches the initial portion of  $e_1$ . Note that we do not expand across sentence boundaries.

In general, to accomplish this entity boundary expansion, we search for entities that are substrings of other entities. We say that an entity  $e_1$  *dominates* another entity  $e_2$  if  $e_2$  is a substring of  $e_1$ . First, we group entities together based on domination, so that entities which are substrings of each other are grouped together. Algorithm 4.1 provides a pseudocode



---

**Algorithm 4.1** Group entities according to dominance.

---

```
1: procedure GROUPENTITIES( $E$ )
   input: List of entities  $E$ 
   output: Set of entity buckets  $B$ 
2:   Initialize set of entity buckets  $B \leftarrow \{\}$ 
3:   Sort entities  $E$  by decreasing length
4:   for  $i \leftarrow 1 \dots |E|$  do
5:     for  $k \leftarrow 1 \dots |B|$  do
6:       if HEAD( $B_k$ ) dominates  $E_i$  then
7:          $B_k \leftarrow B_k \cup E_i$ 
8:         break to next  $i$ 
9:       else if  $E_i$  dominates HEAD( $B_k$ ) then
10:         $B_k \leftarrow B_k \cup E_i$ 
11:        HEAD( $B_k$ )  $\leftarrow E_i$ 
12:        break to next  $i$ 
13:       end if
14:     end for
15:     Add new bucket  $b$  to  $B$  with HEAD( $b$ ) =  $E_i$ 
16:   end for
17:   return  $B$ 
18: end procedure
```

---

listing for this procedure, named GROUPENTITIES. The output for GROUPENTITIES is a set of entity buckets  $B$ , with each bucket  $b$  containing a set of entities, one of which dominates all entities in  $b$  and is designated  $b$ 's *bucket head*, and is denoted HEAD( $b$ ). After initializing the set of output buckets  $B$  (line 2), we sort the entities in decreasing order of length (3). We iterate over each entity  $E_i$  and bucket  $B_k$ —note that initially, since  $|B| = 0$ , the inner loop is not entered when  $i = 1$  (4–16). First, we check whether HEAD( $B_k$ ) dominates  $E_i$ , and if so, we add  $E_i$  to  $B_k$  (7). Otherwise, if  $E_i$  dominates HEAD( $B_k$ ), we add  $E_i$  to  $B_k$ , and set HEAD( $B_k$ )  $\leftarrow E_i$  (11), since the dominance property is transitive and hence  $E_i$  will also dominate all entities in  $B_k$ . If we find an appropriate bucket  $b$  for  $E_i$ , we continue with  $E_{i+1}$ ; otherwise, we create a new bucket  $b$  with HEAD( $B_k$ ) =  $E_i$ , and add  $b$  to  $B$  (15). Eventually, all entities in  $E$  will have been placed into appropriate bucket.

Now that entities have been grouped into buckets based on dominance, we attempt to expand entities within buckets. We implemented two strategies for entity expansion, which

we term *strict* and *loose* expansion. Put simply, strict expansion means that we only expand entities in a bucket  $b$  if they contain enough nearby tokens so that they can be expanded to match  $\text{HEAD}(b)$ . On the other hand, loose expansion attempts to expand each entity in  $b$  using other entities in  $b$ . In particular, we compare each entity  $e \in b$  to each longer entity  $e' \in b$  in order of decreasing length, and we expand  $e$  to  $e'$  if the proper nearby tokens exist that make it equivalent to  $e'$ .

The advantage of strict expansion is that it ensures greater accuracy for expanded entities, since if expansion succeeds, it is unlikely that the expanded entity is erroneous, due to the larger number of tokens required for a successful expansion. However, strict expansion's major drawback is that the head entity of each entity bucket may be unique in the document, affording no opportunity to correct fragmentation errors present in entities in the bucket. That is, simply because an entity is long does not make it very relevant for the document as a whole. For example, consider a document where the NER system collected entities  $e_1$  “[College Park]” (correct),  $e_2$  “College [Park]” (incorrect), and  $e_3$  “[College Park’s Fire Department]” (correct). All these entities would be placed in the same entity bucket, with  $e_3$  as the bucket head. Under strict expansion, each of  $e_1$  and  $e_2$  would be compared with  $e_3$  only. Neither would be expanded, which is fine for  $e_1$ , but  $e_2$  would remain unexpanded and erroneous, since it could not be expanded to match  $e_3$ . However, under loose expansion, in addition to a comparison with  $e_3$ ,  $e_2$  would be compared to  $e_1$  and hence would be correctly expanded due to the appropriate preceding token “College”. To capture more of these cases, we use loose expansion in our entity expansion filter.

#### 4.1.5.2 Entity Prefixes/Suffixes

One problem with NER systems is that entity types may be chosen incorrectly—even for multiple instances of the same entity in the same document—due to differences in the way that entities are referenced. This problem is also known as *coreferencing*. In a linguistic context, this is known as *coreferencing*. For example, an article may initially mention the

person “Paul Washington”, and simply “Washington” later, though both refer to the same person. While the first can easily be recognized as a person due to the presence of both a given name and surname, the second entity may be incorrectly typed as location because it only consists of a surname that is also a common location name. Articles can also refer to people by their given name alone, especially when mentioning childrens’ names or the names of celebrities, since referring to a person by their given name reflects a higher level of familiarity or empathy. At times, organization names may also be typed incorrectly, as in “Kia Motor Cars” which is frequently referred to as simply “Kia”. The former is more easily recognized as an organization than the latter, which may be mistaken for a person’s name or even a location.

This filter attempts to correct these NER type errors for fragments of larger entities found elsewhere in the document. The filter proceeds by selecting source entities from which entity types will be propagated. The selected entities include person entities consisting of at least two tokens (given name and surname), and organization entities consisting of at least three tokens. Furthermore, only entities with scores of above 0.90 are selected, ensuring high quality among the source entities. After selecting the source entities, the first and last tokens are taken from each entity and associated with the source from which they were taken. Finally, entity types are propagated to low scoring entities by searching for entities with scores below 0.60 and containing one of the tokens extracted above. If such an entity  $e$  contains one of the tokens  $t$ ,  $e$ ’s type is set to the type of the entity from which  $t$  was taken. This procedure captures given names and surnames of person entities, as well as the primary portion of organization names. Because only the first and last tokens of each source entity are matched, the filter allows for partial matching of entities, which is useful given the NER system’s penchant for entity fragmentation.

## 4.2 Filtering Toponyms

After finding entities using a combination of the methods described above, we proceed with a sequence of filters that act as postprocessing to remove potential errors. Filters are applied in the order listed and are described in detail below.

### 4.2.1 Toponym Refactoring

Oftentimes, location names can be referred to in multiple ways. For example, locations of a particular type such as “county” often have the word “County” in their names. However, the position of “County” in the location name can vary by locale. For example, in the US, “County” often appears as a suffix, as in “Prince George’s County”. However, counties of Ireland often feature “County” as a prefix, as in “County Kildare”. In addition, abbreviations of “County” such as “Co.” are not uncommon in news articles. Furthermore, a specific type of spot location frequently mentioned in local newspapers are local public and private schools, and these may be written in any number of ways (e.g., “Walter Johnson HS”, “Walter Johnson High”). This filter’s purpose is to account for these entity name variations, and refactor (i.e., restructure) entity names to generate extra query names that will be matched properly in our gazetteer. The filter contains a list of regular expressions to match against entity names, and if a match is made, suitable substitutions are performed.

Table 4.2 contains some of the entity name patterns that are refactored by this filter. “ $\Rightarrow$ ” indicates that a name matching the first pattern will be refactored to a name matching the second pattern, with *X* indicating the word or phrase that is maintained. “ $\Leftrightarrow$ ” indicates that names matching the first pattern will be refactored to the second, and vice versa. The patterns fall into four main classes: prefix abbreviations, suffix abbreviations, suffix shifting, and school expansion. In prefix and suffix expansion, common abbreviations used in location names are expanded. For example, “Ft. Meade” would be expanded to “Fort Meade”. For suffix shifting, location suffixes such as “County” are shifted before and af-

Table 4.2: Entity names modified by the name refactoring filter. Cue words are expanded and shifted within the entity to generate new query names for each entity. Arrows indicate the match and action performed for each pattern.

First name		Second name
Co. <i>X</i>	⇒	County <i>X</i>
Dr. <i>X</i>	⇒	Doctor <i>X</i>
Ft. <i>X</i>	⇒	Fort <i>X</i>
Mt. <i>X</i>	⇒	Mount <i>X</i>
St. <i>X</i>	⇒	Saint <i>X</i>
<i>X</i> Co.	⇒	<i>X</i> County
<i>X</i> Twp.	⇒	<i>X</i> Township
<i>X</i> County	⇔	County <i>X</i>
<i>X</i> County	⇔	County of <i>X</i>
<i>X</i> Lake	⇔	Lake <i>X</i>
<i>X</i> Parish	⇔	Parish of <i>X</i>
<i>X</i> Township	⇔	Township of <i>X</i>
<i>X SchType</i>	⇒	<i>X SchType</i> School

ter the main location name, so a location such as “County Kildare” would be expanded to “Kildare County” and “County of Kildare”. Finally, school expansion searches for partial names of schools, which are indicated by a school name and a school type keyword, such as “Primary”, “Middle”, “MS”, or “High”. Note that the filter may erroneously match and expand query names for entities that are not locations. For example, “Co.” is also a common abbreviation of “Company” and as such frequently appears in business names. Thus, “Ford Motor Co.” will be incorrectly expanded to “Ford Motor County”. However, this erroneous expansion will not be overly problematic as it is in keeping with our goal of high recall in toponym recognition. That is, having erroneous query names such as “Ford Motor County” will not cause problems because they will be corrected by the toponym resolution procedure, either by not being present in the gazetteer, or by having little evidence for such interpretations.

### 4.2.2 Active Verbs

To distinguish between toponyms and other types of entities, we note that many entities tend to be *active*, in that they perform actions (e.g., people, organizations), while locations tend to be *passive*, in that they do not. For example, it would make sense for a person to “say” something, while in general it would not for a location to “say” something. Generally, the grammatical subject of an active voice verb can be thought of as performing the action described by the verb. We use the part-of-speech tags assigned by the POS tagger to find entities that perform actions, which in turn disqualifies them as toponyms.

To find active entities, the filter searches for entities followed by an active voice verb, or by an adverb and an active voice verb. In this way, the method effectively performs a limited shallow parsing of the sentence. For each such entity of type “LOC” (location), the type is reset to “NNPP” (proper noun phrase). In other words, the entity is no longer considered as a location. Note that this method does not provide evidence for a particular entity type—e.g., determining whether such an entity is a person or organization. However, since we are primarily concerned with distinguishing between toponyms and non-toponyms, this lack of evidence can be overlooked. Another caveat with this method, which we do not address here, is that it does not properly account for *metonymy*—toponyms that refer to non-location entities—which will be described further in Section 4.4.

### 4.2.3 Noun Adjuncts

Sometimes, the correct interpretation of toponym evidence itself is in question. For example, consider a sentence beginning: “In Russia, U.S. officials. . .” In this sentence, both “Russia” and “U.S.” refer to countries. However, consider that the form “Russia, U.S.” might be mistaken for a particularly common form of evidence termed *object/container* evidence, which can be briefly described as a pair of toponyms, one of which contains the other in a geographic sense. Considering this evidence interpretation, we might erroneously think that the phrase “Russia, U.S.” might refer to any of several populated places named

Russia in the US in New York, New Jersey, or Ohio.

To help clear up this evidential ambiguity, we use evidence by taking note of another grammatical concept, that of the *noun adjunct*. Noun adjuncts are nouns that function as adjectives by modifying other nearby nouns. In our example sentence, “U.S.” is a proper noun adjunct that modifies the plain noun “officials”. Because of its primary connection with “officials” through the noun adjunct relationship, using it in object/container evidence would not be warranted. By detecting noun adjuncts, we prevent toponyms acting as noun adjuncts from participating in other filters used in toponym resolution. We detect them by finding entities followed by a plain noun.

#### 4.2.4 Type Propagation

Having grouped entities into equivalence classes in the previous step, we now leverage these entity groups to improve the overall quality of entity and toponym recognition. Note that in a group of entities as determined above, some entities will have more specific types than others, due to the heterogeneous nature of our toponym recognition methods. For example, entities found using the POS tagger (i.e., selecting proper nouns, described in Section 4.1.4) will have an unknown type, while entities found using the NER system (described in Section 4.1.5) will have more specific types. We propagate entity types within each group to make the types within a group consistent, in a similar fashion as was done for the NER system’s postprocessing. Having consistent entity types is useful because though the entities in a group have the same referent, the context in which each entity reference appears differs. To propagate entity types, we examine entity types within each group. If a group  $g$  contains untyped entities as well as entities all of a single type  $t$ , we set the untyped entities to type  $t$ . However, if there are more than one type of entities in  $g$ , the types are not propagated. Compared with a simple type voting scheme (e.g., setting the types of all entities in a group to the most frequent entity type), this scheme ensures a high quality of type propagation, since conflicts disqualify type propagation.

### 4.2.5 Lat/Long Assignment

Once we have groups of tokens that were recognized as potential toponyms, we assign location interpretations to toponyms in the form of latitude/longitude values and other location metadata by lookup into a large primary gazetteer of locations. For each toponym, we keep all possible matches from the gazetteer. We currently use the GeoNames [43] gazetteer, a collaborative gazetteer project which contains as of this writing over 8 million entries for locations around the world. In addition to lat/long values, each entry contains additional metadata that will be useful in toponym resolution (Chapters 5–7), such as feature type (e.g., country, city, river, mountain), population, elevation, and positions within a political geographic hierarchy. For example, the “College Park” entry contains pointers to its containers at increasing levels of scope: Prince George’s County, Maryland, United States, North America. We store and query the GeoNames gazetteer in a PostgreSQL database. We also impose a default ordering for the location interpretations of individual toponyms according to our notion of the “prominence” of location interpretations—based on their population. GeoNames also contains over 5 million alternate names, or aliases for locations, in a variety of different languages (though we currently only process English text). In addition to a lookup of each toponym, we also use particular cue words to perform *keyword expansion* on the recognized toponyms. For example, on finding a phrase such as “X, Y, and Z counties”, we would lookup “X County”, “Y County”, and “Z County”, rather than simply “X”, “Y”, and “Z”. As the example shows, this expansion is necessary because redundant or implied toponym types are often omitted from text where the linguistic context makes the types clear. After this final gazetteer lookup, we have a set of toponyms with associated location interpretations. In subsequent toponym resolution steps, described in later chapters, we use various techniques to decide which interpretation for each toponym is correct.



### 4.3 Evaluation

We incorporated our own toponym recognition methods into the NewsStand system [143], and compared with those of two prominent competitors: Thomson Reuters’s OpenCalais and Yahoo!’s Placemaker. Although both OpenCalais and Placemaker are closed-source commercial products, and do not make public how they work, they provide public Web APIs which allow for automated geotagging of documents, with relatively liberal rate limits. As a result, they have been used extensively in state-of-the-art geotagging and entity recognition research (e.g., [3, 97, 119, 148, 156]). Placemaker provides a toponym recognition service, while OpenCalais performs recognition of toponyms, and recognition of other types of entities as well. In addition, both OpenCalais and Placemaker are full geotagging systems—that is, they perform toponym resolution as well. While toponym resolution is an important problem in its own right, in this chapter, we are only concerned with toponym recognition, and instead defer evaluation of toponym resolution to later chapters. As a result, even though OpenCalais and Placemaker assign lat/long values to each toponym reported as output, we disregard these lat/long values in our evaluation. In other words, we use OpenCalais and Placemaker in their toponym recognition capacity only, and do not include toponym resolution in their performance scores. Also note that at the time of writing, neither OpenCalais nor Placemaker offered a means of tuning the precision/recall balance, so we could not explore this aspect of the systems. From our experimental results described in Section 4.3.4, it appears that these systems are tuned for precision, but we could not verify this over a range of precision/recall values due to lack of tuning capability.

We continue by first describing existing geotagged corpora (Section 4.3.1). Next, we examine toponyms in a large subset of NewsStand’s collection, as measured by our own toponym recognition method as well as OpenCalais and Placemaker (Section 4.3.2). Then, we describe a new corpus of annotated news articles that we created from NewsStand’s streaming news collection (Section 4.3.3). We conclude with accuracy measurements for all methods in two corpora of annotated news articles, and in streaming news (Section 4.3.4).

### 4.3.1 Existing Corpora

To get a sense of the corpora used in geotagging research, we present Table 4.3, which contains a listing of researchers and the corpora they used in their geotagging-related research. For each corpus, we give the source and total number of annotated documents and toponyms. In some cases, the exact numbers of documents and toponyms were not possible to determine due to lack of detail, and are shown in the table as “?”. Also note that the sources listed in the table were often used by multiple researchers, and here we present only an example usage of each source. The table reveals the relatively small sizes of annotated corpora used in geotagging research, with the number of annotated documents and annotated toponyms having averages of about 347 and 2,811 and maxima near 1,000 and 7,000, respectively. These numbers stand in stark contrast to the huge volume of news retrieved by NewsStand in just a single day, which is roughly 40,000 documents and 250,000 toponyms. Furthermore, most corpora include articles from only one or two news sources, usually newswire, which amounts to a heavily biased sample, given the variety and number of news sources and writing styles all over the world. Finally, the sources chosen for annotation reveal a prevalent English language bias.

However, one commonality that is apparent from the values in Table 4.3 is that the average number of toponyms in each article is remarkably consistent, with each article having 7–8 toponyms with few exceptions. This range is especially prevalent for corpora consisting of news articles, which is our domain of interest. One exception includes the Wikipedia corpus of Overell and R uger [110], with an average of 1.4 toponyms per article. However, Overell and R uger only considered toponyms in each article that also correspond to links to other Wikipedia pages; since generally only the first instance of an entity mentioned in an article is linked, this explains the seeming lack of toponyms. Other anomalous measures are the 15.1 and 15.2 toponyms per article reported by Roberts et al. [123] and Zong et al. [167]. In the former case these are likely due to the consideration of locations nested within other entities as toponyms (e.g., “[New York] Police Department”). Another, unfortunate

Table 4.3: Corpora used in geotagging-related research, showing sources, and document and toponym counts. Note that document and toponym counts refer to annotated counts, not total counts. Unknown values are denoted with “?”.

Work	Source	Docs	Topos	T / D
Amitay et al. [7]	Web pages	600	7,082	11.8
Buscaldi and Magnini [22]	L’Adige	150	1,042	6.9
Buscaldi and Rosso [23]	GeoSemCor	186	1,210	6.5
Garbin and Mani [42]	Gigaword	165	1,275	7.7
Gouvêa et al. [46]	Folha	230	?	?
Leidner [68]	RCV1	946	6,980	7.4
Lieberman et al. [81]	LGL	588	4,793	8.2
Liu and Birnbaum [84]	Google News	24	33	1.4
Manov et al. [89]	News	101	792	7.8
Martins et al. [97]	CoNLL’02/’03	?	2,813	?
Overell and R�uger [110]	Wikipedia	1,000	1,395	1.4
Roberts et al. [123]	ACE’05	369	5,562	15.1
Sobhana et al. [136]	IITKGP-GEOCORP	200	?	?
Volz et al. [154]	Reuters	250	?	?
Weichselbraun [156]	Reuters	?	?	?
Zong et al. [167]	DLESE	50	760	15.2
Average		347	2,811	8.1

commonality among the corpora used in geotagging research is that most are unavailable due to copyright restrictions, thereby making direct algorithmic comparisons on the same data generally not possible. In addition, a better measure for how frequently toponyms occur in text would be the ratio of toponyms to words, which would better account for variations in news article length. However, this data was not often presented by authors. Nonetheless, 7–8 toponyms serves as a useful rule of thumb for the number of toponyms expected in articles of reasonable length.

### 4.3.2 Toponym Statistics

Now that we have characterized typical toponym counts in news articles, our next experiment determines whether NewsStand’s geotagger has performance that approaches our expectations in terms of toponym recall. To measure performance, we sampled seven days’ worth of news from various days in November 2010, and executed NewsStand’s geotagger on the news articles collected on each day. The days were chosen randomly, except we ensured that we had at least one of each day of the week, to account for the typically lower volume of news published on weekends. We collected articles from news feeds that published at least five articles on each sampling day, to ensure a measure of consistency among the collected data. Furthermore, we limited the sampling to articles with at least a word count of 300, which ensures a reasonable minimum length for the news articles and served to filter out erroneously-processed documents (e.g., articles that had been improperly extracted from their HTML source). Sampling in this fashion resulted in filtering out about half of each day’s articles.

For each set of sampled articles, we tabulated the total number of toponyms recognized by NewsStand’s toponym recognition process. Table 4.4 reflects these counts. “Sources” indicates the number of sampled news sources from which sampled articles were taken. For each day, we include the total number of toponyms reported by our recognition method that have at least one interpretation in our gazetteer. The last column contains the toponym-

Table 4.4: Counts of articles, distinct sources, and geotagged toponyms for several days’ worth of news, sampled at different time periods.

Date	Docs	Sources	Topos	T / D
02 Nov 2010	27,591	2,086	207,110	7.5
06 Nov 2010	13,355	1,245	124,430	9.3
10 Nov 2010	28,795	2,182	208,366	7.2
15 Nov 2010	26,052	1,952	195,669	7.5
19 Nov 2010	24,193	2,018	173,630	7.2
23 Nov 2010	26,937	2,067	194,804	7.2
28 Nov 2010	14,245	1,250	148,996	10.5

document fraction—the number of toponyms with gazetteer interpretations over the number of sampled articles containing those toponyms.

We make several interesting observations from these statistics. First, and most importantly, we see that the majority of sampled days have toponym fractions between 7.2 and 7.5, which fall precisely in our expected range of 7–8 toponyms. The outliers of 9.3 and 10.5 are not totally unexpected given that they were measured on weekends which imply a different pattern of news publication. Overall, the measured toponym fractions are strong indications that our toponym recognition method identifies an appropriate number of toponyms. Next, in examining the number of articles and sources, the numbers show that our sampling resulted in a large number of articles from a variety of sources on each day, which demonstrates the extreme variety in our article samples. This stands in contrast to the small size and homogeneity of corpora used in previous geotagging-related research (described in Section 4.3.1), and the large number and variety of articles lends weight to the credence of our measured toponym fractions.

This evaluation method can be easily applied to very large collections of articles, making it ideal for continual testing of performance on streaming and ever-changing collections of news. Of course, it says nothing of how many of the toponyms are correct, which is addressed in Section 4.3.4.

### 4.3.3 CLUST: A New Corpus of Streaming News

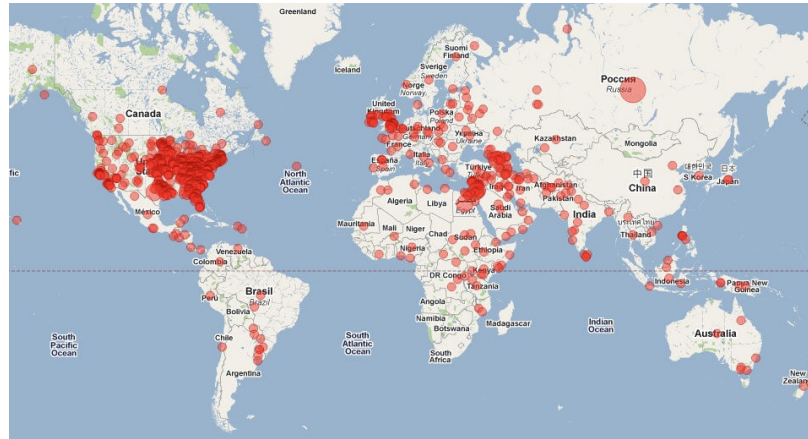
As outlined in Section 4.3.1, existing corpora used in geotagging research tend to have small sizes and are usually only from one or two news sources, and these deficiencies limit their power in characterizing toponym recognition performance. As a result, we leveraged NewsStand’s constantly streaming news data to create our own corpus. We used two corpora in our evaluation. For the first corpus, we used *LGL*, introduced in previous work [81] and used for evaluating toponym resolution on the local scale. This corpus consists of 621 articles from 114 local newspapers, with a total of 4,765 annotated toponyms. The creation and content of *LGL* is described in more detail in Chapter 6, but we provide a brief overview here. The goal in creating *LGL* was to create a collection of news from smaller news sources, rather than the major news sources typically used in creating article corpora, since the former significantly outnumber the latter on the Web. As a result, *LGL* is useful for testing the accuracy of our toponym recognition method for a variety of smaller news sources. However, it does not capture the larger, major news stories that are often described and published in multiple news sources. Note that these major news stories naturally form clusters in NewsStand, and it is not unusual to have clusters of 100, 200, or even 1,000 articles for especially major and ongoing news stories.

To capture these stories, we created another corpus consisting of sizable clusters of news articles found by NewsStand, and termed *CLUST*. To create *CLUST*, we selected clusters that had sizes of 5–100 articles, and contained articles from at least four unique news sources. The clusters were sampled between January and April 2010. This sampling strategy ensures reasonable cluster sizes which allows for enough variation among articles in the cluster. Furthermore, having multiple news sources ensures that different news sources are used, rather than many copies of the same article everywhere, which might result from erroneous preprocessing. In total, we sampled 1,080 clusters containing a total of 13,327 news articles, from 1,607 distinct news sources. For each cluster, we randomly selected one article for manual annotation, resulting in 1,080 annotated articles containing

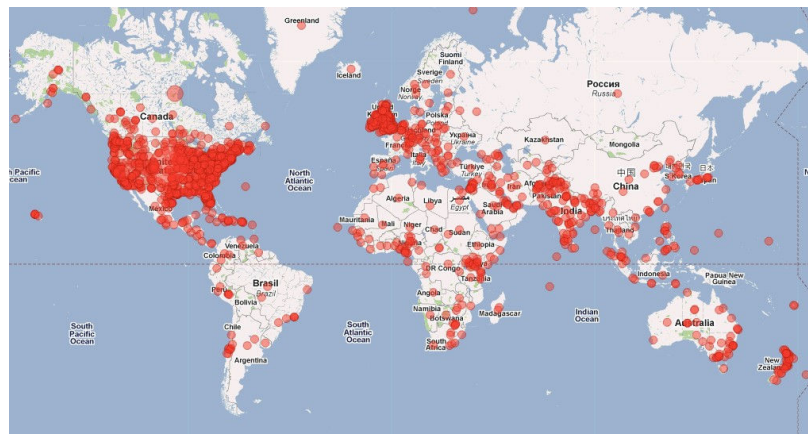
11,962 toponyms, with a median of 8 toponyms per article. Because multiple news sources and by extension their audiences are represented in each cluster, we expect the stories in CLUST to have more journalistic impact, as well as a wider geographic significance, than the stories in LGL.

Table 4.5 summarizes and compares statistics for the LGL and CLUST corpora. CLUST has roughly twice as many annotated articles, and toponyms, as LGL. However, the most striking difference between LGL and CLUST is the composition of toponym types in each corpus. Since LGL was created as a corpus of articles from smaller newspapers, and CLUST as a corpus of larger news stories, we expect the toponyms in LGL to correspond to smaller places, and those in CLUST to correspond to larger places. The type statistics in Table 4.5 reflect these expectations. Nearly half of annotated toponyms in LGL correspond to cities, and of those toponyms, two-thirds are cities under 100,000 population. On the other hand, CLUST, consisting of larger news stories, has only 33% of toponyms corresponding to cities, and of those toponyms, nearly two-thirds are cities over 100,000 population. In addition, the fractions of country and state toponyms in CLUST are larger than those in LGL, while the fraction of county toponyms in LGL is larger than those in CLUST. These measurements reflect our motivations for using LGL and creating CLUST, and show that these corpora, used together, allow for an effective evaluation on both smaller and larger news stories from a variety of news sources.

Next, we characterized the geographic distribution of annotated toponyms in LGL and CLUST. Figure 4.1 contains two maps of toponym lat/long values present in the annotated articles of LGL and CLUST. The maps show a clear bias toward English-speaking areas, which reflects NewsStand's English bias. Despite this bias, the map illustrates that our corpora contain toponyms from all over the world, and experiments on these corpus will show the toponym recognition method's effectiveness for a variety of geographic areas.



(a) LGL



(b) CLUST

Figure 4.1: Geographic distribution of annotated toponyms in the LGL and CLUST corpora.



Table 4.5: Content of the LGL and CLUST corpora, used for evaluating recognition accuracy.

	LGL	CLUST
Articles	621	13,327
News sources	114	1,607
Annotated docs	621	1,080
Annotated topos	4,765	11,564
Distinct topos	1,177	2,320
Median topos per doc	6	8
Location types:		
Total topos	4,765	11,564
City	2,287	3,837
$\geq$ 100k pop	756	2,377
$<$ 100k pop	1,531	1,460
Country	911	3,540
State	784	2,487
County	525	519

#### 4.3.4 Toponym Accuracy

Having established the credibility of our two evaluation corpora, we next examine our toponym recognition method’s accuracy and compare its performance to that of OpenCalais and Placemaker. For each of NewsStand, OpenCalais, and Placemaker, we consider two versions of each method: the original algorithm, referred to as, e.g., “NewsStand”, and the original algorithm with a postprocessing filter that removes output toponyms if they have no interpretations in our gazetteer, denoted with a subscript  $G$ , e.g., “NewsStand $_G$ ”. By doing so, we determine the effect of using a gazetteer on toponym recognition, as well as characterize to some extent the gazetteers used by OpenCalais and Placemaker.

Like many natural language and text processing problems, toponym recognition performance can be cast in terms of two widely-used measures called *precision* and *recall* [153]. For a set of ground truth toponyms  $G$  and a set of system-generated toponyms  $S$ , precision

and recall are defined as:

$$P(G, S) = \frac{|G \cap S|}{|S|}, R(G, S) = \frac{|G \cap S|}{|G|}$$

Put simply, precision measures how many reported toponyms are correct, but says nothing of how many went unreported. In contrast, recall measures how many ground truth toponyms were reported and correct, but does not indicate how many of all reported toponyms are correct.

An ideal toponym recognition system would have perfect precision and recall. However, implementations of such systems often exhibit a tradeoff between precision and recall that can be adjusted by tuning one or more system parameters. To illustrate this tradeoff, consider a toponym recognition system that finds and reports toponyms from an article, along with a confidence score indicating its level of certainty that the reported toponym is correct. The system’s output can then be filtered using a threshold score  $S_t$  by dropping output toponyms whose confidence scores fall below  $S_t$ . Precision can be favored by setting  $S_t$  to a large value, while recall is favored with a small value of  $S_t$ . Thus we see that  $S_t$  can be varied according to which measure is more important for a given application.

In our geotagging system, toponym recognition precedes toponym resolution. Thus, the toponym recognition stage has the effect of providing an upper bound on the recall for the entire geotagging process, since toponyms can only be resolved correctly if they were recognized. We therefore seek to maximize the recall for this stage, possibly at the expense of precision errors. In our toponym resolution stage, we make use of local lexicons and other contextual clues to improve toponym resolution precision (see Section 6.2).

To combine these measures into a single score, we rely on another commonly-used derived measure, the  $F_\beta$ -score [153]:

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

The  $F_\beta$ -score allows us to allocate the importance between precision and recall via the  $\beta$  parameter. For example,  $F_{0.5}$  allocates more importance to precision, as might be needed for an information retrieval application for the general public, while  $F_2$  allocates more importance to recall, which might be needed for expert data analysts who can tolerate precision errors. To combine precision and recall into a single measure for comparative purposes, we use the  $F_1$ -score [153], which is simply the harmonic mean of  $P$  and  $R$ :

$$F_1 = \frac{2PR}{P + R}.$$

In addition, we consider two different criteria for determining whether a ground truth toponym  $g$  matches a system-generated toponym  $s$ . The first, termed *exact* matching, states that  $g$  and  $s$  are equivalent if the starting and ending offsets of each are equal. The second, termed *overlap* matching, relaxes this criterion by allowing  $g$  and  $s$  to simply overlap in their offset ranges for them to match. Both are useful in characterizing the performance of toponym recognition. Exact matching could be considered the gold standard for measuring performance. However, overlap matching is sometimes necessary to avoid improper penalization due to gazetteer differences and other factors. For example, consider a ground truth toponym “[New York state]” and system-generated “[New York] state”, which is correct, but is not an exact match and is an overlap match. Overlap matching serves a similar purpose as methods such as BLEU [111], in that partial matches are not overly penalized.

We measured all algorithms’ performance over both the LGL and CLUST corpora. Table 4.6 contains results for the LGL corpus. In addition, for  $|G \cap S|$ ,  $P$ , and  $R$ , exact and overlap matching are reported as two numbers in the table in “E/O” form ( $|S|$  is unaffected by the matching method used). Comparing NewsStand against OpenCalais and Placemaker reveals that both NewsStand variants greatly outperform their competitors in terms of toponym recall, having at least 0.10 and in some cases 0.20 or higher recall over OpenCalais and Placemaker, when measured using both exact and overlap matching. NewsStand’s high

Table 4.6: Toponym recognition performance for the LGL corpus ( $|G| = 4,765$ ). In all cases, the NewsStand variants have highest toponym recall.

	$ S $	$ G \cap S $ (E/O)	$P$ (E/O)	$R$ (E/O)
NewsStand	23,345	3,879/4,645	0.166/0.199	<b>0.814/0.975</b>
NewsStand <sub>G</sub>	5,960	3,619/3,738	0.607/0.627	<b>0.759/0.784</b>
OpenCalais	1,959	1,830/1,871	0.934/0.955	0.384/0.393
OpenCalais <sub>G</sub>	1,873	1,757/1,791	0.938/0.956	0.369/0.376
Placemaker	4,593	3,129/3,683	0.681/0.802	0.657/0.773
Placemaker <sub>G</sub>	3,796	3,013/3,112	0.794/0.820	0.632/0.653

recall comes at the expense of toponym precision; however, remember that in NewsStand, toponym recognition is only considered one stage of an integrated geotagging process, and toponym precision is restored by later stages of processing. The gazetteer postprocessing done for NewsStand<sub>G</sub> demonstrates this effect, dramatically improving precision with little corresponding decrease in recall. In addition, as mentioned earlier, our geotagging procedure is based on that of Lieberman et al. [81], who report a precision over 0.80 and correspondingly high recall for LGL, thus showing that precision is indeed restored.

Examining OpenCalais’s and Placemaker’s performance, we see that these methods are much more biased toward toponym precision at the expense of recall, which is taken to the extreme in the case of OpenCalais (i.e., at least 50% less than NewsStand). Note that NewsStand and Placemaker are comparable in terms of  $F_1$ -score, while OpenCalais’s is lower, illustrating the potential precision/recall tradeoff. Also, performing gazetteer postprocessing for OpenCalais<sub>G</sub> has little effect, while for Placemaker<sub>G</sub>, a significant boost in precision is noted using exact matching, along with a significant decrease in recall when using overlap matching. These results seemingly indicate that Placemaker’s toponym matching rules differ from our own, though we cannot be sure due to the closed-source nature of Placemaker. Examining differences between exact and overlap matching, we see that NewsStand and Placemaker are significantly affected by allowing overlap matches, while OpenCalais and all the gazetteer-filtered algorithms (i.e., NewsStand<sub>G</sub>, OpenCalais<sub>G</sub>, Placemaker<sub>G</sub>) are

mostly unaffected. For NewsStand, this is likely due to dropping many non-toponyms that were selected by NewsStand’s filters (e.g., proper noun phrases not present in the gazetteer). Comparing  $|S|$  of NewsStand and NewsStand<sub>G</sub>, a very large number of toponyms were dropped by the gazetteer filtering, which accounts for the hefty precision increase. For Placemaker, gazetteer and matching differences can account for the performance difference.

Table 4.7 contains performance results for the CLUST corpus. The NewsStand algorithms again outperform their competitors in terms of recall, by an even larger margin than was seen for LGL, while OpenCalais and Placemaker are tuned for toponym precision. In addition, examining differences between LGL and CLUST, we see that the performances scores for CLUST are generally higher across all algorithms than the corresponding scores in LGL, with the only exception being Placemaker’s recall. This difference indicates that in some sense, CLUST’s toponyms are easier to recognize than those of LGL, likely due to the greater presence of large, easily recognized toponyms such as country names.

### 4.3.5 Streaming Evaluation

We have shown that NewsStand’s multifaceted toponym recognition procedure has a high recall for articles from both small, local news sources (LGL) as well as larger, better-known sources (CLUST). However, measuring performance over an entire static corpus

Table 4.7: Toponym recognition performance in the CLUST corpus ( $|G| = 11,564$ ). As with LGL, NewsStand had highest recall.

	$ S $	$ G \cap S $ (E/O)	$P$ (E/O)	$R$ (E/O)
NewsStand	44,184	10,243/11,330	0.232/0.256	<b>0.886/0.980</b>
NewsStand <sub>G</sub>	13,589	9,909/10,036	0.729/0.739	<b>0.857/0.868</b>
OpenCalais	6,452	6,208/ 6,326	0.962/0.980	0.537/0.547
OpenCalais <sub>G</sub>	6,060	5,843/ 5,941	0.964/0.980	0.505/0.514
Placemaker	9,796	6,782/ 8,549	0.692/0.873	0.586/0.739
Placemaker <sub>G</sub>	7,466	6,469/ 6,593	0.866/0.883	0.559/0.570

does not well reflect day-to-day toponym recognition performance on a constant stream of news data. To better characterize day-to-day performance, we split the CLUST corpus into weekly samples of articles, and measured precision and recall for NewsStand<sub>G</sub> using overlap matching over each sample. Effectively, this test determines whether the NewsStand method would perform well if executed within that time range.

Figure 4.2 shows the performance of our toponym recognition procedure on the CLUST corpus, measured over time. Performance in terms of both precision and recall is relatively consistent over all time periods tested, with a mean of 0.739 precision and 0.868 recall. In addition, the standard deviations of precision and recall are 0.029 and 0.018, which serve as further evidence of the method’s performance stability. These results indicate that the NewsStand toponym recognition process is well suited for streaming news.

#### 4.4 Open Problems

Recall that our multifaceted toponym recognition procedure includes methods for recognizing active verbs following entities, and disqualifying the entity as a location if it is followed by an active verb. For example, for the phrase “[Paris] said”, we would consider “Paris” to be a non-location, since in general, locations do not perform actions, while other entities do, such as people and organizations. However, this method has a significant caveat in that it does not properly account for *metonymy* associated with toponyms. Metonyms are a frequent occurrence in articles about, for example, local or international politics, where a government may be referred to by the city of its primary geographic presence. For example, toponyms such as “Washington”, “Westminster”, and “Hollywood” do have location interpretations, but are often used to refer to non-location entities—i.e., the US Government, the UK Parliament, and US cinema, respectively. These organizations can be considered active entities, as opposed to passive locations. As a result, in a sentence such as “Washington stated on Monday...”, “Washington” would be disqualified as a toponym. However, we note that repeated instances of “Washington” would likely provide a means of correct-

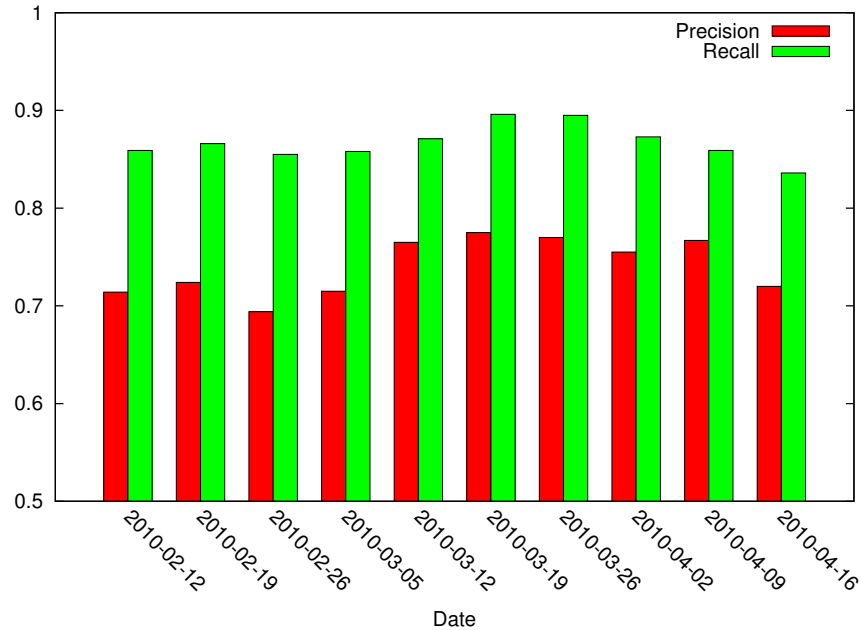


Figure 4.2: Toponym recognition performance for the CLUST corpus measured over time.

ing this error, as metonymic references are relatively uncommon in text [71]. As a result, additional instances of “Washington” would not likely be metonymic, and could be used to correct the earlier error through a voting scheme. Alternatively, we could incorporate a metonymy recognition method into this filter, such as that proposed by Leveling and Hartrumpf [71].

Additionally, we could leverage other tools from natural language processing, in particular by performing a *shallow parse* of each sentence, resulting in groups of nouns, verbs, and other relatively simple structures. Shallow parsing creates additional linguistic structure which can be used in geotagging algorithms, to create groups of tokens that is a step above the simple tokenization that we currently perform, all while retaining relatively good structural accuracy as compared to full sentence parsing. Such parsing would allow association of verbs with their subjects, which would further aid in correct toponym recognition. Also, by extracting sentence clauses, we make evidential ties between toponyms present in the same clause. In our example fragment “In [Russia], [U.S.] officials...”, “Russia” and “U.S.” are present in separate clauses, which would preclude their incorrect association.

Further, we plan to perform a more in-depth investigation of the individual components of toponym recognition used in our procedure, to determine their overall utility, as well as their performance for specific classes of toponyms (e.g., countries, states, large cities). We could also incorporate additional rule-based filtering into a framework such as ANNIE [31], which contains facilities for rule-based matching via grammars. We also plan to investigate our heuristics' use within other NLP applications, such as coreference analysis [98], to determine their suitability in this domain, as well as incorporate them within a machine learning framework. Our toponym recognition methods could be recast in terms of these frameworks, which would allow easy extension and modification for matching parameters.

## 4.5 Summary

In this chapter, we introduced a toponym recognition method that serves as the initial step of a two-part geotagging process, and is followed by the toponym resolution methods described in Chapters 5–7. This multifaceted toponym recognition method is especially suited for the streaming news domain, which poses special challenges. In particular, streaming news is constantly in motion and ever-changing, which advises against the sole use of methods based on static corpora of news. Our recognition method involves many sources of evidence, and in our evaluation, was shown to outperform its competitors in terms of toponym recall, the crucial measure of success.



## Chapter 5

### Comma Groups

In Chapter 4, we described our methods of multifaceted toponym recognition—that is, discovering which words in the article correspond to location names—which serves as the first part of a two-stage geotagging process. Here, we shift our attention to *toponym resolution*—choosing the correct geographic interpretation for each toponym—and explore different types of evidence used by writers to express geographic information about toponyms.

As outlined in Chapter 1, many systems and methods have been developed for geotagging text. These methods tend to apply a variety of heuristics modeled on the evidence typically provided by document authors to help their human readers recognize and resolve toponyms. For example, one very common technique is to search the text for names of especially large or populous places (e.g., country names), as listed in an external database of geographic locations, and resolve them immediately. Another common strategy is to recognize “object/container” pairs of toponyms within the text (e.g., “Paris, France”). Of course, these and other strategies cannot be used in isolation because of the significant potential for errors. Consider the following opening sentence from a news article [44] in the Paris News, a small newspaper based in Paris, Texas:

Madison Sikes, a 5-year-old from Paris, is receiving one Christmas present early this year.

Clearly, resolving “Paris” to the most populous interpretation in France would result in an

error, and additional information is needed to resolve it correctly—in this case, using other toponyms in the article, or using information about the source newspaper’s geographic location. More evidence is needed for most such heuristics used in geotagging text.

In this chapter, our aim is to recognize and resolve toponyms organized using another method commonly employed by authors: *lists*, which for the purposes of exposition we refer to as *comma groups* (though commas need not always separate list items). Comma groups are a natural way to organize groups of related information. In fact, we note that each comma group unifies the entities it contains through a *common thread*—attributes that are shared by all entities in the group. This reasoning leads to our observation that for comma groups of toponyms, these common threads greatly aid in resolving the toponyms correctly. For example, despite each toponym in the comma group “Rome, Paris, Berlin and Brussels” having many possible interpretations (over 40, 60, 130, and 10, respectively), the common thread of large, prominent capital cities allows us to select the correct interpretations. Similarly, the group “Hell’s Kitchen, Chinatown, Murray Hill, Little Italy, and SoHo”, despite containing individually ambiguous location names, exhibits the common thread of neighborhoods in southern Manhattan, New York City, and this allows their correct resolution.

Furthermore, and even more importantly, we observe that unlike typical forms of heuristic evidence used in recognizing and resolving toponyms, comma groups are often self-specified, in that they can be resolved reliably and accurately by inferring their common threads. That is, for a comma group of toponyms, if the common thread is identified correctly, the group’s toponyms can be resolved without relying on other, potentially erroneous toponym resolutions made in the rest of the document. This identification is made easier because of the large number of toponyms in the comma group (three or more). Since all toponyms exhibit the group’s common thread, each additional toponym acts as another sample against which a potential common thread can be compared. These comparisons are especially useful for large comma groups of five, ten, or even twenty toponyms, which are

not uncommon in a variety of text documents on the Web. Thus, despite their seeming triviality, comma groups deserve special attention when geotagging documents, because they offer a means of high quality toponym resolution.

In this chapter, we study comma groups in their own right. We describe methods to recognize comma groups of toponyms and to identify their common threads to effect correct toponym resolution. To do so, we first recognize comma groups of toponyms (Section 5.1) by searching for toponyms in the input text using several methods developed for geotagging text, and then finding toponyms joined by suitable separator tokens. Next, we resolve these comma groups (Section 5.2) by identifying their common threads using one of three heuristics based on the geographic attributes of each group's contained toponyms: their *prominence*, their *proximity*, and *sibling* relationships in a geographic hierarchy. Being heuristic in nature, these techniques do result in errors from time to time, and for each heuristic we provide examples of successes and errors found in news articles taken from the NewsStand system [143] (Chapter 3). We also present the results of a comma group usage evaluation (Section 5.3) for a sampled portion of a large dataset of news articles collected over a two month period from online news sources, indicating the utility of each heuristic for recognizing and resolving comma groups of toponyms. In particular, the proximity and sibling heuristics play a large role in recognizing and resolving comma groups of toponyms. Finally, we present some open problems related to this work (Section 5.4) and conclude the chapter (Section 5.5).

## 5.1 Comma Group Recognition

Our comma group recognition process is intended as a way to find comma groups of entities, regardless of the entities' types. Later, in our comma group resolution procedure (Section 5.2), we determine whether comma groups contain toponyms or non-toponyms, and if they contain toponyms, to choose the correct interpretations of each toponym using comma group heuristics. Note that prior to recognizing comma groups, we perform a topo-

nym recognition step and assign lat/long values for interpretations from our gazetteer, both of which are described in detail in Chapter 4.

We employ several strategies to recognize comma groups, drawing on a variety of internal linguistic structure and external knowledge sources. In general, we found that the more sources of evidence used for recognizing toponyms and comma groups, the better the final results will be. In other words, toponym and comma group recognition most benefit the entire comma group geotagging process by having high recall at the cost of precision—that is, reporting as many potential comma groups as possible, even potentially erroneous ones. Furthermore, because written language found on the Web and in hidden Web text repositories varies in the way that comma groups are written (e.g., “X and Y and Z” versus “X, Y, and Z”), some looseness in toponym and comma group recognition rules is also warranted. Our recognition procedures reflect this requirement.

Our comma group recognition process searches for groups of three or more toponyms, all separated by suitable separator tokens, and all in the same sentence. The separator tokens used include commas and conjunctions, such as “and” and “or”. At times, articles such as “the” and “a” also appear before toponyms in comma groups, such as in “France, the USA, and Singapore”. These words are also allowed after separator tokens by our group recognition rules. Despite its simplicity, this recognition process is fairly robust to errors because of the requirement for multiple toponyms in the group. Furthermore, it is not used in isolation, but is the first step in a combined recognition and resolution process. In other words, groups of toponyms erroneously tagged as comma groups will be filtered in the comma group resolution stage (Section 5.2), when no suitable common thread can be found for the comma group.

To ease our exposition, we present pseudocode for our group recognition algorithm, named `FINDCOMMAGROUPS` and listed as Algorithm 5.1. Input for `FINDCOMMAGROUPS` includes an input document and list of toponyms  $T$  recognized for the sentence under consideration, and it produces a set of comma groups  $O$  as output. To find the groups, toponyms

---

**Algorithm 5.1** Find comma groups.

---

```
1: procedure FINDCOMMAGROUPS( $E$ )
   input: Input document, list of toponyms  $T$ 
   output: Set of comma groups  $O$ 
2:    $E \leftarrow \text{SORTBYSTARTOFFSET}(T)$ 
3:    $G \leftarrow \{T_1\}$ 
4:   for  $i \leftarrow 2 \dots |T|$  do
5:     if  $\text{SUITABLESEPARATOR}(T_i, G_{-1})$  then
6:        $G \leftarrow G \cup \{T_i\}$ 
7:       continue
8:     end if
9:      $O \leftarrow O \cup \{G\}$ 
10:     $G \leftarrow \{T_i\}$ 
11:  end for
12:   $O \leftarrow O \cup \{G\}$ 
13:  return  $\{g \in O : |g| \geq 3\}$ 
14: end procedure
```

---

are first sorted by their starting offset position within the document (represented by `SORTBYSTARTOFFSET` in line 2). A single pass is then made through the toponyms in order of increasing offset, creating comma groups along the way (4–11). In the loop,  $G$  refers to the current comma group that we are constructing, and is initialized to the first toponym  $T_1$  (3). For each toponym  $T_i$ , we check whether  $T_i$  is separated from the last toponym added to  $G$  (denoted as  $G_{-1}$ ) by suitable separator tokens, i.e., a comma or coordinating conjunction (shown as `SUITABLESEPARATOR` in line 5), and if so, we add  $T_i$  to  $G$  and continue with the next toponym (6–7). Otherwise, we terminate the comma group  $G$ , adding it to the output set  $O$ , and reinitialize  $G$  to the single toponym  $T_i$  (9–10). After all toponyms have been examined and groups added to  $O$ , we simply return the groups in  $O$  with at least three toponyms as true comma groups, and disregard the rest (13).

`FINDCOMMAGROUPS` makes one pass over the toponyms  $T$  and thus has runtime  $\mathcal{O}(T)$ . Also note that `FINDCOMMAGROUPS` does not impose strict rules on the individual group separators used: Any combination of separators are allowed in constructing comma groups, so that “ $V, W$  and  $X, Y, \text{ or } Z$ ”, “ $X$  or  $Y$  or  $Z$ ”, and “ $X, Y, \text{ and } Z$ ” would all be recognized and analyzed. This reasoning contrasts with a recognition process that, e.g., searches

for toponyms strictly of the form “X, Y, and Z”. Furthermore, our process does not consider differences in the particular conjunctions, articles, or other separators being used. That is, “and” is equivalent to “or” for the purposes of comma group recognition.

However, for comma group recognition, the above looseness is intentional and is necessary because of the difficulty in predicting the way individual authors construct comma groups. Various writing styles and editorial standards dictate a surprising variety of ways in which comma groups are written, even for the relatively limited domain of news articles. However, as noted earlier, enough evidence is given by the multiple toponyms in comma groups to choose the proper interpretations for spatial comma groups, and incorrect interpretations can be quickly filtered.

Furthermore, this comma group recognition procedure is not necessarily exclusive of other types of recognition processes. Geographic language and spatial forms abound in most news articles and in many other document domains. For example, another common type of geographic evidence that appears frequently in the news is the “object/container” form, where a geographic place is suffixed by its container, as in “Zurich, Switzerland”. Clearly, this type of evidence overlaps and may be confused with comma groups, since their separators (commas, conjunctions) and toponyms may coincide. Authors also mix comma groups with object/container forms, as for “Chicago, Atlanta, Louisville, Ky., and Buffalo, N.Y.”. We present further examples of mixed evidence used in comma groups in the comma group resolution section (Section 5.2). Therefore, in a system for correctly geotagging this text, the process of recognizing and resolving comma groups should be done in parallel with other processes for examining different types of evidence, and the most-evidenced result used for output.

## 5.2 Comma Group Resolution

Resolving comma groups amounts to finding the common thread binding the group together. Finding this common thread may be quite difficult for an arbitrary comma group, as

the contained entities may be of any type and have any connection. However, for comma groups of toponyms, the situation is more manageable, as we have observed that for much text on the Web, toponyms related through comma groups tend to share geographic attributes as well. As a result, our strategy for resolving comma groups involves checking whether the toponyms in each group follow particular toponym heuristics. In Sections 5.2.1–5.2.3, we present three heuristics harnessing useful geographic attributes of location interpretations for toponyms in comma groups:

1. *Prominence* of location interpretations, based mainly on population, where larger is better;
2. *Proximity* in terms of the geographic distance between location interpretations, where closer is better;
3. *Sibling* location interpretations that share a parent in a geographic hierarchy.

To resolve toponyms in a comma group, we check the toponyms using each of these heuristics in the order listed, stopping when we find a set of location interpretations that satisfies the heuristic under consideration. If no such interpretations are found for any of the three heuristics, we consider the comma group to contain non-toponyms.

Note that our resolution checks are done without knowing the true types of entities in each comma group. However, if the entities in the group truly are toponyms, their types will be readily apparent due to the mutual evidence imparted by the heuristic checks. That is, the evidence given by location interpretations of toponyms in comma groups tends to be apparent, and hence it is difficult to mistake non-toponym comma groups for toponym comma groups and vice versa. Furthermore, the geographic evidence for particular interpretations of the toponyms is mostly independent of global or external evidence such as the overall geographic focus of the document being geotagged. That is, the comma group can be thought of as a highly local, self-specified form of geographic evidence. Furthermore, these interpretations are much clearer with a large number of entities in the comma group,

since the additional toponyms and toponym interpretations serve as more evidence toward a location interpretation of the comma group. Large comma groups of five, ten, and even twenty toponyms are not uncommon in textual domains such as news articles.

In addition, for each of our three heuristics described in the following sections, we provide several examples of comma groups in news articles from the NewsStand system (Chapter 3) that were resolved using the heuristic, including examples of where an initial geotagging using the heuristic caused toponym resolution errors. These examples are intended to illustrate that our heuristics, while useful for a large number of comma groups found in text, are not infallible and are not intended to be used completely in isolation from additional geographic evidence. The resolution errors presented here were later fixed using additional evidence, such as additional gazetteers and geographic information about the source document, and we describe how each was addressed.

### 5.2.1 Prominence

Our first test is for collective *prominence* of location interpretations within the comma group. This prominence check is intended to select interpretations in the *global lexicons* (see Chapter 6) of most readers—that is, locations that would be known to a majority of readers without additional qualifying evidence. For the purposes of this work, we deem continents, countries, and other places with a population greater than 100,000 as “prominent”. We check whether all toponyms in the group have a prominent location interpretation, and if so, resolve the toponyms in the group accordingly.

Obviously, this definition of prominence based primarily on population has its problems, as many large places around the world would not be considered prominent and lead to erroneous location interpretation selection when used with large gazetteers. For example, for US readers, “Salem” would most likely be interpreted as a city in Massachusetts (famous for the eponymous witch trials of the late 17th century) or as the capital of Oregon. However, the most populous interpretation of “Salem” is actually a city in Tamil Nadu,



India, with over 1.5 million population. This India interpretation dwarfs those of Massachusetts and Oregon, which have about 40,000 and 150,000 residents, respectively. From this example we see that the concept of prominence is more nuanced than simply raw population, and that a more involved measure is needed to capture these cases.

Note that simply checking for consistent—i.e., prominent—interpretations of all toponyms in the group will be problematic for large comma groups. As the number of toponyms in the group increases, the likelihood also increases that one or more toponyms will not be matched properly, due to a variety of reasons. For example, the gazetteer may be incomplete and not contain location records for a given toponym, or it may not contain all aliases of a given toponym, such as “Big Apple” when referring to New York City. Other mismatches can result from typos, misspellings, and other language errors, which, though uncommon in the news articles we examined, did appear from time to time.

To account for these possibilities, note that the requirement for all comma group toponyms to have a prominent location interpretation is overly strict. For example, if we found 18 of 20 toponyms in a comma group have a prominent interpretation, we should still consider the comma group as one of prominent locations. In particular, if we find a subset  $G_p$  of the toponyms  $G$  in the comma group that have a prominent location interpretation, such that  $\frac{|G_p|}{|G|} \geq 0.75$ , we resolve each toponym in  $G_p$  to its prominent interpretation, and suppress all interpretations for the remaining toponyms  $\overline{G_p} = G \setminus G_p$  as erroneous. In other words, for the toponyms in  $\overline{G_p}$ , we do not choose e.g. the most populous interpretation, but instead do not report them as locations. This suppression may result in geotagging recall errors, since the toponyms in  $\overline{G_p}$  go unreported. However, given that comma group evidence is self-specified, and that we have determined the comma group’s common thread of prominent locations, suppressing these non-prominent interpretations is a reasonable action. This action also avoids potential precision errors, which are undesirable for casual usage.

Figure 5.1 contains several examples of prominence comma groups from various sampled news articles and containing a variety of prominent locations around the world. The

Shot in Las Vegas, Mumbai, New Mexico and Los Angeles, Kites also stars  
...  
... as well as distinctive parks in Boston, Detroit, Milwaukee, Chicago, Atlanta,  
Louisville, Ky., and Buffalo, N.Y.  
... in and around Louisville and Lexington, Kentucky, Nashville and Cordova,  
Tennessee, Richmond, Virginia, Fort Lauderdale and Orlando, Florida, Indi-  
anapolis, Indiana and Atlanta, Georgia.

Figure 5.1: Examples of prominence comma groups.

first example comes from an article [145] in The Hindu about a movie called Kites and contains a comma group of prominent locations where the filming took place. Notice that this group’s locations are well-known, prominent places, rather than sharing geographic characteristics such as proximity or containment. The second example, taken from an Associated Press article [114] about the landscape architect Frederick Law Olmstead, mentions multiple US cities in which Olmstead designed urban parks. In addition to having prominent cities, this comma group contains two object/container references, namely “Louisville, Ky.” and “Buffalo, N.Y.” which were resolved separately. This example illustrates the mixed forms of location resolution evidence that sometimes appear together—in this case, comma group and object/container. Our final example, from a press release [106] posted in the Earth Times online newspaper, shows where relying on prominence evidence alone can go wrong. This document contains another mixture of comma group and object/container evidence that caused our geotagger to erroneously tag “Cordova” to Córdoba, Spain, instead of the correct interpretation of Cordova, Tennessee. The error was caused by initial improper recognition of the type of evidence intended to be used to resolve the locations in the comma group. However, note that this comma group as written is difficult to parse even for humans, and especially so for humans unfamiliar with the locations in the group.

The notion that comma group evidence is self-specified may not be strictly true, and an improved comma group geotagging algorithm could use knowledge from additional sources. For example, if we know that the article in question comes from the local news

section of a newspaper, we might instead not allow the above global prominence measure to play a role, since comma groups of prominent places tend not to appear in these articles. Furthermore, for these articles, we might take into account other meanings of prominence rather than simply global prominence. For example, locations in the reader’s *local lexicon*—smaller, local places that the reader and other nearby readers know about (further described in Chapter 6)—could be considered “prominent” and might appear in these articles. However, given that these locations will tend to be geographically proximate, this case will be covered by the proximity rule described in the following section.

### 5.2.2 Proximity

Our second comma group rule involves a test for mutual geographic *proximity* of toponyms within a comma group. That is, we wish to find a set of interpretations for all the comma group toponyms that meet some test for proximity. In contrast to the prominence comma groups described previously, proximity comma groups tend to appear frequently in news articles about smaller, local places.

For our proximity test, we iterate over potential location interpretations for the first toponym  $t_1$  in the comma group  $G$ , and check whether the remaining toponyms  $t_i, 2 \leq i \leq |G|$  have an interpretation within a distance threshold  $d$  of the first. In each iteration, we initialize an output set of interpretations  $L$  to the single pair  $(t_1, loc_1)$ . Next, we iterate over the remaining toponyms  $t_i, 2 \leq i \leq |G|$  in the group. For each such toponym, we check whether  $t_i$  has an interpretation  $loc_2$  where  $\text{DISTANCE}(loc_1, loc_2) \leq d$ , and if so, we add  $(t_i, loc_2)$  to the set of output interpretations  $L$ . We currently use a threshold of  $d = 50$  miles. Finally, after all toponyms  $t_i$  have been examined, we check whether all toponyms in the group have a viable, proximate location interpretation (i.e., whether  $|L| = |G|$ ), and if so, use  $L$  as the interpretations for this comma group. Note that for each toponym, we check and add location interpretations to  $L$  according to the default ordering from our gazetteer lookup. This ordering ensures reasonable results despite the greedy nature of our algorithm.

This simple resolution algorithm does have its drawbacks in that it applies a uniform distance test, without regard to a human perception of nearness. Different humans tend to have different ideas about what is near and far [93, 103]. For example, a person from Manhattan, New York, who is accustomed to walking or subways for transportation, would have a different conception of distance than a person from, e.g., Helena, Montana. The proximity algorithm could reflect these differences by having a variable distance threshold based on, e.g., the geographic area of interest. We plan to evaluate various other factors to determine an appropriate, human-based conception of proximity.

Figure 5.2 contains two examples of comma groups resolved correctly using geographic proximity, and one where the proximity test resulted in errors. The first excerpt comes from an article [15] in *ThisWeek* of central Ohio, and mentions three cities in Delaware County, Ohio, namely Delaware, Powell, and Sunbury. Note that despite the city of Delaware sharing its name with the better-known state of Delaware, its presence in the comma group and its common thread of geographic proximity allowed us to select the correct interpretation. Our second example, from an article [138] in the *Santa Cruz Sentinel*, contains a comma group of several small cities in Santa Cruz County, including “Ben Lomond”. Here, even though “Ben” is a common given name, we recognized and resolved “Ben Lomond” using the proximity rule. However, the third example shows the limitations of naively applying proximity. This example, an excerpt from an article [85] about stargazing in *HeraldNet*, an online newspaper based in northwest Washington state, mentions several stars and constellations, including Vega, Altair, and Deneb. Interestingly, these are also the names of three

...and all three Delaware County historical societies — in [Delaware](#), [Powell](#) and [Sunbury](#).

It took more than an hour for fire crews from [Boulder Creek](#), [Ben Lomond](#), [Felton](#) and [Zayante](#) to control the blaze.

...you can still see the Summer Triangle of stars, [Vega](#), [Altair](#) and [Deneb](#), which are the brightest stars in their respective constellations.

Figure 5.2: Examples of proximity comma groups.

mountains in the Star Mountains range of Indonesia and Papua New Guinea, which also contain a number of other peaks named after stars, and it is this range to which the star names were initially tagged. We later resolved these errors by discounting the mountain interpretations as highly unlikely, given that HeraldNet is a small news source based in Washington state.

### 5.2.3 Sibling

Our third and final comma group check is for toponym interpretations that are of the same geographic type and share a parent container within a geographic hierarchy, which we term *sibling* interpretations. Siblings include states in the same country, counties in the same state, and so on down the geographic hierarchy. We found that sibling comma groups appeared in a variety of contexts, whether local, national, or international. Siblings that are high in the hierarchy, such as countries, are already recognized properly by the prominence test described earlier, so the sibling test is intended mainly for smaller location interpretations such as counties. Note that sibling locations need not be proximate. For example, New York and California are siblings, both being US states, but are not geographically proximate. Likewise, proximate locations are not necessarily siblings, as in the case of Ontario, a Canadian province, and New York, a US state, despite their being geographically adjacent.

The sibling test can be best viewed as a counterpart to the proximity test described in the previous section, and the algorithm is likewise similar. As before, we iterate over interpretations  $loc_1$  of the first toponym  $t_1$  in the comma group  $G$ . For each remaining toponym  $t_i$ ,  $2 \leq i \leq |G|$ , we check whether  $t_i$  has an interpretation  $loc_2$  that is a sibling of  $loc_1$ —that is,  $loc_2$  is of the same type and has the same parent as  $loc_1$ —and if so, select it as the interpretation for  $t_i$ . If we find suitable interpretations for all toponyms in  $G$ , we select these interpretations as the correct resolutions for the toponyms. We again check interpretations using the default ordering imparted by our gazetteer lookup.

In Figure 5.3, we present excerpts from articles where the sibling rule was applied,

The California Zephyr stops in Burlington, Mount Pleasant, Ottumwa, Osceola and Creston.

... as well as the Athens, Macon and Columbus areas.

But the Hawks persevered, earning big wins over Taylorsville, Skyline and Jordan.

Figure 5.3: Examples of sibling comma groups.

two of which were correct, and the last was initially wrong. The first excerpt, from an article [113] in the Des Moines Register about passenger trains, contains mentions of a number of cities in southern Iowa served by a train route called the California Zephyr. Even though the correct interpretations of these cities straddle southern Iowa and are not considered proximate, and furthermore the word “California” appears close by, suggesting (erroneous) interpretations of place names in the state of California, the fact that all lie in Iowa, and hence that all are sibling cities, allowed their correct resolution. Our second example comes from an article [2] from 11alive.com, an NBC affiliate in Atlanta, Georgia. This article mentions three relatively distant cities, but since all are siblings with a parent of Georgia, correct resolutions were achieved. Furthermore, note that Athens and Columbus have much more prominent interpretations in Greece and Ohio respectively, but their presence in the comma group allowed us to select the correct interpretations. Our final example is from an article [48] in the Salt Lake Tribune in Salt Lake City, Utah, concerning high school basketball competitions. The excerpt mentions “Taylorsville”, “Skyline”, and “Jordan”, which in fact refer to high schools in a local school district, rather than location names. However, they were initially erroneously tagged to three small localities in Kentucky. This example demonstrates a situation where relying solely on sibling evidence can be misleading. As with the previous proximity errors, these were resolved by incorporating extra filtering based on the source newspaper’s location, which would not warrant interpretations in Kentucky for a story highly local to Salt Lake City.

### 5.3 Usage Evaluation

To further investigate our comma group heuristics' utility for geotagging text on the Web, we implemented them in a geotagger. Note that this geotagger was designed only to recognize and resolve comma group toponyms, and did not incorporate any other methods of recognizing and resolving toponyms. Normally, comma group geotagging would be incorporated into a larger geotagging framework that draws on a wider variety of evidence. In this fashion, we tested comma group geotagging's utility as an isolated process. In later chapters, we explore comma groups' utility when compared to other sources of evidence in a full geotagging process.

Using the geotagger, we processed a sample of two months' worth of news articles gathered from RSS feeds of English language news sources on the Web. These news sources include newspapers large and small, newswire feeds, and blogs of various types, mostly based in the US. Table 5.1 presents several statistics about our dataset of articles and comma group usage within these articles as determined by the geotagger. In total, our sampled subset consisted of approximately 87,000 articles that were geotagged with at least one comma group of toponyms. Furthermore, in this sampled subset, 106,000 comma groups and 435,000 comma group toponyms were resolved using our heuristics. These counts demonstrate that comma groups play a nontrivial role in resolving toponyms from news articles. One caveat with these measurements is that they only reflect comma groups that were recognized and resolved by the geotagger, and says nothing of how many were missed. Furthermore, comma groups incorrectly recognized as containing toponyms instead of other entity types are also included in these counts. Further experiments with annotated articles are needed to better determine the frequency of comma groups in this text.

Table 5.1: Comma group usage statistics.

Sampled articles	87,405
Comma groups of toponyms	105,701
Toponyms part of a comma group	434,657

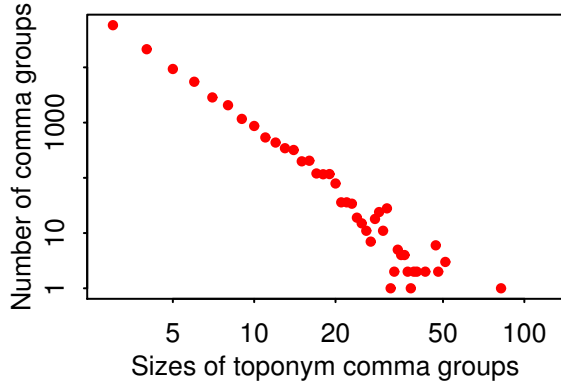


Figure 5.4: Comma group sizes for our article dataset.

We also measured the sizes of comma groups in our article dataset, and these measurements are presented in Figure 5.4. Note the log scale for both axes. As the figure shows, a large number of comma groups have relatively small sizes, and a smaller number are exceptionally large. However, note that a sizable number of comma groups are quite large, with about 25% of the 106,000 recognized comma groups having five or more toponyms, and the largest—from a report [122] posted on the Earth Times website—having 82 toponyms. As noted earlier, these large comma groups prove especially useful in resolving the contained toponyms correctly, since each additional toponym provides additional evidence toward determining the correct common thread.

Next, to investigate the accuracy of our three comma group heuristics, we randomly selected three samples of 20 articles that contained at least one prominence, proximity, and sibling comma group, respectively. For each sample, we manually verified whether the comma groups present in each article contained correctly or incorrectly resolved to-

Table 5.2: Heuristic precision measurements.

Heuristic	P(Groups)		P(Toponyms)	
Prominence	19/20	(0.95)	135/136	(0.99)
Proximity	18/20	(0.90)	67/ 71	(0.94)
Sibling	19/20	(0.95)	71/ 74	(0.96)
Total	56/60	(0.93)	273/281	(0.97)



ponyms as a measure of our comma group geotagging’s precision. If an article contained several comma groups, we randomly chose one to evaluate. For this evaluation, a comma group was considered correct if all its toponyms were recognized and resolved correctly, and incorrect otherwise. Table 5.2 contains the results of this verification, with precision numbers reported both in terms of comma groups and comma group toponyms. Bearing in mind our somewhat small sample size, overall precision of these heuristics in our sample of documents is quite high, at about 95% or higher for all three heuristics, indicating that inferring common threads of comma groups can be a source of highly accurate evidence for geotagging toponyms. As before, recall was not tested, so these measurements carry the same caveat described earlier. Interestingly, the toponym counts reflect the considerably larger comma groups present in the prominence sample, which were due to large comma groups of countries present in those articles.

Finally, for the 106,000 comma groups recognized by the geotagger, we measured how often each resolution heuristic was employed to resolve comma group toponyms. For comma groups where more than one heuristic applied to the contained toponyms, we give priority for the most specific (i.e., geographically local) heuristic used to resolve the toponyms, since geographic locality is additional evidence for the correct toponym interpretations. In particular, for comma groups to which both the prominence and proximity heuristics applied, we counted the group for proximity. Similarly, for comma groups containing both prominent and sibling toponyms, we counted the group toward the sibling heuristic, since siblings tend to more geographically localized. Our usage results are listed

Table 5.3: Heuristic usage measurements.

Heuristic	Count	Fraction
Any	105,701	1.000
Proximity	12,728	0.120
Sibling	51,423	0.486
Prominence	41,550	0.393

in Table 5.3. From our measurements, we found that 51,000, or approximately 49%, of comma groups were resolved using the sibling heuristic. Of the remaining comma groups, about 42,000 (39%) were resolved using prominence, and 13,000 (12%) were resolved using the proximity heuristic. These counts demonstrate that while prominence plays some role in recognizing and resolving comma groups, proximity and sibling evidence cannot be ignored. All three heuristics are needed to resolve comma groups correctly.

## 5.4 Open Problems

In this section, we describe some open problems related to comma groups. Note that currently, the prominence and proximity heuristics use static thresholds to identify common threads. However, human notions of prominence and proximity vary depending on context [93, 103], and these variations may be reflected in comma groups intended for readers in different geographic regions. For example, locations considered to be prominent in a rural area may not be thought to be prominent in urban areas. Similarly, the concept of “far” for a person living in an urban area may be on the order of blocks, while in a rural area, “far” may signify tens of miles. Extending this idea, it may be natural and correct to allow looser interpretations of prominence and proximity for comma group interpretations of rural places, and this looseness could be determined by factors such as population density or region sizes. Thorough investigations of these ideas are warranted.

Also, on a more basic level, it would be interesting to see how many different structural varieties of comma groups are present in news articles, to see whether more general recognition rules can be devised to capture them. In some sense, the adaptive context methods described in Chapter 7 generalize the comma group methods described in this chapter, but are somewhat overkill in that the former also match toponyms that are more distant than those in comma groups. In general, these methods need to be reconciled to determine at what point they become alike.

## 5.5 Summary

In combination with our other geotagging heuristics described in the subsequent chapters, comma groups are important and useful sources of evidence that aid the accurate geotagging of text, and recognizing and resolving comma groups is greatly aided using distance-based proximity and container hierarchy-based sibling heuristics, in addition to population-based prominence.

## Chapter 6

### Local Lexicons

In the previous chapter, we explored one type of evidence we use to correctly resolve comma groups of toponyms. Here, we introduce another such form of evidence. First, recall that there are many approaches to the geotagging process (e.g., [7, 69, 78, 118, 121]). Two prominent ones are MetaCarta [121] and Web-a-Where [7]. MetaCarta assumes that a toponym such as “Paris” corresponds to “Paris, France” approximately 95% of the time, and thus reasonably good geotagging can be achieved by placing it in “Paris, France”, unless there exists strong evidence to the contrary. On the other hand, Web-a-Where assumes that the text document being geotagged contains a number of proximate geographic locations often of the nature of a container (e.g., the presence of both “Paris” and “Texas”) that lend supporting evidence to each other. These approaches performed quite poorly in our evaluation domain of corpora of news articles, which motivated our research.

The key observation that we make in this chapter is that news articles (and more generally, documents on the Web) are written to be understood by a human audience, and therefore geotagging will benefit from processing (i.e., reading) the document in the same way as an intended reader. In doing so, the geotagger’s seemingly daunting task of identifying the correct instance of “Paris” out of the more than 60 possible interpretations will be much easier when we note that the reader is unlikely to even be aware of most of these interpretations, and thus there is no need to even consider them as possibilities in the toponym resolution step.

This leads to the key point in this chapter which is that the reader’s *spatial lexicon*—those locations that the reader can identify and place on the map without any evidence—is very limited. In fact, even more importantly, this inherent limitation means that a common spatial lexicon shared by all humans cannot exist, which is one of the key principles used by systems such as MetaCarta and Web-a-Where. To illustrate the importance of understanding readers’ spatial lexicons, consider the following opening in an online May 2009 newspaper article [151]:

PARIS — Former champion Serena Williams and Jelena Jankovic led Saturday’s women’s winners at the French Open tennis tournament in Paris.

For this article, “Paris” does refer to “Paris, France”. However, consider the following contemporary article [146] published in the Paris News, a local newspaper in Texas:

Restoration of the historic Grand Theater marquee in downtown Paris is gaining momentum.

This instance of “Paris” actually refers to the city in “Texas”, which typical readers would recognize immediately, since the correct interpretation of “Paris” exists in their spatial lexicon. For these articles, MetaCarta would erroneously place “Paris” in “France” as it assumes that “Paris” refers to “Paris, France” 95% of the time, even to readers living in “Paris, Texas”, which is clearly not true. On the other hand, Web-a-Where assumes a single spatial lexicon consisting only of very prominent places around the world and does not consider local possibilities, such as “Paris, Texas”, at all.

In essence, our key premise is the existence of a reader’s *local spatial lexicon* or simply *local lexicon* that differs from place to place, and that it is separate from a *global lexicon* of prominent places known by everyone. In other words, to readers in Texas, “Paris” refers primarily to “Paris, Texas”, rather than the distant, but more prominent, geographic location—“Paris, France”. Furthermore, in most cases, the local lexicon supersedes the global lexicon. For example, as shown in Figure 6.1, the local lexicon of readers living in “Columbus,



Figure 6.1: The local lexicon for readers living in the vicinity of Columbus, Ohio, USA. Notice the many local places that share names with more prominent places elsewhere.

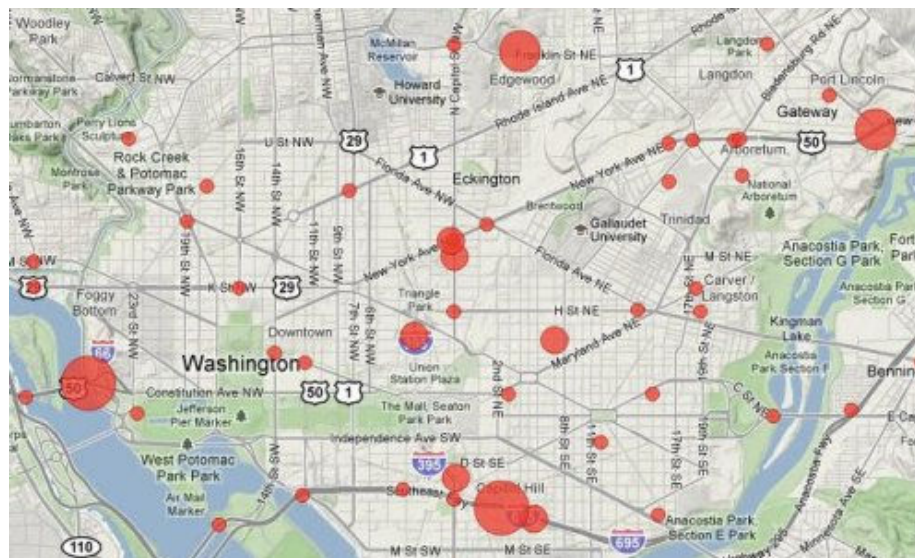


Figure 6.2: Traffic hotspots in the Washington, DC area obtained by geocoding address intersections from tweets.

Ohio” includes places such as “Dublin”, “Amsterdam”, “London”, “Delaware”, “Africa”, “Alexandria”, “Baltimore”, and “Bremen”. In contrast, readers outside the Columbus area, lacking the above places in their local lexicons, would think first of the more prominent places that share their names. The local lexicon is even more necessary when geographically indexing locations with smaller spatial extent which correspond to address intersections as shown in Figure 6.2, since street names are even more ambiguous than regular toponyms. In this chapter we present algorithms developed in tandem with the NewsStand system (Chapter 3) that automatically identify the local lexicons of document sources on the Web, which, according to our experimental analysis, leads to significant improvements in geotagging accuracy.

In the rest of this chapter, we first describe our methods for automatically inferring the local lexicons associated with news sources (Section 6.1). Next, we describe how to leverage these local lexicons within the context of a rule-based toponym resolution framework, and how local lexicons affect other sources of resolution evidence (Section 6.2). After, we describe experiments to evaluate our methods for inferring local lexicons, toponym recognition, and toponym resolution, using existing and new corpora of streaming news for this purpose (Section 6.3). Finally, we discuss some open problems related to local lexicons (Section 6.4) and conclude the chapter (Section 6.5).

## 6.1 Inferring Local Lexicons

As noted earlier, audiences’ local lexicons play a key role in how news authors write for those audiences. Toponyms are usually *underspecified* when they exist in the audience’s local lexicon (i.e., referring to simply “Paris”, rather than “Paris, Texas”), so simply using contextual clues in the article text will not suffice to recognize and resolve them. Using the local lexicon will improve geotagging on articles from a given news source. Therefore, given such a source, we must employ a method to infer the intended audience’s local lexicon, if one exists. For a newspaper, we might consider simply using the postal ad-

dress of the newspaper's headquarters to determine the local lexicon of its audience, as lists of such addresses are available commercially. However, the address alone gives no hint as to the geographic distribution of the newspaper's audience, and will therefore not help in determining whether a local lexicon exists for the newspaper. Also, for newspapers with multiple sections and multiple RSS feeds, it will not aid in determining which feeds are meant for a local audience. Also, even if we manually determined local lexicons for newspapers—NewsStand currently indexes over 10,000 newspapers—it would be infeasible to do the same for purely online newspapers, as well as user-oriented news sources such as the multiple millions of blogs and Twitter users. We therefore require an automated, scalable method for extracting local lexicons from online news sources, which includes not only online newspapers, but also the multiple millions of blogs and Twitter users.

To automatically infer local lexicons, we rely on three key characteristics of them:

1. *Stability*: A local lexicon is constant across articles from its news source.
2. *Proximity*: Toponyms in a local lexicon are geographically proximate.
3. *Modesty*: A local lexicon contains a considerable but not excessive number of toponyms.

The first property tells us that by observing and analyzing toponyms in a collection of articles from a news source, we should be able to determine the local lexicon as a common geographic theme among these articles. Note that this stability applies not only to local lexicons, but also to global lexicons as well. We must therefore use the second property of proximity to distinguish between local lexicons and more general spatial lexicons. In other words, a spatial lexicon can be classified as a local lexicon if and only if the toponyms within it are geographically proximate. The proximity property thus serves as a means of filtering and validation on an audience's local lexicon. The final modesty property highlights the notion that a person's local lexicon, while limited geographically, should at least contain several toponyms. In other words, it would be rare for a person to know of only one



or two local toponyms. We enforce the modesty property by specifying a minimum local lexicon size.

Note that we may infer local lexicons for a news source in a way analogous to geotagging a single article, but on a larger scale. In other words, we may perform collection-wide analogs of toponym recognition and resolution as a means of inferring local lexicons. We may first recognize a spatial lexicon by observing the toponyms in a collection of articles, and retaining those that are common to the collection. We then resolve the spatial lexicon in a geographic sense by classifying it as a local lexicon if the toponyms are geographically proximate, and if it has a reasonable size.

This analogy suggests a simple way to determine a collection’s local lexicon: we simply geotag each article in the collection, thereby collecting a set of resolved toponyms, select the most frequent toponyms in the collection, and check whether the toponyms are geographically proximate and reasonable in number. However, this presents a bootstrapping problem, in that determining a local lexicon relies on correct geotagging of individual articles in the collection, but correct geotagging relies on knowing the local lexicon.

To break this dependency cycle, we use a geotagging process termed *fuzzy geotagging* that does not fully resolve toponyms in a single article. Instead, this process determines a weighted set of possible resolutions for each toponym in an article. Note that we do not expect perfect geotagging accuracy from fuzzy geotagging, since we lack an established local lexicon, and we are therefore missing an important and necessary source of evidence for correct geotagging. Instead, because of the local lexicon’s stability property, we only require that fuzzy geotagging performs adequately across the entire set of articles  $A$  used to infer a local lexicon.

Fuzzy geotagging can best be understood as a variant of a traditional heuristic-based geotagging process. In such a traditional process, a toponym recognition system first finds the toponyms  $T$  in an article  $a$ . A gazetteer is then used to associate each  $t \in T$  with the set of all possible resolutions  $R_t$  for  $t$ . Next, the geotagging process uses toponym resolution

heuristics to filter resolutions from the set of  $R_t$ . Finally, for all  $t$  that are still ambiguous (i.e.,  $|R_t| > 1$ ), all but a single “default sense”  $r$  are filtered from  $R_t$ . This default sense is usually based on another heuristic, such as the resolution with largest population or largest geographic scope in terms of a geographic hierarchy. In this way, each recognized toponym  $t$  is resolved to a single pair of geographic coordinates. To accomplish fuzzy geotagging, for each recognized toponym, we do not assign the toponym with its final default sense at the end of the resolution process. Instead, for each  $t$  and  $r \in R_t$ , we assign a weight  $w_r$  to  $r$ , either uniformly or using default sense heuristics. Finally, we sum the weights for each  $r \in R_t$  across all articles in  $A$ .

An attractive feature of fuzzy geotagging is that even though it depends on an underlying geotagging process, it is mostly independent of the underlying geotagging implementation. Because of a geographic averaging effect across all articles in  $A$ , any geotagger can be used, as long as it performs reasonably. Furthermore, probabilistic geotagging methods (e.g., [73]) can be adapted for fuzzy geotagging by simply using each resolution’s assigned probability as its weight. Of course, a high quality underlying geotagger will result in better performance when inferring local lexicons. For fuzzy geotagging, we use our own toponym recognition methods (described in Chapter 4) and toponym resolution methods (Section 6.2), respectively.

One concern is that the geotagging process might assign a large weight to an incorrectly resolved toponym. For example, there exist over 60 locations named Paris in our gazetteer, but a naive underlying geotagger would assign any mention of Paris to the French capital, due to its relative prominence. For news sources with a different Paris in their local lexicons, this will be a consistent error in the fuzzy geotagging process. However, note that it is not likely for several distinct prominent locations to coexist within a single local lexicon, because larger places tend to be geographically distributed. Also, proximate geographic locations with the same name would cause confusion among local residents. For example, given a small geographic area, it is unlikely to find prominent places named Paris, Athens,

Dublin, and Alexandria, and even less likely to find multiple places named Paris. Therefore, errors of assigning large weights to incorrectly resolved toponyms will be committed infrequently for a single news source.

---

**Algorithm 6.1** Infer an intended audience’s local lexicon.

---

**input:** Set of articles  $A$ , maximum diameter  $D_{max}$ , minimum lexicon size  $S_{min}$   
**output:** Local lexicon  $L$ , or  $\emptyset$  if none

- 1: **procedure** INFERLOCALLEXICON( $A, D_{max}, S_{min}$ )
- 2:      $G \leftarrow \emptyset$
- 3:      $L \leftarrow \emptyset$
- 4:     **for all**  $a \in A$  **do**
- 5:          $G \leftarrow G \cup \text{FUZZYGEOTAG}(a)$
- 6:     **end for**
- 7:      $G \leftarrow \text{ORDERBYWEIGHT}(G)$
- 8:     **for**  $i \in \{1 \dots |G|\}$  **do**
- 9:          $H \leftarrow \text{CONVEXHULL}(L \cup G_i)$
- 10:         **if** DIAMETER( $H$ ) >  $D_{max}$  **then**
- 11:             **break**
- 12:         **end if**
- 13:          $L \leftarrow L \cup G_i$
- 14:     **end for**
- 15:     **if**  $|L| < S_{min}$  **then**
- 16:          $L \leftarrow \emptyset$
- 17:     **end if**
- 18:     **return**  $L$
- 19: **end procedure**

---

With the above in mind, we infer local lexicons using procedure INFERLOCALLEXICON, listed as Algorithm 6.1. The procedure takes as input a set of articles  $A$  from a single news source, as well as parameters  $D_{max}$ , used to determine the measure of geographic locality of an inferred spatial lexicon, and  $S_{min}$ , the minimum allowed size of a local lexicon. We determined appropriate values for these parameters experimentally (described in Section 6.3.3). We begin by initializing a set of resolved toponyms  $G$  and the eventual inferred local lexicon  $L$  to the empty sets (lines 2–3). Next, we loop over all articles  $a \in A$  (4–6), recognizing and resolving toponyms from each article in turn. We subject each article  $a$  to the aforementioned fuzzy geotagging process with Procedure FUZZYGEOTAG, which returns a set of toponyms found in  $a$ , and their potential interpretations and weights (5). We

aggregate these resolved and weighted toponyms into  $G$ , merging repeated interpretations and summing their weights. For example, if articles  $a_1 \dots a_k$  in the collection each contain a mention of “College Park”, and the fuzzy geotagging process assigned these toponyms to College Park, MD with weights  $w_1 \dots w_k$  (not necessarily equal), we merge these  $k$  interpretations in  $G$  to a single grounded toponym with weight  $\sum_i w_i$ .

At this point,  $G$  serves as a weighted spatial lexicon for the articles in  $A$ . We proceed to extract a local lexicon from the resolved toponyms  $r \in G$  by noting that the most heavily weighted  $r \in G$  are common to a large number of articles in  $A$ , and should be considered as part of a potential local lexicon. To this end, we first order the resolved toponyms in  $G$  by decreasing order of weight (7), and consider adding each toponym in turn to the local lexicon  $L$  (8–14). For each resolved toponym  $G_i$ , we determine the *convex hull*  $H$  of the geographic coordinates of the toponyms in  $L$  combined with the new toponym  $G_i$  (9). We then measure the diameter of  $H$ , and check whether it exceeds  $D_{max}$ ; if so, we cease adding toponyms to  $L$  (10–12). We do so to enforce the proximity property of local lexicons. Otherwise, we add  $G_i$  to  $L$  (13), and continue with  $G_{i+1}$ . After  $G$ ’s toponyms have been considered, we check whether the collected lexicon is larger than  $S_{min}$ , which qualifies it as a true local lexicon, and nullify  $L$  if it does not reach our minimum limit (15–17). Finally, we return  $L$ , which is the extracted local lexicon, or  $\emptyset$  if no local lexicon was found (18).

To illustrate this procedure, Figure 6.3 shows the local lexicon inferred by INFER-LOCALLEXICON for 137 articles from the Paris News, a small newspaper in Paris, Texas, which is approximately 100 miles northeast of Dallas, Texas. In the figure, Paris lies in the northeast quadrant of the inset. Each point represents a toponym found in an article published in the Paris News, with the color indicating its frequency across all articles in the collection. By far, the most frequently geotagged toponym was Paris (22 mentions). Other toponyms included Lamar County (13 mentions) and Dallas (5 mentions), in addition to a variety of toponyms unrelated to the local lexicon. Starting with Paris, the most frequently occurring toponym, we add toponyms to  $L$  in decreasing order of frequency until the di-



Figure 6.3: The local lexicon inferred for the Paris News, a small newspaper in Paris, Texas (upper right in inset), with  $D_{max} = 150$  miles and  $S_{min} = 5$ . The final convex hull (dashed red) has a diameter (solid red) of about 130 miles.

ameter of the convex hull of  $L > D_{max}$  (for this example,  $D_{max}$  was set to 150 miles). The final convex hull is shown in dashed red, with its diameter of approximately 130 miles highlighted in solid red. A final test ensures that  $|L| > S_{min}$  (for this example,  $S_{min}$  was chosen to be 5).

By considering resolved toponyms in order of decreasing weight, INFERLOCALLEXICON makes use of the stability property of local lexicons, since the most heavily weighted toponyms will have been resolved consistently across many articles in  $A$ . It also ensures that the returned local lexicon  $L$  falls within a geographic footprint with diameter smaller than  $D_{max}$ , thus enforcing the proximity property, and nullifies  $L$  if it violates the modesty property by having  $|L| < S_{min}$ . In our evaluation of INFERLOCALLEXICON (see Section 6.3.3), we determine suitable parameter values for  $D_{max}$  and  $S_{min}$  and find that the overall procedure performs well.

## 6.2 Resolution with Local Lexicons

Having established a method for determining local lexicons, we now apply these lexicons for toponym resolution. The main idea behind our geotagging process is to model how an

article author establishes a geographic framework within an article, to make it easier for human readers in the author's intended audience to recognize and resolve toponyms. Authors create this framework by using linguistic contextual clues that we detect using heuristic rules. Furthermore, readers are expected to read articles linearly, so article language has a contextual and geographic flow. Toponyms mentioned in a sentence will establish a geographic framework for subsequent text. To ensure correct geotagging, we therefore process the article text in a linear fashion.

Finally, and of greatest importance, an article author will keep in mind the nature of the expected audience's spatial lexicon, and in particular the local lexicon, to underspecify those toponyms in situations where adding geographic context would be redundant. For these underspecified toponyms, we only consider those possible interpretations that are known to intended readers, either due to relative prominence (such as countries and capital cities) or existence in their local lexicon, rather than all possible interpretations from the multiple millions of entries in our gazetteer, which is a much larger set of locations than any human could possibly know. If no resolution is found that satisfies our constraints, then we drop the toponym as a false positive (i.e., ignore it), rather than assuming the toponym recognition process was correct and hence assigning it a default sense (e.g., the most populous interpretation). This procedure restores the precision lost in our high recall toponym recognition process, described in Chapter 4, by retaining interpretations that have some internal or external supporting evidence. Thus, unlike many existing geotagging approaches, we view successful geotagging as a single integrated process, rather than as separate toponym recognition and resolution systems that are chained together.

After recognizing toponyms from an article to be geotagged (see Chapter 4), we proceed to resolve toponyms using a number of heuristic rules. Table 6.1 lists the set of heuristics used in our toponym resolution process, as well as examples of when each heuristic would be applied. These heuristics are inspired by how humans normally read news articles. We apply the heuristics in the order listed in Table 6.1. For toponyms that can be resolved by

Table 6.1: Heuristics used in our toponym resolution process.

---

$\mathcal{H}_1$	Dateline	Resolve dateline toponyms using: $\mathcal{H}_4, \mathcal{H}_5, \mathcal{H}_6$ . Resolve other toponyms proximate to resolved dateline. LONDON, Ont. - A police... Paris, TX (AP) - New...
$\mathcal{H}_2$	Relative Geog.	Resolve anchor toponym using: $\mathcal{H}_1, \mathcal{H}_4, \mathcal{H}_5, \mathcal{H}_6$ . Resolve toponyms proximate to defined geographic region. ...4 miles east of Athens, Texas. ...lives just outside of Lewistown...
$\mathcal{H}_3$	Comma Group	Resolve toponym group using: $\mathcal{H}_6, \mathcal{H}_5, \text{Geographic Proximity}$ . ...California, Texas and Pennsylvania.
$\mathcal{H}_4$	Obj/Container	Resolve toponym pairs with a containment relationship. ...priority in Jordan, Minn., ...
$\mathcal{H}_5$	Local Lexicon	Resolve toponyms proximate to local lexicon centroid. ( <i>Examples are news source dependent</i> )
$\mathcal{H}_6$	Global Lexicon	Resolve toponyms in a list of well known places. ...issues with China, knowing...
$\mathcal{H}_7$	One Sense	Resolve toponyms sharing names with already-resolved ones. ( <i>Examples are article dependent</i> )

---

multiple heuristics, we use the resolution suggested by the highest ranked heuristic. Our highest-ranked heuristics establish a geographic context for large portions of the article, i.e., Dateline ( $\mathcal{H}_1$ ) and Relative Geography ( $\mathcal{H}_2$ ). We continue with heuristics favoring contextual language clues, namely Comma Group ( $\mathcal{H}_3$ ) and Object/Container ( $\mathcal{H}_4$ ). Finally, we conclude with default sense heuristics using the reader’s Local Lexicon ( $\mathcal{H}_5$ ) and Global Lexicon ( $\mathcal{H}_6$ ). In addition, we use a One Sense ( $\mathcal{H}_7$ ) heuristic modeled after the one sense per discourse assumption [164] that all instances of a repeated toponym will have the same resolution. We apply  $\mathcal{H}_7$  after each of  $\mathcal{H}_1$ – $\mathcal{H}_6$ , which propagates a toponym resolution to all later repeated mentions of the toponym. This heuristic enforces a consistent resolution of the same toponym in the same article. Note that despite the Local Lexicon heuristic’s low ranking as  $\mathcal{H}_5$ , several other heuristics, namely  $\mathcal{H}_1$ – $\mathcal{H}_3$ , appeal to the Local Lexicon heuristic for correct resolution. The local lexicon thus plays a large role in our toponym resolution procedure. In our evaluation, we measure how often each heuristic was used in geotagging our evaluation corpora (see Section 6.3.5).

For the sake of clarity, we now provide more detailed descriptions of heuristics  $\mathcal{H}_1$ – $\mathcal{H}_6$ , and give examples of each.

### 6.2.1 $\mathcal{H}_1$ : Dateline

We examine the article, checking for the presence of dateline toponyms, which if present appear at the article’s beginning and establish the general geographic locality of the events described in the article. If happening in a place unfamiliar to the author’s audience, then authors generally use object/container clues (e.g., “LONDON, Ont. —”). Otherwise the location will be underspecified (e.g., “LONDON —”), since it already exists in the audience’s spatial lexicon. Therefore, we attempt to resolve dateline toponyms using the Object/Container ( $\mathcal{H}_4$ ), Local Lexicon ( $\mathcal{H}_5$ ), and finally Global Lexicon ( $\mathcal{H}_6$ ) heuristics. If we can successfully resolve dateline toponyms, then we resolve more toponyms from the article that are geographically proximate to the resolved dateline toponyms.



### 6.2.2 $\mathcal{H}_2$ : Relative Geography

Certain phrases in article text denote relative geography, which is language that defines a usually imprecise geographic region in terms of distance from or proximity to another geographic location. These imprecise regions are important because they usually target the geographic areas where the events in an article took place, and therefore are useful for resolving the article’s toponyms. Example instances of relative geography include “4 miles east of Athens, Texas” and “just outside of Lewistown”. We refer to the toponyms in such phrases as *anchor toponyms*, and we term the resulting regions as *target regions*.

Notice that anchor toponyms follow the same specification patterns as those used for dateline toponyms. Therefore, to resolve target regions, we first resolve the anchor toponyms, using the same heuristics as used for the Dateline ( $\mathcal{H}_1$ ) heuristic, namely Object/Container ( $\mathcal{H}_4$ ), Local Lexicon ( $\mathcal{H}_5$ ), and Global Lexicon ( $\mathcal{H}_6$ ). After resolving the anchor toponym, we set the target region in terms of proximity to the anchor toponym (as in “just outside of Lewistown”) or proximity to a geographic point defined relative to the anchor toponym (as in “4 miles east of Athens, Texas”). Finally, we resolve all toponyms in the article that are geographically proximate to the target region.

### 6.2.3 $\mathcal{H}_3$ : Comma Group

Recalling the discussion in Chapter 5, lists of toponyms in articles are a frequent occurrence in news articles, and we refer to these lists as *comma groups*. Authors generally organize toponyms into concise groups when they share a common characteristic, such as all being prominent places (e.g., “California, Texas and Pennsylvania”, all states in the USA) or all being mutually geographically proximate (e.g., “College Park, Greenbelt and Bladensburg”, all small places near College Park, MD). We resolve all toponyms in comma groups by applying a heuristic uniformly across the entire group. First, we check whether all toponyms exist in the Global Lexicon ( $\mathcal{H}_6$ ) or the Local Lexicon ( $\mathcal{H}_5$ ). We also check whether interpretations exist that are all constrained to a small geographic area, not necessarily the

same as the local lexicon region. Finding and using comma groups is described in detail in Chapter 5.

#### 6.2.4 $\mathcal{H}_4$ : Object/Container

Authors commonly provide contextual evidence for a toponym by specifying its containing toponym, in terms of a geographic hierarchy. For example, an author might mention “College Park, Maryland”, which indicates that the correct instance of College Park lies within its container toponym, Maryland. They may also use abbreviations for the container, such as “Jordan, Minn.” (referring to Minnesota). To resolve these toponyms, we appeal to our gazetteer and choose a pair of interpretations that satisfies the hierarchy constraint.

#### 6.2.5 $\mathcal{H}_5$ : Local Lexicon

If we inferred a local lexicon for the article’s news source (see Section 6.1), we now use the local lexicon to resolve article toponyms. We first compute the geographic centroid of the source’s inferred local lexicon, which has meaning because of the proximity property of toponyms in the local lexicon. We then resolve those toponyms that are geographically proximate to the centroid. If the news source has no local lexicon, as would occur for a newspaper with a widely dispersed audience, then we do not apply this heuristic.

#### 6.2.6 $\mathcal{H}_6$ : Global Lexicon

Our final heuristic uses a curated global lexicon of toponyms which we regard as prominent enough to be known by audiences regardless of their geographic location. We created an initial global lexicon by adding prominent geopolitical divisions such as continents and country names, as well as large regions and cities with over 100,000 population. Note that population is a coarse measure and finally serves as a substitute for “prominence”, but works adequately for our purposes.

### 6.2.7 Postprocessing

We perform several postprocessing measures after the geotagging process. Once we have settled on a set of resolved toponyms for the article, we next determine the article’s geographic focus and scope by selecting a subset of prominent resolved toponyms from the article. One simple measure to use is the frequency of each distinct toponym’s occurrence within the article. We modify this frequency measure to better account for the *inverted pyramid* [131] structure of news articles. Each resolved toponym occurrence is assigned a weight based on its distance from the beginning of the article, with weights linearly decreasing with distance from the beginning. This scheme will assign the highest weights to toponyms in the dateline and first few sentences, and lower weights to tangential toponyms mentioned later in the article. We then aggregate weights for distinct toponyms, and rank them using the resulting weight sums.

In addition, we draw on our online clustering algorithm (detailed in Section 3.2.4) to mitigate potential geotagging errors. If the article was placed in a cluster with other geotagged articles, we take advantage of a geographic averaging effect, similar to that used for establishing local lexicons. Each article in the cluster will be composed slightly differently by different authors, some of whom may provide additional contextual evidence in an article that our geotagger can use for the entire cluster. These slight differences between articles about the same news story provide multiple, somewhat independent trials for our geotagger and can be used to correct geotagger errors. Consider a cluster containing two articles mentioning “College Park”, which is underspecified as “College Park” in the first article, but is more fully specified as “College Park, MD” in the second. If we found that geotagging resulted in a different sense of College Park in the first article, we would correct it based on the better-specified mention in the second.

Also, if we have determined that the article’s focus and scope are limited to a small geographic area, we perform an additional *hyperlocal* geotagging process, where we examine the set of recognized toponyms and consider very small geographic features that ordinar-

ily would be unknown except to people living in the vicinity. These features include spot features such as schools, churches, and other buildings, hydrographic features such as local rivers and lakes, and other regions such as parks.

## 6.3 Experiments

In this section, we describe the results of experiments performed to determine the efficacy of our local lexicon inference, toponym recognition, and toponym resolution methods, as measured over two datasets of hand-geotagged news articles. After describing the evaluation measures used to report performance (Section 6.3.1), and our test datasets (Section 6.3.2), we show experimental results for local lexicon inference, toponym recognition, and toponym resolution (Sections 6.3.3–6.3.5). We also measure how often our heuristics were used to resolve toponyms, to determine their relative importance (Section 6.3.6).

### 6.3.1 Evaluation Measures

To evaluate the performance of a toponym recognition system on a given document, we must decide what constitutes a match between a system-generated toponym and a ground truth toponym from the document. Henceforth, in the context of toponym recognition, we consider toponyms to match if their constituent words match exactly, even if their positions in the document are different. To measure performance, we use the well-known measures *precision* and *recall*, which for a set of ground truth toponyms  $G$  and a set of system-generated toponyms  $S$ , are defined as

$$P(G, S) = \frac{|G \cap S|}{|S|}, R(G, S) = \frac{|G \cap S|}{|G|}.$$

We also make use of the  $F_1$ -score, defined as the harmonic mean of precision and recall:

$$F_1 = \frac{2PR}{P + R}.$$

These measures are described in more detail in Section 4.3.4.

### 6.3.2 Datasets

We used two datasets of news articles in our evaluation. The first is a subset of the ACE 2005 English SpatialML Annotations [87], available from the Linguistic Data Consortium, which we refer to as *ACE*. *ACE* contains 428 documents in total that represent a variety of spatially-informed data sources, including news wire and blog text, as well as online newsgroups and transcripts of broadcast news. Each document is annotated using SpatialML [88], an XML-based language which allows the recording of toponyms and their geographically-relevant attributes, such as their lat/long position, feature type, and corresponding entry in a gazetteer. For this evaluation, we limited our test collection to news stories, resulting in 104 news articles from prominent newspapers and news wire sources.

Unfortunately, since news wire is usually written and edited for a broadly distributed geographic audience, the *ACE* corpus is quite limited for the purposes of evaluating local lexicons' impact on geotagging, and is hardly representative of data from smaller newspapers with a more localized audience, which have a large presence on the Web. As a result, we created our own corpus of news articles by sampling from the collection of over 4 million articles indexed by the NewsStand system [143], which we call the *Local-Global Lexicon* corpus, or simply *LGL*. We focused on articles from a variety of smaller, geographically-distributed newspapers. To find this set of smaller newspapers and thereby ensure a more challenging toponym resolution process, we first ranked toponyms in our gazetteer by ambiguity, and selected highly ambiguous toponyms such as Paris and London. We then selected newspapers based near these ambiguous toponyms. For example, some US-based newspapers located near a Paris include the Paris News (Texas), the Paris Post-Intelligencer (Tennessee), and the Paris Beacon-News (Illinois). For each newspaper, we chose several articles to include in *LGL*, and manually annotated the toponyms in these articles, including the corresponding entries from our gazetteer.

Table 6.2: Comparison of the ACE and LGL corpora.

	ACE	LGL
Number of data sources	4	78
Number of articles	104	588
Number of tokens	48,036	213,446
Number of toponyms	5,813	4,793
Distinct toponyms	915	1,297
Prevalent Toponym Types		
Countries	1,685	961
Administrative divisions	255	1,322
Capital cities	454	318
Populated places	178	1,968

Table 6.2 summarizes statistics for the ACE and LGL corpora. These statistics show the limitations of ACE in terms of source breadth, as only four news sources are represented in the corpus—Agence France-Presse (AFP), Associated Press World, New York Times, and Xinhua—with 42, 40, 5, and 17 annotated articles from each source, respectively. On the other hand, LGL contains 588 articles from 78 newspapers, with an average of 5 articles per newspaper. Also, as the toponym counts show, the articles in ACE tend to be more toponym-heavy, with over 50 toponyms per article, in contrast to LGL articles with an average of 8 toponyms per article. Examining the prevalent toponym types in both data sources reveals that the ACE collection is also very international in scope, with 1,685 of 5,813 toponyms (29%) corresponding to country names. In contrast, LGL’s set of toponyms are more local. Out of 4,793 total toponyms, 1,968 (41%) are smaller populated places, and 1,322 (28%) are administrative divisions such as states and counties. These statistics show that the ACE corpus is better suited for evaluating geotagging on an international scope, while LGL is better-suited for testing geotagging on a local level.

### 6.3.3 Inferring Local Lexicons

We tested our local lexicon inference procedure with fuzzy geotagging, described in Section 6.1. The idea behind our evaluation procedure is that for a small, local newspaper, the newspaper’s audience will be geographically proximate to the newspaper’s geographic focus. As a result, the local lexicon of the newspaper’s audience will consist of multiple places near the newspaper’s geographic focus. We measure how successful we are in establishing a given newspaper’s audience’s local lexicon by checking whether the centroid of our inferred local lexicon is within a given distance  $\delta$  to a ground truth annotation of the newspaper’s geographic focus. For larger newspapers annotated with no geographic focus and hence no assumed local lexicon, we check that our local lexicon inference procedure also returned no local lexicon. In other words, we test our inference procedure first in terms of binary classification (“has local lexicon” or “no local lexicon”) and second in terms of geographic distance from the ground truth focus. As earlier, we use precision and recall to measure performance, with the ground truth foci and the local lexicon foci returned by our inference procedure serving as the ground truth and system-generated sets, respectively.

Table 6.3 summarizes the situations in which we consider the ground truth  $N_G$  to match our system-generated local lexicon  $N_S$  for a given news source  $N$ . We consider the results to match if both  $N_G$  and  $N_S$  do not exist (i.e., the ground truth had no geographic focus for news source  $N$  and our inference procedure did not return a local lexicon). Also, the results match if both  $N_G$  and  $N_S$  exist, and further, that the distance between the geographic

Table 6.3: For a given news source  $N$ , situations in which we consider our local lexicon inference procedure’s focus  $N_S$  to match the ground truth focus  $N_G$ .

$N_G$ exists	$N_S$ exists	$D(N_G, N_S) \leq \delta$	Match
No	No	—	Yes
Yes	No	—	No
No	Yes	—	No
Yes	Yes	No	No
Yes	Yes	Yes	Yes

centroids of  $N_G$  and  $N_S$  is less than a distance threshold  $\delta$ . Otherwise, we consider the results to differ, and penalize precision and recall as appropriate.

To establish our ground truth, we examined each of the 4,867 active newspaper sources in the NewsStand system, and manually annotated news sources with geographic foci as appropriate. Figure 6.4 shows the mapped geographic foci of news sources in the USA, highlighting those with articles in LGL. To create a collection of news stories to use for determining local lexicons, we gathered approximately two months' worth of news stories in February and March 2009, resulting in a total of 1,266,119 articles. From this large collection of articles, 7,654 were from the 78 news sources in LGL. However, the distribution of articles in sources is highly skewed, with over half of 78 news sources having under 50 articles total. For each news source (regardless of whether it is in LGL) we then tried to detect a valid local lexicon using INFERLOCALLEXICON (described in Section 6.1).

For each such local lexicon found, we selected the geographic centroid of all locations in the lexicon as its geographic focus. For a given news source, our inference procedure was deemed to match the ground truth in one of two cases:

1. The news source had a large geographic scope (and therefore no local lexicon) and no local lexicon was found by our procedure;
2. The news source had a local geographic scope (and consequently a local lexicon), and a local lexicon was found by our procedure, and the geographic distance between our found local lexicon and the ground truth was less than a distance threshold  $\delta$ .

The goal for our first test is to evaluate the efficacy of INFERLOCALLEXICON in terms of determining how far, measured by  $\delta$  in the ranges  $[0,50)$ ,  $[50,100)$ , and  $[100,\infty)$ , it placed the geographic focus of a source's local lexicon from its ground truth value, while also varying the maximum diameter  $D_{max}$  of the convex hull of the locations in the lexicon found between 50 and 1,000 miles in 50 mile increments. Note that setting  $\delta = \infty$  effectively results in a test for local lexicon inference without regard to its distance from



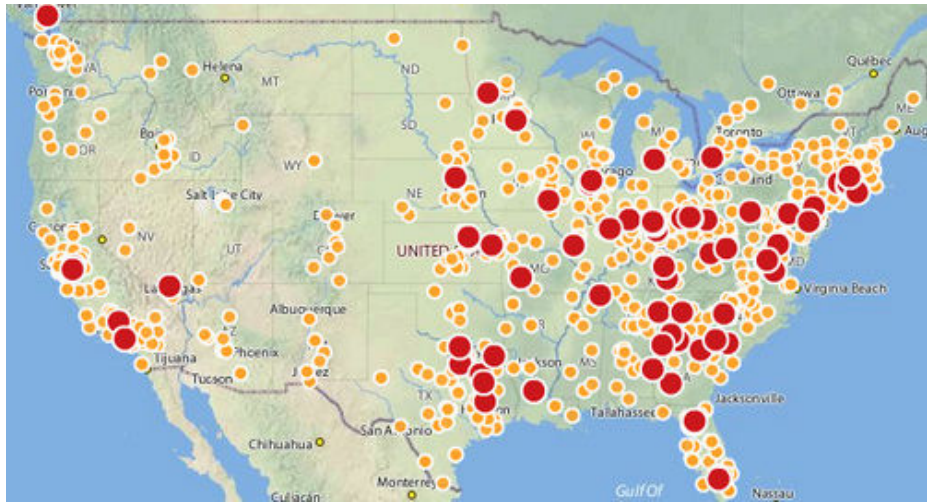


Figure 6.4: Local lexicon foci for news sources in the USA (LGL in red).

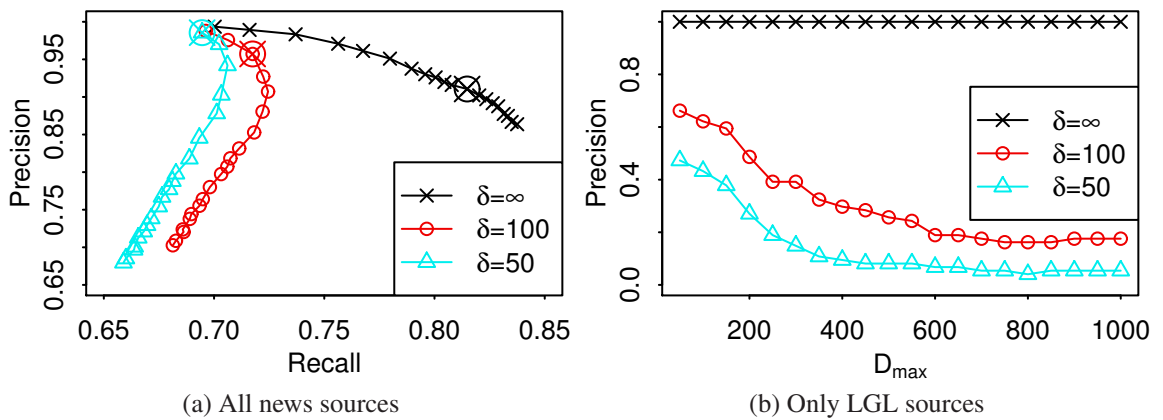


Figure 6.5: Performance on local lexicon inference when varying maximum hull diameter  $D_{max}$  between 50–1,000 miles and keeping minimum lexicon size  $S_{min}$  fixed at (a) 5 and (b) 3.

the true geographic focus, i.e., whether the corresponding source was classified correctly as having a local lexicon or not. For this test, we kept the minimum lexicon size,  $S_{min}$ , fixed at 5 for our test on all feeds and 3 for our test on only LGL feeds. Figure 6.5 shows our performance results for inferring local lexicons on all news sources and only those sources in LGL, respectively. In Figure 6.5a, the point where all three lines coincide corresponds to the minimum value of  $D_{max}$  tested, namely 50 miles. Each successive plotted point corresponds to a 50 mile increase in  $D_{max}$ . Also, the points with maximum  $F_1$ -score are highlighted. Observe that a smaller value of  $D_{max}$  results in higher precision at the expense of recall, corresponding to the high precision points in the left portion of each plot. For all sources and  $D_{max}$  values tested, we obtained precision between 0.65–1.00, and recall between about 0.65–0.85. The performance results indicate that our local lexicon inference procedure tends to have high precision overall, with values well above 0.90 for both sets for relatively small  $D_{max} < 200$  miles. Above this value, precision suffers, but with little corresponding gain in recall. In fact, for  $D_{max} > 150$  ( $\delta = 50$ ) and  $D_{max} > 250$  ( $\delta = 100$ ), both precision and recall decrease, because any gains in recall from detecting a local lexicon are more than offset by penalties from having local lexicon centroids too distant from the ground truth. Figure 6.5b shows a plot of  $D_{max}$  values and the corresponding precision of our inference process for news sources in LGL. Recall was omitted because all sources in LGL were marked with a geographic focus in the ground truth, and furthermore, our local lexicon inference procedure always found a local lexicon for all sources as well. As a result, recall always equaled precision in this test. We found that the minimum value of  $D_{max}$  tested, 50 miles, resulted in the best precision of 0.46 for  $\delta = 50$  and 0.66 for  $\delta = 100$ . In general, our inference algorithm performed well when simply detecting the presence of a local lexicon, as evidenced by the high precision and recall values of the  $\delta = \infty$  curves in both figures. Also, performance was better across all news sources than for those only in LGL, mainly due to the relative scarcity and highly skewed distribution of stories produced by the LGL news sources. The tests indicate that a value of  $D_{max} = 200$  miles is

reasonable for inferring local lexicons with INFERLOCALLEXICON.

For our second test of INFERLOCALLEXICON, we varied the minimum acceptable local lexicon size,  $S_{min}$ , between 4–15, while keeping the maximum convex hull diameter  $D_{max}$  fixed at 200 for all feeds and 150 for LGL feeds, to discover a suitable value for our use. Figure 6.6 shows the results of our tests. The smallest value of  $S_{min}$  tested (4) corresponds to the rightmost points of each curve in the graphs, which demonstrates that small  $S_{min}$  results in higher recall at the expense of precision. For all feeds (Figure 6.6a), a small increase in  $S_{min}$  to the range of values 5–7 results in a large jump in precision with little drop in recall, and for larger values of  $S_{min}$ , the inference procedure quickly converges to near 1.00 precision and about 0.70 recall. Similar results can be seen for only LGL sources (Figure 6.6b), where the points with smallest value of  $S_{min}$  (4) have the highest recall and also highest  $F_1$ -scores. When increasing  $S_{min}$ , precision rapidly jumps, but at the heavy expense of recall, since a larger  $S_{min}$  will result in fewer local lexicons being found. Again, we attribute the relatively low recall numbers for LGL sources to the skewed distribution of news articles in LGL. In general, both plots show that INFERLOCALLEXICON is a high precision procedure, so small values of  $S_{min}$  such as 5 are best. This relatively small number makes sense when considering that it is rare for even a few toponyms to occur frequently in articles from a given news source to also be geographically proximate, unless the news source’s geographic focus is in the area.

### 6.3.4 Toponym Recognition

Next, we evaluated our multifaceted toponym recognition procedure described in Chapter 4 (denoted here as the “hybrid” method) against a simpler method using only the Stanford named-entity recognizer, trained on a variety of news corpora. For the named-entity recognizer, we varied a threshold parameter that controls the minimum confidence level of output by the recognizer. Varying this parameter allowed control over whether precision or recall was to be favored. As is typical for statistical named-entity recognizers, setting a

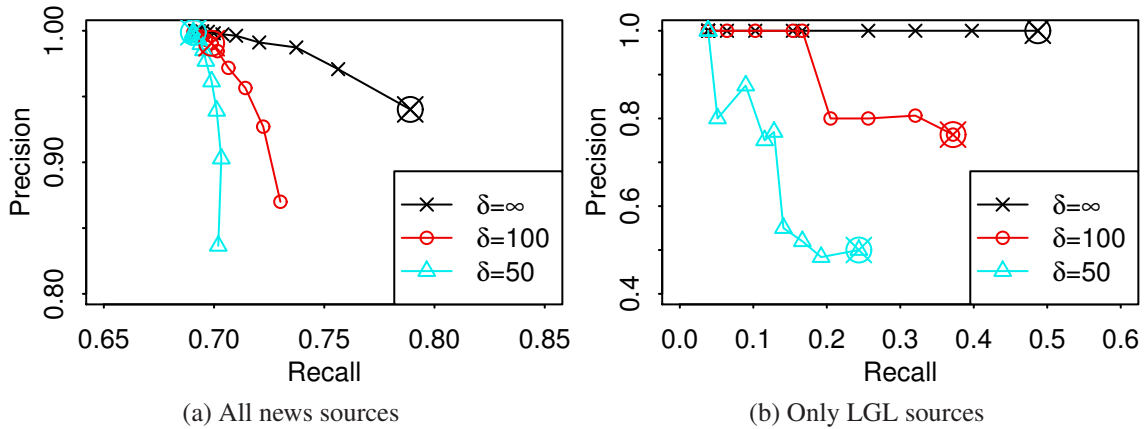


Figure 6.6: Performance on local lexicon inference by varying minimum lexicon size  $S_{min}$  between 4–15 and keeping maximum hull diameter  $D_{max}$  fixed at (a) 200 and (b) 150.

high value for the threshold parameter favors precision at the expense of recall, while a low threshold value favors recall at the expense of precision. Also, we considered toponyms to match only if they coincided exactly in the text; partial or otherwise overlapping toponyms were considered errors.

Figure 6.7 details performance results of the hybrid (H) and named-entity (NE) recognition procedures in the form of precision-recall diagrams. We also tested variants of the hybrid and named-entity procedures where the system-generated toponyms were filtered to only those toponyms that have entries in our gazetteer, labeled as  $H_G$  and  $NE_G$ , respectively. In other words, rather than blindly using the set of toponyms returned by the recognition process as our system-generated set, we remove those toponyms that are not present in the gazetteer. We do so to test our toponym recognition procedure as a standalone process, separate from its purpose as the first stage in a combined geotagging process. This is necessary in order to take into account the possibility that some of the toponyms are not in the gazetteer, which can happen in the case of names of regions that do not have specific boundaries on account of not being formal entities such as “New England” and “Upper West Side”. In this way we avoid penalizing the precision for these gazetteer failures. Note that our hybrid recognition procedure (H and  $H_G$ ) does not have an explicit tuning parameter to adjust the precision/recall tradeoff, and so Figure 6.7 contains a single data point for each.

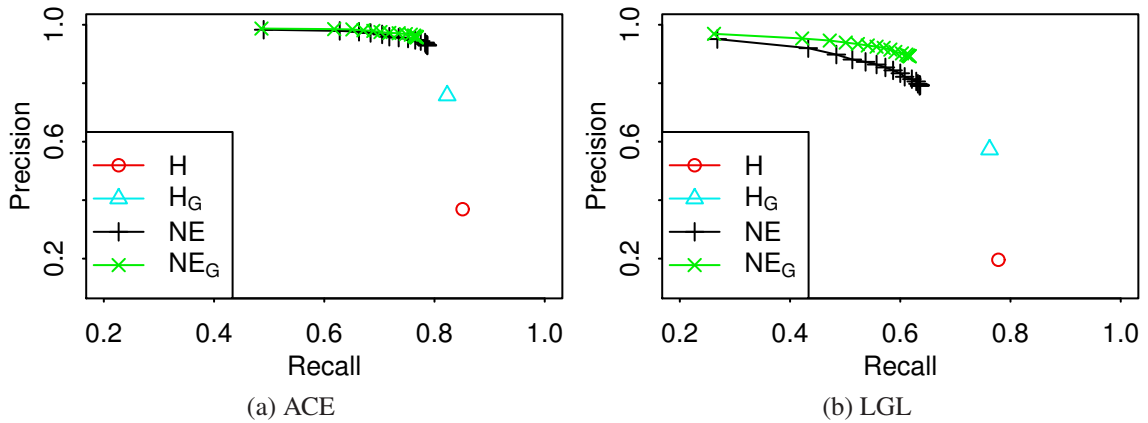


Figure 6.7: Precision-recall diagrams for toponym recognition performance using both our hybrid procedure (H) and a statistical named-entity recognizer (NE).

Figure 6.7a shows recognition results on the ACE corpus. As can be seen, the named-entity recognizer is highly tuned for precision. At all values of the threshold parameter, recognition precision was above 0.920, with the corresponding recall ranging between 0.490–0.787. Notice that gazetteer filtering did not have much of an effect on the NE method. This is not surprising, because in essence the training set which it uses plays a similar role to a gazetteer, but is very limited in scope. This limitation serves to ensure high precision but at the expense of recall as we have observed. In contrast, our hybrid recognition procedure emphasized toponym recall, with 0.369 precision and 0.851 recall before and 0.758 precision and 0.823 recall after gazetteer filtering. Though the precision greatly increases by almost 0.39, recall drops slightly by about 0.03, due to fewer toponyms being recognized because they do not exist in our gazetteer. This drop reflects the fact that most gazetteers are still rather incomplete or at least not in sync with the frequency of use of location descriptions that do not have formally defined boundaries.

Figure 6.7b shows similar results from our LGL corpus, although the difficulty of recognizing smaller, less prominent toponyms is reflected in both methods’ decreased performance relative to the ACE corpus. The named-entity recognizer again garnered high precision, varying between 0.892–0.969, and had recall between 0.262–0.617. In contrast, our hybrid recognition method, in combination with gazetteer filtering, resulted in 0.762 recall

and 0.573 precision. These performance numbers indicate clearly that our hybrid procedure is much better-suited for toponym recall than purely statistical recognition methods.

### 6.3.5 Toponym Resolution

Our measure of correctness for toponym resolution is the same as that in our toponym recognition evaluation, except that in addition to an exact toponym match, for a grounded toponym to be considered correct, it must have been placed a maximum of 10 miles from the ground truth toponym. This small distance range is required to account for small lat/long differences in the gazetteers used in annotating our evaluation corpora (IGDB [92] for ACE, GeoNames [43] for LGL). Exceptions to this rule include features with extent, such as countries and states. We measured precision, recall, and  $F_1$  of the entire toponym recognition and resolution process. In addition, we took two sets of measurements using different toponym recognition procedures to evaluate different stages of our geotagging process. Our first set of measurements used our own toponym recognition procedure, described in Chapter 4, which resulted in measurements of our entire geotagging procedure’s accuracy. For our second set of measurements, we assumed a *toponym oracle* for toponym recognition that ensures perfect toponym recognition by using the annotated ground truth toponyms, which effectively tests our toponym resolution procedure in isolation.

We also took two sets of measurements that deal with repeated toponyms in a single article differently, to reflect the different needs of different geographic information retrieval applications. Our first set of measurements considered all toponyms in an article separately, regardless of whether they had been repeated (labeled “All toponyms” in our performance tables), while our second set of measurements merged repeated instances of the same toponym in both ground truth and system-generated toponym sets, resulting in only distinct toponyms in the final toponym sets (labeled “Distinct toponyms”). To help illustrate these measurement strategies, consider an article containing three mentions of “Paris”, and the correct sense of all three mentions was Paris, Texas. In other words, the ground truth reso-

lution set for this article would be  $G = \{\text{Texas}, \text{Texas}, \text{Texas}\}$ . Furthermore, assume that a toponym resolution procedure assigned two of these mentions to Paris, Texas, and the third to Paris, France, resulting in a system-generated resolution set  $S = \{\text{Texas}, \text{Texas}, \text{France}\}$ . Using the first measurement, all repeated instances would be considered separately, resulting in precision  $P = \frac{2}{3}$  and recall  $R = \frac{2}{3}$ . Under the second measurement, repeated interpretations in  $G$  and  $S$  would be merged to form  $G' = \{\text{Texas}\}$  and  $S' = \{\text{Texas}, \text{France}\}$ , yielding precision  $P = \frac{1}{2}$  and recall  $R = 1$ . More concisely, the first measurement strategy treats  $G$  and  $S$  as multisets, while the second treats them as normal sets.

We compared our own geotagging procedure, referred to as IGeo, with implementations inspired by other noted geotagging methods. In particular, we created implementations using MetaCarta’s [121] confidence-based toponym resolution, Web-a-Where’s [7] gazetteer hierarchy resolution procedure, and the class-based weight heuristics of Volz et al. [154], henceforth referred to as MC, WaW, and VKM, respectively. In cases where the authors’ implementations were loosely specified, we used defaults that ensured reasonable performance. Also, despite our best efforts, we were unable to obtain the annotated corpora used by these previous researchers, either due to inactivity or restrictive copyright policies, and hence could not directly validate our implementations.

Table 6.4 details toponym resolution performance across both the ACE and LGL corpora. Maximum values for each evaluation method and corpus are highlighted in the table. For LGL, we also tested our IGeo procedure without using local lexicon evidence, listed as IGeo<sub>NL</sub>. Using our own toponym recognition, IGeo outperformed the other implementations across both corpora, in terms of precision, recall, and F<sub>1</sub>-score. For the ACE corpus, all the geotagging methods performed reasonably, with precision and recall values generally above 0.70. WaW most closely approached IGeo’s precision and recall, with nearly identical values. These performance numbers reflect the relative ease of geotagging news wire text, since toponyms are usually prominent places or well-specified with geographic contextual clues. However, examining performance in the LGL corpus, we see significant

Table 6.4: Toponym resolution performance results.

	Toponym recognition			Toponym oracle		
	<i>P</i>	<i>R</i>	<i>F</i> <sub>1</sub>	<i>P</i>	<i>R</i>	<i>F</i> <sub>1</sub>
ACE						
IGeo	<b>0.800</b>	<b>0.774</b>	<b>0.787</b>	<b>0.968</b>	0.890	<b>0.928</b>
WaW	0.795	0.773	0.784	0.962	<b>0.891</b>	0.925
MC	0.731	0.752	0.741	0.945	0.870	0.906
VKM	0.603	0.709	0.652	0.859	0.816	0.837
LGL						
IGeo	<b>0.826</b>	<b>0.654</b>	<b>0.730</b>	<b>0.964</b>	<b>0.817</b>	<b>0.885</b>
IGeo <sub>NL</sub>	0.698	0.450	0.548	0.788	0.546	0.645
WaW	0.651	0.452	0.534	0.761	0.628	0.689
MC	0.477	0.494	0.485	0.712	0.629	0.668
VKM	0.351	0.475	0.404	0.590	0.567	0.578

performance penalties for competing methods that neglect the local lexicon. IGeo outperforms its nearest competitor WaW by almost 0.20 in terms of precision, recall, and  $F_1$ -score. Notice that adding the local lexicon caused a large increase of about 0.13 precision and 0.20 recall over IGeo<sub>NL</sub>. This increase in both precision and recall stands in contrast to many information retrieval techniques, which usually increase either precision or recall at the expense of the other.

With a toponym oracle, performance results for all resolution methods are much higher, with  $F_1$ -scores for ACE approaching or exceeding 0.90. In particular, WaW’s performance nearly matched that of IGeo, in some cases being slightly better. However, when moving to the more difficult LGL, we again see a large performance difference of about 0.20 in terms of  $F_1$ -score between IGeo and competing methods. Again, for LGL, both precision and recall improve as a result of using local lexicon evidence. Interestingly, we see that IGeo’s precision for both the ACE and LGL corpora stays constant at approximately 0.96, which indicates that local lexicons serve as a high precision source of evidence for geotagging. Comparing performance results between our own toponym recognition procedure and the toponym oracle, IGeo’s performance gain using the toponym oracle in terms of  $F_1$ -score was about 0.10 for ACE, and about 0.15 for LGL. This difference reflects the greater dif-



ficuity in toponym recognition and resolution of the smaller, less prominent toponyms in LGL, which affects the performance of the non-IGeo methods. Furthermore, IGeo’s performance difference in terms of  $F_1$ -score between using toponym recognition and the toponym oracle was the least of all resolution methods (excepting IGeo<sub>NL</sub>), being 0.141 for ACE and 0.155 for LGL. This finding indicates that of all resolution methods, IGeo depended the least on using the toponym oracle for toponym recognition, which is artificial.

We also measured toponym resolution performance in terms of  $F_\beta$ , for values of  $\beta$  between 0.25–2.0, presented in Figure 6.8. The left portion of each figure corresponds to values of  $\beta$  favoring precision, while the right portion shows values favoring recall. As in Table 6.4, IGeo outperforms competitors across all values of  $\beta$ , but especially so for the LGL corpus.

### 6.3.6 Heuristic Usage

Our final experiment measured how much each heuristic listed in Table 6.1 played a part in geotagging precision across our evaluation corpora. Figure 6.9 shows our usage results. In the figures, each column represents a different heuristic, labeled  $H_1$ – $H_6$ , and corresponding to  $\mathcal{H}_1$ – $\mathcal{H}_6$  in Table 6.1. The One Sense heuristic ( $\mathcal{H}_7$ ) is not shown as it was applied after each of  $\mathcal{H}_1$ – $\mathcal{H}_6$ . Toponyms resolved using  $\mathcal{H}_7$  were counted toward whichever of  $\mathcal{H}_1$ – $\mathcal{H}_6$  was responsible for the propagated resolution. Figure 6.9a shows the usage distribution for ACE. In each column, the first bar (+) shows how often the heuristic contributed to a correctly resolved toponym, while the second bar (–) counts instances where the heuristic led to an error. Examining Figure 6.9a reveals that the most important heuristics for toponym resolution were Global Lexicon ( $\mathcal{H}_6$ ) and Dateline ( $\mathcal{H}_1$ ). This is not overly surprising, as ACE consists mostly of news wire of international scope, so most toponyms mentioned in ACE articles will be prominent places. Also, being news wire, the Local Lexicon ( $\mathcal{H}_5$ ) played no role in toponym resolution.

Figure 6.9b shows heuristic usage in the LGL corpus. In each column, the first two

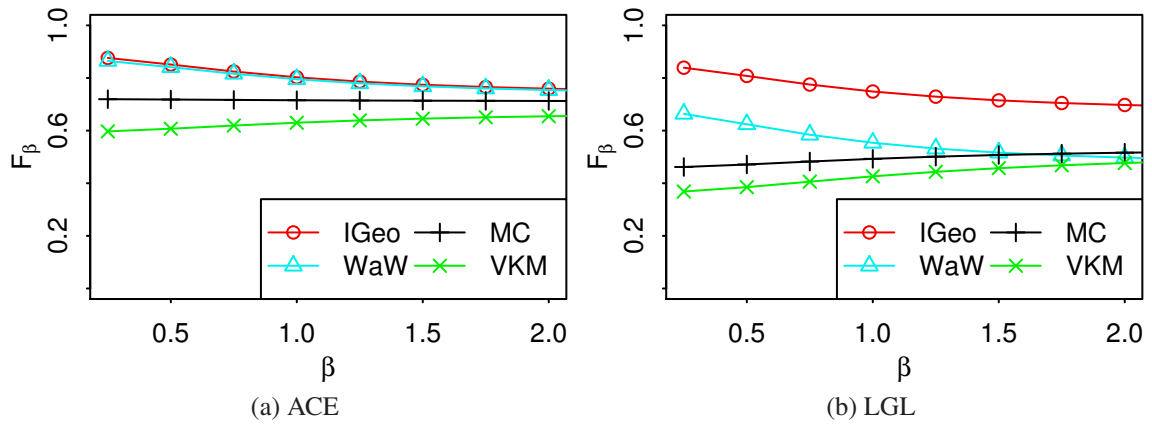


Figure 6.8: Toponym resolution performance measured with  $F_\beta$  for  $0.25 \leq \beta \leq 2.0$ , in 0.25 increments.

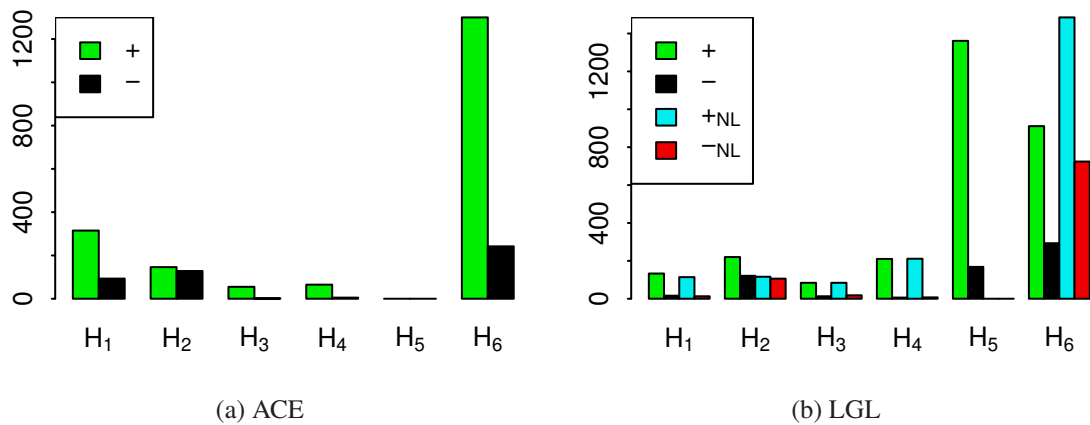


Figure 6.9: Heuristic usage in toponym resolution for both ACE and LGL corpora.

bars are successes (+) and errors (−) as before, while the second two show successes (+<sub>NL</sub>) and errors (−<sub>NL</sub>) when disregarding local lexicon evidence. It is clear that the Local Lexicon ( $\mathcal{H}_5$ ) plays a large role in correct toponym resolution, and suffers from relatively few errors. This result is in keeping with our earlier observation that using the local lexicon affords high precision. Interestingly, the Global Lexicon ( $\mathcal{H}_6$ ) plays an even larger role in correct toponym resolution when used without a Local Lexicon ( $\mathcal{H}_5$ ) but it also causes even more errors, but the relative difference is surprisingly the same as when used with a Local Lexicon ( $\mathcal{H}_5$ ). Also, the Dateline heuristic ( $\mathcal{H}_1$ ) has less use in LGL than in ACE, which reflects the lack of dateline toponyms in many smaller newspapers. From both figures, we note that Relative Geography ( $\mathcal{H}_2$ ) provides some resolution benefit, but is also noisy in that it causes almost as many precision errors as successes in the case of ACE and in LGL when ignoring the Local Lexicon ( $\mathcal{H}_5$ ). We also observe that for LGL, using the Local Lexicon ( $\mathcal{H}_5$ ) slightly improved the performance of the Dateline ( $\mathcal{H}_1$ ) and Comma Group ( $\mathcal{H}_3$ ) heuristics, which partly rely on  $\mathcal{H}_5$ .

## 6.4 Open Problems

Here, we discuss several open problems related to local lexicons. First, associating a single local lexicon with each data source allows for a variety of applications. However, it may be possible to fine-tune the use of spatial lexicons in situations involving different types of content. For example, a blog may track several different topics simultaneously, and use different spatial lexicons for each topic. Furthermore, individual authors may write for specific audiences as well, as in the case of journalists stationed in certain geographic areas and concentrating on stories in that area. It thus might be beneficial to determine separate spatial lexicons assumed by different authors, and further improve geotagging performance. More generally, we might associate a particular spatial lexicon with any type of entity found in each document, be they authors, persons, organizations, or particular keywords. For example, upon finding a mention of “Robert Mugabe”, we might assume a spatial

lexicon including Zimbabwe and nearby locations, even without mentions in the text.

It would also be interesting to detect and observe evolving spatial lexicons over time for data sources with evolving geographic interests, thus further improving geotagging on these sources. For example, the first few articles of an ongoing, prominent news story will often fully specify the toponyms relevant to the story. Later articles in the series, however, will often underspecify the same toponyms, since they had already been introduced into the audience's spatial lexicon and been fully resolved in earlier articles.

Finally, we plan to further investigate improvements to our heuristics for better performance on a variety of data sources, such as mailing lists (e.g., ProMED [55]) and custom document repositories. Each different data source has different structure and a different audience, which will in turn affect any resulting spatial lexicon. We plan to develop more annotated corpora to allow measurement of heuristic performance across several domains.

## 6.5 Summary

We have shown that modeling and using spatial lexicons, and in particular local lexicons, are vital to ensure successful geotagging. As newspapers and other data sources continue to move into the virtual space of the Web, knowing and using spatial lexicons will be ever more important. Previously localized newspapers will cater to a broader, global audience, and thus will adjust their notion of their audiences' spatial lexicons, perhaps limiting or doing away with an assumed local lexicon altogether. On the other hand, as more people publish highly individual and geographically local content, inferring individual local lexicons will be a necessity for correct geotagging. Geotagging with knowledge of local lexicons will thus continue to play a large role in enabling interesting geospatial applications.

## Chapter 7

### Adaptive Context Features

In the previous two chapters we explored two forms of toponym resolution evidence in the form of comma groups and local lexicons. In this chapter, we introduce another such form of evidence. As a first reminder, recall that geotagging consists of two steps: finding all textual references to geographic locations, known as *toponyms*, and then choosing the correct location interpretation for each toponym (i.e., assigning lat/long values) from a *gazetteer* (database of locations). These two steps are known as *toponym recognition* and *toponym resolution*, the second of which we investigate here, and are difficult due to ambiguities present in natural language. Importantly, both these steps can be considered as *classification* [36] problems: Toponym recognition amounts to classifying each word in the document's text as part of a toponym or not, and toponym resolution amounts to classifying each toponym interpretation as correct or incorrect. With this understanding, and with appropriately annotated datasets, we leverage techniques from supervised machine learning to create an effective geotagging framework. These techniques take as input sets of values known as *feature vectors*, along with a class label for each feature vector, and learn a function that will predict the class label for a new feature vector. Many such techniques for classification, and other machine learning problems, exist and have been used for geotagging purposes, including SVM [8, 56, 97], Bayesian schemes [33, 52, 159], and expectation maximization [17].

The effectiveness of such techniques for a given problem domain depends greatly on

the design of the input *features* that comprise each feature vector. One common feature used for geotagging is the population of each interpretation, since larger places will tend to be mentioned more frequently and are more likely to be correct. However, using population alone or overly relying on it, as many methods do, resulted in greatly reduced accuracy in our experiments, especially for toponym recall. Instead, in this chapter, we consider a new class of features to improve the accuracy of toponym resolution, termed *adaptive context features*. These features construct a window of variable size around each toponym  $t$ , and use the other toponyms in the window to aid in resolving  $t$  correctly by considering the geographic relationships between interpretations  $l_t$  of  $t$  and those of other toponyms in the window. In particular, we search for interpretations that are geographically *proximate* to  $l_t$ , or are *siblings* of  $l_t$  in terms of a geographic hierarchy (e.g., cities in the same state). The more such relationships appear in the window, the greater evidence there is that  $l_t$  is the correct interpretation of  $t$ . These window features are a natural extension of other *context-sensitive* features which depend on other words nearby the toponym, such as *object/container* and *comma group* [80] evidence (see Chapter 5), as well as pairing notions such as *pair strength* (part of STEWARD’s [78] disambiguation algorithm, described in Section 2.3.3).

We call these features *adaptive* because the window’s parameters can be varied for different domains, or to achieve different ends. Some relatively small windows can contain a significant number of highly ambiguous toponyms, and considering all possible combinations of interpretations places inhibitive penalties on feature computation speed. For

... in and around [Louisville 17] and [Lexington 31], [Kentucky 6],  
[Nashville 27] and [Cordova 55], [Tennessee 5], [Richmond 69], [Virginia 42],  
[Fort Lauderdale 1] and [Orlando 9], [Florida 96], [Indianapolis 3], [Indiana 8]  
and [Atlanta 22], [Georgia 12].

Figure 7.1: Excerpt from an Earth Times press release [106] with toponyms and their number of interpretations highlighted, showing the extreme ambiguity of these toponyms and illustrating the need for adaptive context features.

example, consider Figure 7.1, which is an excerpt from a press release [106] published in the Earth Times newspaper, with toponyms highlighted and the numbers next to each toponym indicating the number of interpretations in our gazetteer for the toponym. If we consider all possible combinations of resolutions for these toponyms, this results in about  $3 \cdot 10^{17}$  possibilities, an astonishingly large number for this relatively small portion of text, which is far too many to check in a reasonable time. Instead, we set parameters which we term the window's *breadth* and *depth*, named analogously to breadth-first and depth-first search, which control the number of toponyms in the window and the number of interpretations examined for each toponym in the window, respectively. The adaptive context features thus afford us flexibility since by varying these parameters, we control a tradeoff between feature computation time and resolution accuracy. The more toponyms and toponym interpretations we examine, the more likely we are to find the correct interpretation, but the longer resolution will take, and vice versa. Some textual domains such as Twitter, where tweets arrive at a furious rate, demand faster computation times, while in other, offline domains, the time constraint is relaxed and we can afford to spend more time to gain higher accuracy. While window-like features and heuristics have been used in other work related to geotagging (e.g., [70, 72, 97, 121, 135]), these features' adaptive potential has not been explored.

In the rest of this chapter, we first introduce the geotagging framework that enables us to test our adaptive context features, and describe our toponym recognition and resolution processes as a whole (Section 7.1). We also introduce several other features that complement our adaptive context features and serve as baselines for comparison. Next, we describe our new adaptive context features, as well as algorithms for their computation (Section 7.2). We detail extensive experiments showing our methods' performance benefits over OpenCalais and Placemaker, that also test various feature combinations and parameters (Section 7.3). Finally, we offer potential avenues of future work (Section 7.4) and conclude the chapter (Section 7.5).

## 7.1 Geotagging Framework

In this section, we present the framework that enables testing of our geotagging methods. This framework was originally developed for and is an integral component of the NewsStand [143] (described further in Chapter 3) and TwitterStand [130] systems. We describe our toponym recognition (Section 7.1.1) and resolution (Section 7.1.2) procedures, as well as a set of baseline features (Section 7.1.3) that we use in combination with our adaptive context features, to be described in later sections.

### 7.1.1 Toponym Recognition

Our toponym recognition procedure is designed as a multifaceted process involving many types of recognition methods, both rule-based and statistics-based. For our toponym recognition method here, we use the recognition methods introduced previously in Chapter 4. This multifaceted recognition procedure is designed to be flexible to capture variations that appear in streaming news, and also to be as inclusive as possible when recognizing toponyms, to maximize toponym recall, which comes at the cost of lower precision. Our recognition procedure’s high recall is also corroborated by experimental results in Section 7.3.3. Since toponym recognition is only the first step in a two-part geotagging process, our toponym resolution methods will serve to restore precision to the entire process.

### 7.1.2 Toponym Resolution

As mentioned earlier, geotagging can be understood as a classification problem, and we use methods from supervised machine learning to implement toponym resolution. Specifically, we cast it as a *binary* classification problem, in that we decide for a given toponym/interpretation pair  $(t, l_t)$ , whether  $l_t$  is correct or incorrect. These location interpretations are drawn from a *gazetteer*, which is a database of locations and associated metadata such as population data and hierarchy information. Our gazetteer, which is based on GeoN-



ames [43], is vastly larger than many gazetteers typically used in geotagging methods, which both increases our methods' utility as well as geotagging's difficulty. We characterize our gazetteer further in our experiments in Section 7.3.1.

For our classifier, we use a decision tree-based ensemble classifier method known as *random forests* [20], which has state-of-the-art performance for many classification tasks. Briefly, given an annotated training dataset, the random forests method constructs many decision trees based on different random subsets of the dataset, sampled with replacement. In addition, each decision tree is constructed using random subsets of features from the training feature vectors. Because the features and subsets are chosen randomly, a variety of trees will be in the forest. Classifying a new feature vector is relatively simple: each tree in the forest votes for the vector's class, and the consensus is taken as the result. Note that individual trees may be excellent or poor class predictors, but as long as some features allow better-than-random classification, the forest taken as a whole will be a strong classifier. Another useful aspect of random forests is that the number of trees that vote for a given class can be used as a confidence score for the classification, and provides a means of tuning the precision/recall balance of the classifier. Assuming the score is an accurate estimate of the method's predictability, accepting classifications with a lower score will result in lower precision but higher recall, and vice versa. For our implementation, we used the fast random forest implementation [142], integrated with the Weka machine learning toolkit [49].

As an alternative to classification, Martins et al. [97] considered the use of SVM *regression* to estimate a distance function based on feature vector values that is intended to capture the distance between a given  $l_t$ , and  $t$ 's ground truth interpretation. They use the resulting distance values to rank the interpretations, essentially using them as confidence scores, and select the one with smallest distance value as the interpretation for  $t$ . However, a significant drawback of this technique is that it assumes that all toponyms input to the toponym resolution process are not erroneous, i.e., that the toponym recognition procedure is perfect in identifying toponyms, while in reality, no such procedure is perfect. The dis-

tance measures they compute, while useful for ranking, are not necessarily meaningful as confidence scores for deciding whether a given  $l_t$  has strong enough evidence to consider it correct. For example, an inferred distance of “10” may indicate strong evidence for a given  $l_t$ , but weak evidence for another. On the other hand, our framework using random forests and their confidence scores provide consistent and meaningful scores for deciding classification strength.

### 7.1.3 Resolution Features

In addition to the adaptive context features introduced in the next section, we use several baseline toponym resolution features in our methods. To borrow terms from linguistics, these features, which will be computed for each toponym/interpretation pair  $(t, l_t)$ , can be loosely classed as what we term *context-sensitive* and *context-free* features. Put simply, context-sensitive features depend on  $t$ 's position in relation to other toponyms in the document, while context-free features do not. Note that our adaptive context features subsume and generalize context-sensitive features, so we describe them in the next section. On the other hand, the context-free features we use include the following:

- I:** *interps.* Number of interpretations for  $t$ ; more interpretations means more opportunities for errors.
- P:** *population.* The population of  $l_t$ , where a larger population indicates that  $l_t$  is more well-known.
- A:** *altnames.* Number of alternate names for  $l_t$  in various languages. More names indicates greater renown of  $l_t$ .
- D:** *dateline.* Geographic distance of  $l_t$  from an interpretation of a *dateline* toponym, which establishes a general location context for a news article.

**L:** `locallex`. Geographic distance of  $l_t$  from the newspaper’s *local lexicon* [81], the expected location of its primary audience (further described in Chapter 6).

The `interps`, `population`, and `altnames` features are domain independent, i.e., they can be used in any textual domain, while the `dateline` and `locallex` features are specific to the news domain. In our experiments in Section 7.3.4, we consider these features alone and in various combinations to understand each feature’s relative utility.

## 7.2 Adaptive Context Features

In this section, we present our adaptive context features to aid in the resolution of toponyms. These features reflect two aspects of toponym cooccurrence and the evidence that interpretations impart to each other, which include:

1. *Proximate* interpretations, which are both nearby in the text as well as geographically proximate, and
2. *Sibling* interpretations, which are nearby in the text and share containers in a geographic hierarchy.

We capture these interpretation relationships and encode them in features. To compute these features, we examine for each toponym  $t$  a window of text around  $t$ , and compare interpretations of toponyms in the window with the interpretations of  $t$ . That is, a given interpretation  $l_t$  of  $t$  is promoted if there are other interpretations of toponyms in the window that are geographically proximate to it, or are sibling interpretations. In addition, we vary two parameters of the window termed *window breadth* and *window depth*, which control a tradeoff between computation speed and discriminative utility for the features by changing the number of toponyms in the window, and the number of interpretations per toponym, respectively.

Figure 7.2 is a schematic representation of the algorithm used to compute our features. Each box represents a toponym, and the lines under the boxes represent location interpreta-

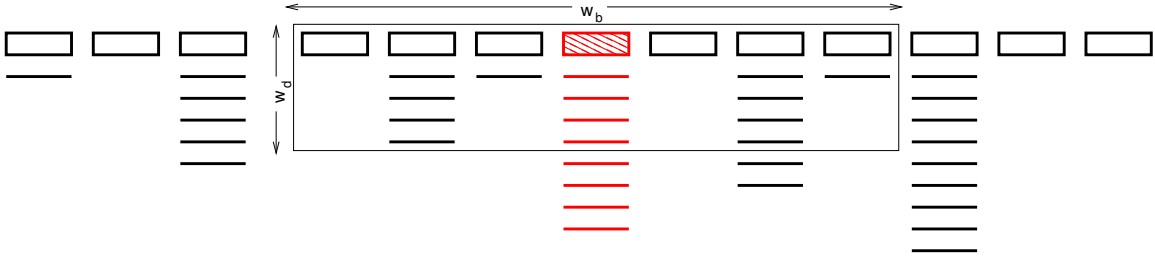


Figure 7.2: Computing adaptive context features, illustrating the window breadth  $w_b$  and window depth  $w_d$ .

tions for each toponym. Different toponyms have different levels of ambiguity, as measured by the number of interpretations for the toponyms. In Figure 7.2, we are computing adaptive context features for the highlighted toponym and its interpretations in the middle. We compute these features for all toponyms at a document distance of less than the window breadth  $w_b$ , and we compare the first few interpretations of each toponym in the window, up to a maximum of  $w_d$  interpretations, the window depth.

Note that our proximity and sibling features subsume and generalize other commonly-used features in toponym resolution. In particular, these features generalize *context-sensitive* features, which compute a toponym interpretation’s likelihood of correctness based on the other toponyms nearby to it in the document. One example of these context-sensitive features includes the *object/container* pair (e.g., “[Paris], [Texas]”, “[Dallas] in [Texas]”), consisting of two toponyms, one of which contains the other. Authors use them when their audiences are not familiar with the location in question, and use the containing toponym to provide a geographic context for the toponym. Object/container pairs are a particularly common type of evidence used in many types of documents, and much research has investigated its utility (e.g., [7, 28, 72, 81, 121, 135]). This type of evidence can be understood as an extreme case of our sibling feature, in the case where the window is restricted to the immediately next or preceding toponym. More general than object/container pairs is the *comma group* [80] (investigated in detail in Chapter 5), which consists of a sequence of toponyms adjacent to each other separated by connector tokens (e.g., “[Paris], [Dallas], [San Antonio] and [Houston]”) that share geographic characteristics (in our example, all

cities in Texas), and hence provide mutual evidence for each others' correct interpretations. These relationships can be captured using our features with a window of appropriate size to contain all the toponyms. Another difference between these sources of evidence and our adaptive context features is that we do not assume any meaning for the specific position of toponyms within the window. For example, we do not consider the grammatical structure involved, or the tokens present between toponyms in the window. This increases the flexibility of our features as compared to, e.g., comma groups, whose recognition depends on specific wording and organization of the toponyms. As noted in Chapter 5, comma groups in particular can be constructed in various ways that can mislead rule-based heuristics, such as in our original example in Figure 7.1.

The following sections describe our proximity (Section 7.2.1) and sibling (Section 7.2.2) features, and the algorithms we use to compute them (Section 7.2.3). We also describe a strategy for propagating significant feature values for a toponym to its other instances in the same document (Section 7.2.4).

### 7.2.1 Proximity Features

The *proximity* features we use are based on geographic distance. Because this distance is continuous, appropriate thresholds for what is considered “near” and “far” are not apparent. Thus, it behooves us to defer their definitions to learning algorithms that can learn appropriate and meaningful distance thresholds.

To compute our proximity features for a toponym/interpretation pair  $(t, l_t)$ , we find for each other toponym  $o$  in the window around  $t$  the closest interpretation  $l_o$  to  $l_t$ . Then, we compute the proximity feature for  $(t, l_t)$  as the average of the geographic distances to the other interpretations. Thus, a lower feature score indicates a higher level of overall geographic proximity for toponyms within the window. This feature strategy also balances fairness with optimism, in that it allows all toponyms in the window to contribute to  $(t, l_t)$ 's feature score, while each toponym in the window contributes its best (i.e., geographically

nearest) interpretation to the feature score. It has the additional benefit that no distance thresholds are hard-coded into the feature. Instead, the learning procedure can learn appropriate distance thresholds from its training data.

### 7.2.2 Sibling Features

Our second class of adaptive context features are those based on *sibling* relationships between interpretations in a geographic hierarchy. In other words, this feature will capture the relationships between textually proximate toponyms that share the same country, state, or other administrative division. The sibling feature is intended to capture interpretations that are at the same level in the hierarchy (e.g., states in the same country, cities in the same state) as well as interpretations at different levels (e.g., a state and its containing country, a city inside its containing state). The first case captures “true” sibling relationships, while the second case captures containment relationships, which can be considered siblings at a coarse granularity (e.g., “College Park” and “Maryland” are state-level siblings).

We compute sibling features in a similar way as the proximity features. For each toponym/interpretation pair  $(t, l_t)$ , we use as our sibling feature value the number of other toponyms  $o$  in the window around  $t$  with an interpretation that is a sibling of  $l_t$  at a given resolution. We consider three levels of resolution, which correspond to three sibling features for each  $(t, l_t)$ : country-level, state-level, and county-level.

Given that the sibling features are so related to the proximity features, at first glance, the sibling feature appears to be redundant in that some toponym interpretations that are siblings will tend to be geographically proximate as well (e.g., “[Paris], [Texas]” and “[Dallas], [Texas]”). However, in some cases the sibling feature will prove helpful in distinguishing toponym relationships. For example, cities that are positioned at opposite ends of a given state might be too far to be considered geographically proximate, but would still be considered siblings. Similarly, the notion of geographic distance for area objects such as countries and states depends on their representation. If we represent, e.g., a country by a single point,

such as its centroid or the location of its capital city, it might be considered geographically distant from many cities contained in it, while the sibling feature would correctly capture these relationships. Another difference between the proximity and sibling features is that the geographic hierarchy is discrete, while geographic distances are continuous values. As a result, we do not have the same thresholding problem as for the proximity features, as our “thresholds” are effectively the same as the hierarchy levels.

### 7.2.3 Feature Computation

As noted earlier, our adaptive context features are based on computing features within a *window* of context around each toponym  $t$ . We consider two variables related to the search for a correct interpretation of  $t$ :

1. *Window breadth*, denoted  $w_b$ , which corresponds to the size of the window around  $t$  to be considered.
2. *Window depth*, denoted  $w_d$ , which is the maximum number of interpretations to be considered for each toponym in the window.

The window breadth  $w_b$  controls how many toponyms around a given toponym  $t$  are to be used in aiding the resolution of  $t$ . With a larger  $w_b$ , more toponyms will be used to resolve  $t$ , thus reducing the resolution algorithm’s speed but hopefully increasing its accuracy. Similarly, the window depth  $w_d$  controls the number of interpretations to be considered for each toponym in the window. A larger  $w_d$  means that more interpretations will be checked, with a resulting decrease in speed, but with more potential for finding corroborating evidence for a correct interpretation of  $t$ .

Because the window depth may preclude examination of all interpretations for a given toponym, the order in which the interpretations are examined is important. Ideally, interpretations would be ordered using context-free attributes of each interpretation. In a sense, the ordering is based on an apriori estimate of each interpretation’s probability of being

mentioned in a given document, though we do not formalize this notion here. We order or rank these interpretations using various factors, which include, in order of importance:

1. Number of alternate names for the location in other languages. GeoNames, being a multilingual gazetteer, contains alternate names and the number of names can indicate the place's renown.
2. Population of the location, where a larger population generally indicates a more well-known place.
3. Geographic distance from a local lexicon location.

These ranking factors can be considered context-free, in that the ordering of interpretations for a given toponym is independent of its position in the document. We could use additional factors such as each interpretation's geographic distance from a dateline toponym interpretation, but because we include these factors as separate features we do not need to include them in the ranking here.

One seeming drawback with regard to the window depth is that it may not seem effective in that most toponyms in our gazetteer have only one or two possible interpretations, as our experiments in Section 7.3.1 show. However, toponyms that are well-known by virtue of having a well-known interpretation (e.g., "Paris", widely known as the French capital), will tend to be mentioned more frequently in documents, and these will be more ambiguous. This is also reflected in measurements made on the toponyms present in our experimental datasets (Section 7.3.2).

In addition, rather than using all toponyms in the window around each  $t$ , we perform some pre-filtering to remove toponyms that detract from the usefulness of our adaptive context features. For example, we do not use toponyms in the window that have the same name as  $t$ , since they will have the same set of interpretations as  $t$ , which will impart no useful information. In addition, and more generally, we may not be sure which of the words in the window correspond to toponyms, due to ambiguities in toponym recognition.



Our toponym recognition process (described in Chapter 4) is designed for high recall and as a result we consider many words which are not true toponyms. In other cases, we may not be sure of the appropriate interpretation for a given toponym. For example, consider the phrase “University of Maryland”, which could be interpreted as a whole, “[University of Maryland]”, referring to the school, or as “University of [Maryland]”, the state. Rather than immediately deciding on one of these toponyms, our recognition process keeps both, even though they overlap. Thus, we keep and consider all of them in toponym resolution, though they must be appropriately filtered when processing toponyms in the window.

---

**Algorithm 7.1** Compute adaptive context features.

---

```

1: procedure ADAPTIVECONTEXT( $T, w_b, w_d$ )
   input: Toponyms  $T$ , window breadth  $w_b$ , window depth  $w_d$ 
   output: Proximity and sibling features
2:   for  $t \in T$  do
3:      $P \leftarrow \{\}$ 
4:      $O \leftarrow \{o \in T : \text{NAME}(t) \neq \text{NAME}(o) \wedge \text{DOCDIST}(t, o) \leq w_b\}$ 
5:     for  $o \in O$  do
6:        $L_o \leftarrow \text{LOCS}(o)[1 \dots \min\{w_d, |\text{LOCS}(o)|\}]$ 
7:       for  $l_t \in \text{LOCS}(t)$  do
8:          $d_{\min} \leftarrow \min\{\forall l_o \in L_o, \text{GEODIST}(l_t, l_o)\}$ 
9:          $P[l_t] \leftarrow P[l_t] \cup \{d_{\min}\}$ 
10:        for  $lev \in \{country, admin1, admin2\}$  do
11:          if  $\exists l_o \in L_o : \text{SIBLING}(l_t, l_o, lev)$  then
12:            Increment  $\text{SIBFEATURE}(t, l_t, lev)$ 
13:          end if
14:        end for
15:      end for
16:    end for
17:    for  $l_t \in \text{LOCS}(t)$  do
18:       $\text{PROXFEATURE}(t, l_t) \leftarrow \text{AVG}(P[l_t])$ 
19:    end for
20:  end for
21: end procedure

```

---

Our algorithm for computing adaptive context features, called ADAPTIVECONTEXT, is shown in Algorithm 7.1. It takes as input the toponyms  $T$  in the document being processed, as well as the window breadth  $w_b$  and window depth  $w_d$  under consideration. The algorithm

proceeds by iterating over all toponyms  $t \in T$  (line 2). For each  $t$ , an array  $P$  is initialized which will hold minimum distances to interpretations of toponyms in the window around  $t$ , which will be used in computing the proximity features for  $t$  (3). Next, other toponyms  $O$  within the window around  $t$  are collected by finding toponyms  $o \in T$  whose document distance is smaller than the window breadth  $w_b$ , and also have a different name than  $t$  (4). We then loop over each toponym  $o \in O$  to begin comparing interpretations of  $t$  and  $o$  (5). First, we collect the location interpretations associated with  $o$ , up to a limit of  $w_d$  interpretations, the window depth (6). Then, we loop over each interpretation  $l_t$  of  $t$  (7), and find the interpretation  $l_o$  of  $o$  with minimum geographic distance from  $l_t$  (8). We add the interpretation  $l_o$  to the location set  $P[l_t]$  associated with  $l_t$  which will be used for computation of the proximity feature for  $l_t$  (9). Next, we compute the sibling features for each level  $lev$  of our geographic hierarchy (10) by checking whether there exists an interpretation  $l_o$  of  $o$  with  $l_t$  as its sibling (11). If so, we increment the sibling feature for that level (12). Finally, after looping over all toponyms of  $O$ , the sibling features are fully computed for each interpretation  $l_t$  of  $t$ , but the proximity feature remains to be completed. We do so for each  $l_t$  (17) by averaging the geographic distances computed for  $l_t$ , which results in the final proximity feature values (18). We use the median geographic distance as our averaging measure.

#### 7.2.4 Feature Propagation

Oftentimes, documents will mention the same toponyms multiple times. When considering pairs of toponyms for use in computing adaptive context features, described in the previous section, these toponym repetitions are ignored because they impart no useful information, since the interpretations for each pair will be the same. However, we still make use of toponym repetition within a single document because the toponyms appear in different contexts (i.e., at different offsets) within the document. Since our adaptive context features are context-sensitive, we apply stronger feature values computed for the toponym in one context to the same toponym in other, weaker contexts.

To leverage these repetitions, as a final processing step, we compute additional features for each  $(t, l_t)$  pair by *propagating* feature values among toponyms in the document that share the same name. We propagate feature values that indicate strong evidence that a given toponym interpretation is correct. For the proximity feature, this corresponds to the lowest average distance values, while for the sibling features, we propagate the largest sibling counts for each level of resolution we consider.

### 7.3 Experiments

In this section, we describe extensive experiments performed on our own and competing geotagging methods. We first establish the general difficulty of geotagging using our large gazetteer, due to a large amount of toponym ambiguity (Section 7.3.1), and then introduce the datasets to be used for measuring geotagging performance, and characterize the toponyms present in them (Section 7.3.2). In terms of geotagging accuracy, we compare our own adaptive method, referred to as “Adaptive”, against two existing prominent competing methods: Thomson Reuters’s OpenCalais, and Yahoo!’s Placemaker. Both OpenCalais and Placemaker are closed-source commercial products, but they do provide public Web APIs which allow for automated geotagging of documents, and hence they have been used extensively in state-of-the-art geotagging and entity recognition research (e.g., [3, 97, 119, 148, 156]). Also note that neither OpenCalais nor Placemaker offer a means of tuning the precision/recall balance, so we could not explore this aspect of the systems. We discuss how well these systems fare against our own methods in terms of toponym recognition (Section 7.3.3) and toponym resolution (Section 7.3.4). For the latter, we also consider various combinations of features and show how they affect resolution accuracy, and use a feature ranking method to measure the importance of each feature when used in resolving toponyms. Finally, we vary the adaptive context parameters of window breadth ( $w_b$ ) and depth ( $w_d$ ), and show how they affect the feature computation time and accuracy of the Adaptive method (Section 7.3.5).

Note that in all our accuracy experiments, we measure performance using *precision*, *recall*, and  $F_1$ -scores as measured over the correct interpretations (see Section 4.3.4 for full descriptions of these measures). We also used 10-fold cross validation to avoid misleading performance numbers due to potential overfitting. Also, we used 100 trees in our random forests, with 5 attributes for each tree, and accepted classifications with at least 0.5 confidence score (i.e., at least half of the trees voted for the interpretation). All experiments were conducted on a Dell Precision 470 workstation with a dual core Intel Xeon 3GHz CPU and 8G RAM.

### 7.3.1 Gazetteer Ambiguity

First, we examined our gazetteer to understand the level of ambiguity of toponyms present in it. The gazetteer contains over 8 million location interpretations, 10 million distinct names, and 5 million alternate names in languages other than English. The gazetteer's large size ensures a high level of ambiguity and ensuing greater difficulty in performing geotagging correctly, when compared to gazetteers used by other systems such as Web-a-Where [7]. For each toponym in the gazetteer, we counted the number of interpretations associated with it, and plotted the results. Results are shown in Figure 7.3. Toponyms in the gazetteer exhibit a power-law relationship in terms of the number of interpretations, in that the vast majority of toponyms a small number of interpretations, while a few toponyms have a very large number of interpretations. Of course, most of these unambiguous toponyms will not be mentioned in a given document, and in our datasets, described in the next section, the documents' toponyms have higher levels of ambiguity.

### 7.3.2 Datasets

In choosing the datasets for our evaluation, we wanted news data from a variety of sources, and for a variety of audiences. To achieve this end, we used three datasets of news in our evaluation: *ACE*, *LGL*, and *CLUST*. The first, *ACE* [87], consists of articles from four

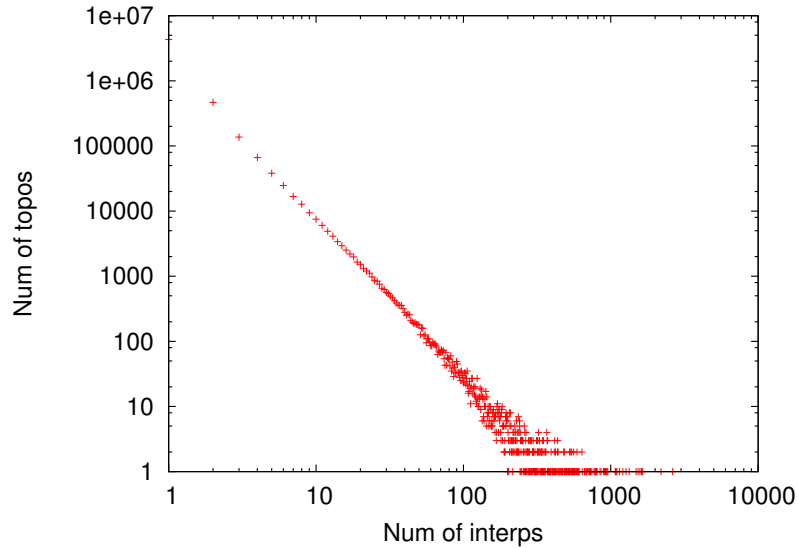


Figure 7.3: Characterization of gazetteer ambiguity. Toponyms and the number of interpretations they have exhibit a power-law relationship.

large news sources: Agence France-Presse, Associated Press World, New York Times, and Xinhua. These articles tend to have a broad world interest and concern topics such as international diplomacy and trade, so they tend to mention large, well-known places. Thus, *ACE* serves in our evaluation as a test of the geotagging methods’ capability for correctly recognizing and resolving well-known, prominent places. On the other hand, to test smaller places, we used the *LGL* [81] dataset (introduced in Chapter 6), which consists of articles from about 100 smaller, more local news sources. These articles are intended for more geographically localized audiences, and concern local events that mention small places. Our third dataset, *CLUST* [76] (introduced in Chapter 4), contains a variety of articles from both large and small news sources.

Table 7.1 presents statistics that broadly illustrate characteristics of our three test corpora. *ACE* is relatively small compared to *LGL* and *CLUST*, both in terms of number of documents and news sources. However, *ACE* tends to have more toponyms per article, which may be due to the content consisting of generally international news involving many different countries and other locations, which would all be mentioned in the articles. In addition, we measured toponym ambiguity in the articles, computing the median number

Table 7.1: Corpora used in evaluating geotagging.

	<i>ACE</i>	<i>LGL</i>	<i>CLUST</i>
Documents	104	621	13,327
Median doc word count	236	242	309
News sources	4	114	1,607
Annotated docs	104	621	1,080
Annotated topos	2,359	4,765	11,564
Distinct topos	295	1,177	2,320
Median topos per doc	12	6	8
Median topo ambig per doc	3	14	7

of interpretations present for toponyms in each document. *LGL* has the largest amount of toponym ambiguity, followed by *CLUST* and *ACE*. This is not overly surprising, given that *LGL* was constructed deliberately focusing on highly ambiguous toponyms [81]. However, the measurements show a high level of ambiguity in all three datasets.

We also classified the annotated locations present in the documents according to their types, which are shown in Figure 7.4. We normalized the type counts for each corpus to illustrate the fractions of each type within each corpus. For cities, we further divided the locations into large cities (over 100,000 population) and small cities (less than 100,000 population). These location types clearly show the important differences between the three corpora. The vast majority of *ACE*'s toponyms, 83%, consist of countries and large cities, indicating *ACE*'s broad geographic scope. This is not overly surprising given that it consists of newswire, which is usually intended for a broad geographic audience. In contrast, 60% of *LGL*'s toponyms are small cities, counties, and states, and among all three datasets, *LGL* contains the smallest fraction of countries and large cities, showing that *LGL* mainly concerns smaller, more local places, with a correspondingly smaller geographic audience. *CLUST* falls in the middle, with the largest fraction of states among the three datasets, and in between the other two in terms of countries, counties, and small cities. Bearing these observations in mind, in terms of overall geographic relevance, *ACE* and *LGL* can be said to have wide and narrow relevance respectively, while *CLUST* falls in the middle, illus-

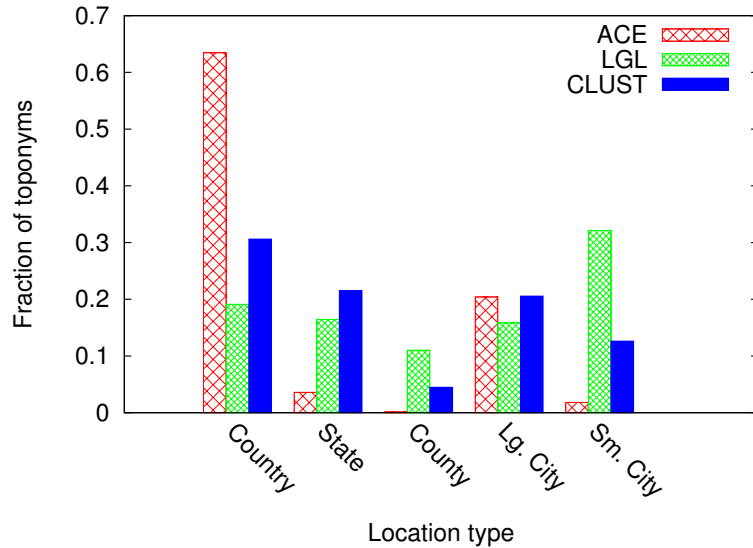


Figure 7.4: Breakdown of location types within each of our test corpora.

trating our three datasets’ utility in testing geotagging at coarse, middle, and fine-grained geographic scopes.

### 7.3.3 Recognition Accuracy

Though the main focus of this chapter is improved toponym resolution, for completeness, we tested each system’s toponym recognition performance when isolated from the subsequent toponym resolution step. Note that OpenCalais and Placemaker also provide lat/long values with each toponym, but we disregard these when testing toponym recognition using these systems because it is more information than we need for this experiment. Table 7.2 shows the performance results for each method’s toponym recognition step. For all three datasets, the Adaptive method shows higher recall performance than either OpenCalais or Placemaker, as well as a higher overall  $F_1$ -score for *LGL* and *CLUST*. While OpenCalais and Placemaker do have higher precision, this is mitigated by their relative lack of recall. Also, Adaptive’s precision is restored by its toponym resolution processing, which will be shown in the next section. These results are also consistent with previously-reported performance [76].

Table 7.2: Recognition performance.

	P	R	F <sub>1</sub> -score
<i>ACE</i>			
Adaptive	0.748	0.867	0.804
OpenCalais	0.883	0.681	0.769
Placemaker	0.899	0.767	0.828
<i>LGL</i>			
Adaptive	0.671	0.723	0.696
OpenCalais	0.588	0.222	0.322
Placemaker	0.675	0.658	0.666
<i>CLUST</i>			
Adaptive	0.732	0.861	0.791
OpenCalais	0.759	0.425	0.545
Placemaker	0.798	0.692	0.741

### 7.3.4 Resolution Accuracy

In the next experiment, we measured the accuracy of each method’s toponym resolution in isolation—that is, if each method were given a set of toponyms, how well the method would select the correct lat/long interpretation for each toponym. Because OpenCalais and Placemaker do not allow for the specification of ground truth toponyms, it is not possible to make direct comparisons of toponym resolution’s recall for these systems. Instead, we report the precision for the resolution process in isolation ( $P_{\text{Resol}}$ ), and the recall for the combined recognition and resolution processes ( $R_{\text{Recog+Resol}}$ ). Also, for the Adaptive method, we used a window breadth  $w_b$  of 80 tokens and unlimited window depth  $w_d$ . Table 7.3 shows the performance results. Of all three methods, the Adaptive method has the best overall precision, especially so for the *LGL* and *CLUST* datasets. Adaptive also maintains this high precision while having high toponym recall. This is best seen for the *LGL* dataset where Adaptive has a 17% advantage over OpenCalais, and a 22% advantage over Placemaker, along with a recall advantage of 32% over OpenCalais and 6% advantage over Placemaker. These performance numbers indicate our method’s superior performance in terms of the toponym resolution task. Examining performance for all the methods across



Table 7.3: Resolution accuracy of various methods.

	$P_{\text{Resol}}$	$R_{\text{Recog+Resol}}$
<i>ACE</i>		
Adaptive	1635/1659 = 0.986	1635/2359 = 0.693
OpenCalais	1062/1080 = 0.983	1062/2359 = 0.450
Placemaker	1161/1219 = 0.952	1161/2359 = 0.492
<i>LGL</i>		
Adaptive	2799/2970 = 0.942	2799/4765 = 0.587
OpenCalais	1260/1632 = 0.772	1260/4765 = 0.264
Placemaker	2516/3466 = 0.726	2516/4765 = 0.528
<i>CLUST</i>		
Adaptive	7143/7440 = 0.960	7143/11564 = 0.618
OpenCalais	5397/6352 = 0.850	5397/11564 = 0.467
Placemaker	7524/8642 = 0.871	7524/11564 = 0.650

the three datasets, the methods performed best on *ACE*, worst on *LGL*, and in the middle for *CLUST*. These results follow our intuition that correctly geotagging documents containing smaller, less well-known locations (*LGL*) is more difficult than for larger, more well-known locations (*ACE*).

Our next set of experiments tested various combinations of features used in the Adaptive method, to illustrate each feature’s overall utility. We used different combinations of the features described in Section 7.1.3, as well as the adaptive context features described in Section 7.2. Table 7.4 contains the performance results, with feature abbreviations corre-

Table 7.4: Toponym resolution accuracy using different feature combinations.

	<i>ACE</i>			<i>LGL</i>			<i>CLUST</i>		
	P	R	F <sub>1</sub>	P	R	F <sub>1</sub>	P	R	F <sub>1</sub>
$\mathcal{F}_I$	0.91	0.41	0.57	0.96	0.26	0.40	0.93	0.30	0.45
$\mathcal{F}_{I,P}$	0.97	0.59	0.74	0.96	0.47	0.63	0.98	0.38	0.55
$\mathcal{F}_{I,P,A}=\mathcal{F}_{B_1}$	0.99	0.84	0.91	0.96	0.61	0.75	0.98	0.71	0.82
$\mathcal{F}_{B_1,D}$	0.99	0.90	0.94	0.96	0.62	0.75	0.98	0.72	0.83
$\mathcal{F}_{B_1,L}$	0.98	0.86	0.92	0.95	0.90	0.93	0.97	0.77	0.86
$\mathcal{F}_{B_1,D,L}=\mathcal{F}_{B_2}$	0.99	0.90	0.94	0.95	0.90	0.93	0.97	0.76	0.86
$\mathcal{F}_{B_1,W_{80,\infty}}$	0.98	0.88	0.93	0.94	0.65	0.77	0.96	0.71	0.82
$\mathcal{F}_{B_2,W_{80,\infty}}$	0.99	0.88	0.93	0.94	0.88	0.91	0.96	0.73	0.83

sponding to those used in Section 7.1.3, and feature combinations indicated with commas (e.g.,  $\mathcal{F}_{I,P}$  combines `interps` and `population`). In addition, we considered two baseline feature combinations  $\mathcal{F}_{B_1}$  and  $\mathcal{F}_{B_2}$ .  $\mathcal{F}_{B_1}$  tested only the domain-independent features ( $\mathcal{F}_{I,P,A}$ ), while  $\mathcal{F}_{B_2}$  also included those features tailored for the news domain ( $\mathcal{F}_{D,L}$ ). We again used our adaptive context feature ( $\mathcal{F}_{W_{80,\infty}}$ ) with window breadth of 80 tokens and unlimited window depth. In general, resolution precision was high for all feature combinations, so the main difference was resolution recall. For *ACE* and *CLUST*, the `dateline` and `locallex` features did not improve  $\mathcal{F}_{B_1}$  much, but `locallex` did make a large difference for *LGL*. Our adaptive context features in general improved  $\mathcal{F}_{B_1}$ , for *LGL* in particular. However, in combination with  $\mathcal{F}_{B_2}$ , the adaptive context features showed little improvement and in some cases lower performance, which is not overly surprising in that domain-specific features will exhibit domain-specific performance, and sometimes, adding features to a model will decrease performance. However, taken as a whole, the results illustrate our adaptive context features’ utility for general geotagging purposes, especially over more simplistic features such as `population`.

We also conducted an experiment to measure the importance or utility of our features for classifying toponym interpretations. This process, also known as *feature selection* or *dimension reduction* [36], ranks the individual features in terms of their overall utility. For our feature importance measure, we used the *gain ratio* [36], a commonly-used, entropy-based measure for decision tree construction. We computed the gain ratio for each feature, and normalized the resulting importance values within each dataset. Results are presented in Figure 7.5. Interestingly, for each dataset, the `interps` and `altnames` features outranked `population`. The `locallex` feature was highly important for *LGL*, though this is not too surprising considering the dataset’s content of smaller, local news articles. The `windowprox` and `window sib` have lower importance values, but interestingly, the `windowprox` feature has almost the same feature value as `population`. `window sib`’s low importance value may be due to it being little-used in the three datasets.

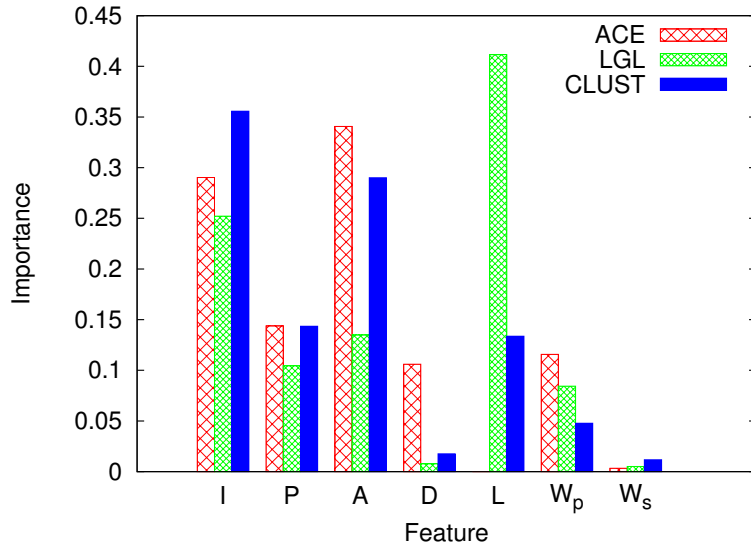
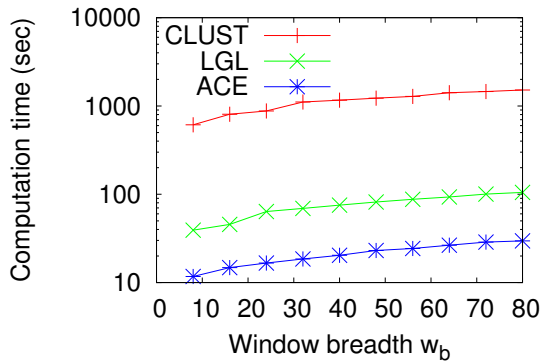


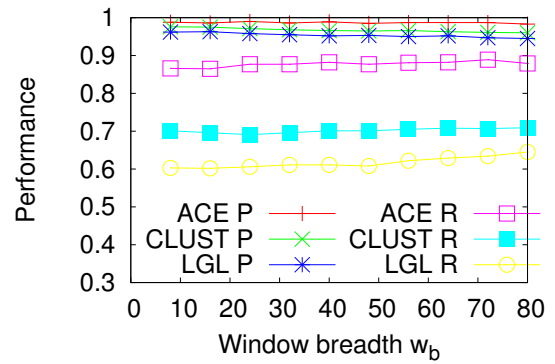
Figure 7.5: Importance of features used in the Adaptive method, as measured by the gain ratio.

### 7.3.5 Adaptive Parameters

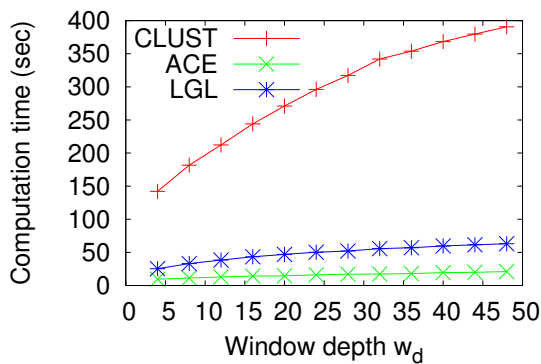
In our final set of experiments, we tested how varying the adaptive parameters of our window features, namely the window breadth  $w_b$  and window depth  $w_d$ , would affect the speed and accuracy tradeoff for our methods. We used our adaptive context features in combination with our first baseline comparison method,  $\mathcal{F}_{B_1}$ , described in the previous section, which is a combination of the `interps`, `population`, and `altnames` features. First, we varied the window breadth between 1–80 tokens and measured the resulting tradeoff. Figures 7.6a and 7.6b show the results in terms of computation time and method accuracy, respectively. As window breadth increases, the computation time increases linearly, which is to be expected. The computation time for *CLUST* is larger than for the other datasets due to its size. Interestingly, even with a small window breadth, precision remains high, and recall is respectable for all datasets, giving evidence that the features are applicable even for domains with little time available for geotagging. Also, while increasing the window breadth, recall also increases for the datasets, showing the time/accuracy tradeoff as expected. Results are similar with varying window depth (Figures 7.6c and 7.6d).



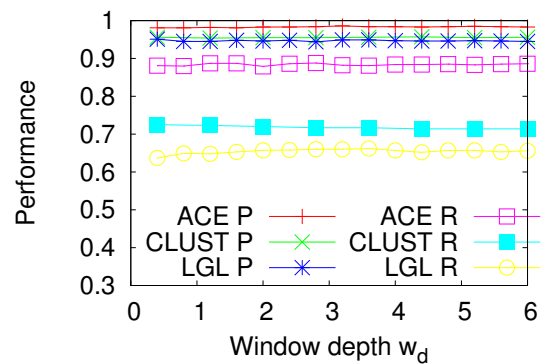
(a)



(b)



(c)



(d)

Figure 7.6: Performance results when varying adaptive window parameters, including varying window breadth in terms of (a) time and (b) accuracy, and varying window depth in terms of (c) time and (d) accuracy.

## 7.4 Open Problems

In future work, we plan to test different weightings of toponyms in the window to judge their effect on resolution accuracy. For example, toponyms that are further away in the window could be given less weight, using linear or Gaussian weighting schemes. In addition, we could consider clusters of news articles about the same topic, which are collected in the NewsStand system, and design other features using these clusters. For example, we might examine other documents in a cluster to get additional toponyms for consideration in geotagging the current document. This can be thought of as creating one large, virtual document consisting of some or all of the documents in a cluster, and then extending the window to include toponyms in those other documents. As before, with large clusters, we may not want to consider all toponyms or all interpretations in other documents in the cluster, due to inhibitive performance penalties.

Also recall that our adaptive context features generalize the comma group methods described in Chapter 5. However, these two methods can be regarded as opposite ends of a context resolution spectrum: comma group evidence captures toponyms that are immediately adjacent to each other, while adaptive context evidence captures toponyms that are more distant. It would be interesting to find the window parameters for which the adaptive context and comma group methods become essentially equivalent.

## 7.5 Summary

Our investigations of adaptive context features have shown their utility and flexibility for improving the geotagging of streaming news. These features, in combination with comma groups (Chapter 5) and local lexicons (Chapter 6) serve as a flexible, useful addition to multifaceted geotagging algorithms for streaming news and other textual domains.

## Chapter 8

### Conclusions and Future Work

This dissertation demonstrated the importance of streaming news, as well as systems that understand the prominently geographic component of streaming news via multifaceted toponym recognition and toponym resolution algorithms. We first introduced the STEWARD (Chapter 2) and NewsStand (Chapter 3) systems which were developed to enable the spatio-textual analysis and querying of documents in the hidden Web and streaming news articles, respectively. These systems crucially involve geotagging algorithms to enable exploration of unstructured text documents using a map query interface, and thus served as convenient platforms on which to test the geotagging algorithms described in the dissertation. They also resulted in innovations in terms of database design and querying, where each interaction in the map query interface is mapped to a top- $k$  query or set of queries in the database. STEWARD was also used as the base for an infectious disease tracking system by geotagging published PubMed articles and ProMED-mail emails. In addition to the geotagger, NewsStand's architecture involves a large number of additional processing modules that compartmentalize the various stages of document processing, and their successful execution required the development of a central pipe server to coordinate module execution and work flows.

Next, we continued with our exposition of the geotagging algorithms developed for these systems. We first described a multifaceted toponym recognition procedure (Chapter 4), using a combination of rule-based and machine learning-based methods that results

in a high toponym recall, at the expense of precision. This high recall is crucial to the geotagging process since toponym recognition effectively upper-bounds recall for the entire geotagging process. Also, our toponym resolution steps, which we described next, restore the geotagging process's toponym precision. Our toponym resolution process incorporates several new types of evidence that improve resolution accuracy. First, we recognized and resolved lists of toponyms termed *comma groups* (Chapter 5) that share common geographic characteristics within the group, namely prominence (population), proximity, or sibling (container-based) relationships. These common characteristics were used to resolve toponyms in each comma group simultaneously. Next, we introduced the notion of *local lexicons* (Chapter 6), which capture the smaller geographic locations known to intended readership. Knowledge of these locations allow for the proper resolution of toponyms present in articles from smaller newspapers, which comprise the vast majority of news sources on the Web. Finally, we considered windows of text around individual toponyms, termed *adaptive context* (Chapter 7), which improve the resolution of toponyms in the window via shared geographic attributes. The window's parameters, in terms of window depth and window breadth, can be varied to exploit a tradeoff between execution speed and resolution accuracy. These forms of recognition and resolution evidence were further encoded as sequences of rules as well as features in a machine learning-based geotagging framework.

In addition to introducing the above systems, algorithms and techniques, we performed extensive experimental evaluations demonstrating the effectiveness of our methods. For NewsStand, we presented data volume and throughput statistics over time, and database querying performance to demonstrate NewsStand's ability to support large numbers of users and queries on its large collection of streaming news. In terms of geotagging evaluation, we created two new corpora of hand-annotated streaming news articles named *LGL* and *CLUST* that are larger than typical collections created for this purpose, and have a greater focus on local news, which as noted before is more representative of most news on the Web. In addition to enabling evaluations of our geotagging algorithms, these cor-

pora allowed for the training of supervised machine learning models, and hence enabled the testing of our machine learning features. We evaluated our various geotagging methods in combination and individually, and also compared them with existing state-of-the-art systems such as Thomson Reuters's OpenCalais, Yahoo!'s Placemaker, and others, showing great improvements over these systems. We also investigated the relative usefulness of each of our geotagging heuristics both in terms of each heuristic's ability to recognize and resolve toponyms, as well as importance scores assigned by machine learning models.

Our multiple, varied geotagging methods have demonstrated the effectiveness of a multifaceted, combined approach to the geotagging of streaming news, involving many sources of evidence. In particular, an emphasis on local, human-based knowledge of locations is vital for geotagging success, especially in the Internet age where local newspapers, bloggers, tweeters, and other local data sources take over the Web. This stands in contrast to previous, simpler geotagging methods using the populations of toponym interpretations alone, or heavily relying on them (i.e., selecting toponyms and interpretations with large population), which will not perform well in this new age of local information. Of course, previously localized newspapers, by virtue of their Web presence, will cater to a broader, more global audience, which may reduce the importance of localized evidence such as local lexicons. On the other hand, as more and more people publish highly individual and geographically local content, such local evidence will be a necessity for correct geotagging. Thus, localized geotagging methods like ours will continue to play a large role in enabling interesting geospatial applications.

Further, the domain of streaming news poses particular challenges that are addressed by our methods. As the prevalence of streaming data on the Web increases, systems such as NewsStand that are capable of quickly processing this streaming data will have ever increasing importance. We believe that the increasing prevalence of geotagged content on the Web will enable compelling applications for systems like STEWARD and NewsStand in other knowledge domains. On another level, and an important contribution of this work,



is that we have highlighted the need for Web-based publication standards that would facilitate and enhance spatio-textual querying and browsing capabilities. Adoption of such standards would enable more up-front rather than backend processing approaches, which would resolve some of the ambiguities mentioned above, and hence greatly improve data mining capabilities on the Web. Of course, with the ever-increasing amount of streaming data on the Web, manual tagging approaches, which are currently prominent, will be replaced by fully-automated approaches such as ours, due to the latter's superior scalability. Additionally, the ever-changing nature of streaming news advises against the sole use of methods based on static, unchanging corpora, which will become obsolescent. As more news sources move online, algorithms like ours which are tailored for streaming news will be vital to handle the resulting data deluge.

In the remaining sections we propose additional avenues of research based on the work described in this dissertation.

## 8.1 Clustering Evidence

One potentially fruitful avenue of research involves the use of clustering methods in toponym recognition and resolution. The NewsStand system [143] (Chapter 3) executes online clustering of the articles it retrieves from the Web. This clustering allows us to group articles from different newspapers that are about the same news story. Note that despite this clustering's seeming redundancy, clustering is useful in several ways. Different articles in the cluster will be written by different authors, each of which impart their own spin, biases, and details to their version of the story. Furthermore, since different articles in a cluster tend to be from different newspapers, they will also be written for different audiences, so authors must tailor their articles to their respective audiences' assumptions and world knowledge.

Importantly, as noted in our previous work on local lexicons (Chapter 6), these assumptions about world knowledge extend to knowledge of locations and location names. As a result, in each different news article in a cluster, locations will be referred to in different

ways, and we can leverage these differences to improve geotagging. We do so by applying stronger location evidence present in one article in a cluster to resolve the same location present in other articles in the cluster, but presented with weaker location evidence. For example, consider a cluster of news articles about a recent US Senator’s visit to “Laurel, MD”, a locality nearby to “College Park, MD”. A news article in The Diamondback (University of Maryland’s student newspaper) that mentions the nearby city of “Laurel, MD” will refer to the location as simply “Laurel”, since the majority of its readers will be familiar with the location and needs no further location evidence to assist in its resolution. On the other hand, a geotagger encountering “Laurel” with no other location evidence might have difficulty resolving it correctly, due to over 40 interpretations of “Laurel” in the US alone. However, another article in the same cluster but published in the New York Times would mention “Laurel, MD”, since having a geographically wide audience, the “Laurel” in question would be unfamiliar to most of its readers. This explicit location evidence is likewise easier to capture in an automated geotagger. Furthermore, since the Diamondback and New York Times articles appeared in the same cluster, the geotagger can use the “Laurel, MD” evidence from the New York Times article to resolve “Laurel” in the Diamondback article. This reuse of resolved locations in the cluster based on stronger location evidence is the essence of our proposed approach.

## 8.2 Streaming Lexicon

As noted earlier, online news has a significant streaming nature, in that the constantly evolving news cycle results in differing entities appearing in the news for short time scales. As a result, this severely limits methods based on small, static corpora of news. However, we can take advantage of the streaming nature of news by maintaining a collection of current events and important keywords in the news, including prominent people and, especially, important locations. In essence, we can create a *streaming lexicon*, analogous to a local lexicon of geographic knowledge, except that the streaming lexicon constantly evolves with the news

cycle. That is, the streaming lexicon contains entities that exist in the popular consciousness, and thus serve as default interpretations for particular entities. For example, a 2012 article mentioning “Obama” would likely be referring to the US politician, and not the city in Japan, given the prominence of the former in the news cycle when compared to the latter.

### 8.3 Toponym Corpora

In terms of evaluation, we have several ideas that can innovate in this area. To start, recall that the domain of news feeds is dynamic with a constantly evolving stream of live news articles. Thus, traditional methods, which rely on small, hand-annotated, static corpora for evaluation are not very useful in measuring performance in this domain. Moreover, people can simply tune the performance of their system to the elements of the corpus by making adjustments to the underlying geotagging code so that the errors do not arise on subsequent incarnations of the system on the same set of articles. In a machine learning context, this is referred to as *overtraining*.

The main difficulty with regard to traditional linguistic evaluation is the time-consuming and extremely large manual effort required to annotate toponyms and other entities in documents. As a result, these corpora tend to be rather small, with sizes of at most 1,000 documents, and usually under 500, and are generally based on a articles from a single, usually prominent news source (e.g., New York Times, Reuters). In contrast, the NewsStand system [143], described in Chapter 3, retrieves over 50,000 documents from multiple thousand local news sources *per day*. Clearly, such a small corpus cannot adequately represent the sheer volume of information on the Web. Unfortunately, due to the huge effort required for full manual annotation, creating a large corpus of articles from these sources is not viable.

Note that our goal is not to understand language or look for a set of documents that are typical. We want our toponym recognition and resolution methods to work for all data, and not for just one set of articles. Language and usage change over time, while the set of possible toponyms is fixed. Thus, rather than solely using a document corpus of arti-

cles, we can instead use a *toponym corpus* of words that can be interpreted as toponyms. They can be terms that proved particularly troublesome in NewsStand or STEWARD in the past, or that are especially ambiguous and hence difficult to resolve properly. A good example is “Northern District”, which frequently appears in articles as “Northern District of . . .”, while instead it is generally placed in Israel. The idea of creating corpora based on difficult-to-resolve toponyms was captured somewhat in the creation of *LGL*, described in Chapter 6, which was made by selecting articles from news sources that were located in multiply ambiguous places—e.g., places named “Paris”, “London”, and so on. In essence, *LGL*’s creation was motivated by corpora of news source locations, rather than corpora of toponyms themselves.

Using such corpora of toponyms, we could see how performance varies over time as the documents vary from one evaluation to another, while the actual data on which the performance is being evaluated has an element of continuity. This is a way of making the corpus dynamic, as otherwise basing it solely on the documents does not account for changes in usage. Our proposed evaluation process ensures that we correctly recognize and resolve toponyms over all instances of time rather than just for one collection of articles. Thus our techniques are said to work if the precision and recall are relatively constant or, ideally, improve over time. The improvement is a result of learning from our mistakes, which we would be doing by including terms that have not been correctly recognized and/or resolved in past evaluations.

Like evaluation on corpora of documents, this evaluation method requires manual annotation of toponyms in collections of documents, to gauge the geotagger’s performance on these toponyms. However, we would only evaluate the geotagger’s performance on the corpus terms in the article collection. This is an important point, as we need not annotate the entire article. Instead, we only have to annotate the terms that we are looking for. As a result, we can create much larger collections of documents with the same amount of effort.

## Bibliography

- [1] 10gen. MongoDB. URL <http://www.mongodb.org>. Accessed 28 Mar 2012.
- [2] 11alive.com. Metro Atlanta under flood watch Wednesday. URL [http://www.11alive.com/rss/rss\\_story.aspx?storyid=138246](http://www.11alive.com/rss/rss_story.aspx?storyid=138246). Accessed 16 Jan 2010.
- [3] R. Abascal-Mena and E. López-Ornelas. Geo information extraction and processing from travel narratives. In *ELPUB'10: Proceedings of the 14th International Conference on Electronic Publishing*, pages 363–373, Helsinki, Finland, June 2010.
- [4] M. D. Adelfio, M. D. Lieberman, H. Samet, and K. A. Firozvi. Ontuition: Intuitive data exploration via ontology navigation. In *GIS'10: Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 540–541, San Jose, CA, Nov. 2010.
- [5] Agence France-Presse. Russia thrusts into South Ossetia; clashes with Georgia reported. URL <http://afp.google.com/article/ALeqM5hpNRP9ysixHH3P9izLJRjYT1ATkA>. Accessed 09 Aug 2008.
- [6] Alias-i. LingPipe home. URL <http://alias-i.com/lingpipe>. Accessed 26 Aug 2010.
- [7] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-Where: Geotagging web content. In *SIGIR'04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 273–280, Sheffield, UK, July 2004.
- [8] I. Anastácio, B. Martins, and P. Calado. Classifying documents according to locational relevance. In *EPIA'09: Proceedings of the 14th Portuguese Conference on Artificial Intelligence*, pages 598–609, Aveiro, Portugal, Oct. 2009.
- [9] A. Angel, C. Lontou, and D. Pfoser. Qualitative geocoding of persistent web pages. In *GIS'08: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 163–172, Irvine, CA, Nov. 2008.
- [10] Apache Software Foundation. Apache CouchDB, . URL <http://couchdb.apache.org>. Accessed 28 Mar 2012.

- [11] Apache Software Foundation. HBase, . URL <http://hbase.apache.org>. Accessed 28 Mar 2012.
- [12] W. G. Aref and H. Samet. Efficient processing of window queries in the pyramid data structure. In *PODS'90: Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 265–272, Nashville, TN, Apr. 1990.
- [13] Associated Press. AP Mobile. URL <http://www.getapmobile.com>. Accessed 26 Aug 2010.
- [14] E. Banos, I. Katakis, N. Bassiliades, G. Tsoumakas, and I. Vlahavas. PersoNews: A personalized news reader enhanced by machine learning and semantic filtering. In *ODBASE'06: Proceedings of the 5th International Conference on Ontologies, DataBases, and Applications of Semantics*, pages 975–982, Montpellier, France, Oct. 2006.
- [15] M. Bartlett. Photo book commemorates county bicentennial. URL [http://www.thisweeknews.com/live/content/delaware/stories/2009/11/21/1122dephoto-book\\_ln.html](http://www.thisweeknews.com/live/content/delaware/stories/2009/11/21/1122dephoto-book_ln.html). Accessed 16 Jan 2010.
- [16] Belfast Telegraph. Hurricane Igor rapidly strengthens. URL <http://www.belfasttelegraph.co.uk/news/world-news/hurricane-igor-rapidly-strengthens-14947034.html>. Accessed 17 Sep 2010.
- [17] P. N. Bennett, F. Radlinski, R. W. White, and E. Yilmaz. Inferring and using location metadata to personalize web search. In *SIGIR'11: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 135–144, Beijing, China, July 2011.
- [18] D. Besagni and A. Belaid. Citation recognition for scientific publications in digital libraries. In *DIAL'04: Proceedings of the International Workshop on Document Image Analysis for Libraries*, pages 244–252, Palo Alto, CA, Jan. 2004.
- [19] A. Blessing, R. Kuntz, and H. Schütze. Towards a context model driven german geo-tagging system. pages 25–30.
- [20] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct. 2001.
- [21] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *WWW'98: Proceedings of the 7th World Wide Web Conference*, pages 107–117, Brisbane, Australia, Apr. 1998.
- [22] D. Buscaldi and B. Magnini. Grounding toponyms in an Italian local news corpus. In *GIR'10: Proceedings of the 6th Workshop on Geographic Information Retrieval*, Zurich, Switzerland, Feb. 2010.

- [23] D. Buscaldi and P. Rosso. A conceptual density-based approach for the disambiguation of toponyms. *IJGIS: International Journal of Geographical Information Science*, 22(3):301–313, Mar. 2008.
- [24] T. Carson. Oops, wrong Vancouver. URL <http://www.reuters.com/article/idUUSTRE61300C20100204>. Accessed 17 Dec 2010.
- [25] Centers for Disease Control and Prevention. CDC - diseases & conditions. URL <http://www.cdc.gov/diseasesconditions>. Accessed 06 Oct 2010.
- [26] N. Chinchor and B. Sundheim. *Message Understanding Conference (MUC) 6*. Linguistic Data Consortium, Philadelphia, PA, Aug. 2003. LDC Catalog Number LDC2003T13.
- [27] C. Clifton, J. Griffith, and R. Holland. GeoNODE: An end-to-end system from research components. pages 12–14.
- [28] P. Clough. Extracting metadata for spatially-aware information retrieval on the internet. In *GIR'05: Proceedings of the 2005 Workshop on Geographic Information Retrieval*, pages 25–30, Bremen, Germany, Nov. 2005.
- [29] N. Collier, S. Doan, A. Kawazoe, R. M. Goodwin, M. Conway, Y. Tateno, Q.-H. Ngo, D. Dien, A. Kawtrakul, K. Takeuchi, M. Shigematsu, and K. Taniguchi. BioCaster: detecting public health rumors with a Web-based text mining system. *Bioinformatics*, 24(24):2940–2941, Oct. 2008.
- [30] W. Cui, H. Qu, W. Zhang, and S. Skiena. Watch the story unfold with textwheel: Visualization of large-scale news streams. *TIST: ACM Transactions on Intelligent Systems and Technology*, 3(2), Feb. 2012. Article 20.
- [31] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *ACL'02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175, Philadelphia, PA, July 2002.
- [32] M.-Y. Day, T.-H. Tsai, C.-L. Sung, C.-W. Lee, S.-H. Wu, C.-S. Ong, and W.-L. Hsu. A knowledge-based approach to citation extraction. In *IRI'05: Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration*, pages 50–55, Las Vegas, NV, Aug. 2005.
- [33] R. O. de Alencar, C. A. Davis, Jr., and M. A. Gonçalves. Geographical classification of documents using evidence from Wikipedia. In *GIR'10: Proceedings of the 6th Workshop on Geographic Information Retrieval*, Zurich, Switzerland, Feb. 2010.
- [34] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *CACM: Communications of the ACM*, 51(1):107–113, Jan. 2008.

- [35] J. Ding, L. Gravano, and N. Shivakumar. Computing geographical scopes of Web resources. In *VLDB'00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 545–556, Cairo, Egypt, Sept. 2000.
- [36] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, second edition, 2001.
- [37] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL'05: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370, Ann Arbor, MI, June 2005.
- [38] A. Finn, N. Kushmerick, and B. Smyth. Fact or fiction: Content classification for digital libraries. In *Proceedings of the 2nd DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries*, Dublin, Ireland, June 2001.
- [39] W. N. Francis and H. Kucera. *Brown Corpus Manual*. Brown University, Providence, RI, third edition, 1979.
- [40] C. C. Freifeld, K. D. Mandl, B. Y. Reis, and J. S. Brownstein. HealthMap: Global infectious disease monitoring through automated classification and visualization of internet media reports. *JAMIA: Journal of the American Medical Informatics Association*, 15(2):150–157, Mar. 2008.
- [41] E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: Providing personalized newsfeeds via analysis of information novelty. In *WWW'04: Proceedings of the 13th International World Wide Web Conference*, pages 482–490, New York, May 2004.
- [42] E. Garbin and I. Mani. Disambiguating toponyms in news. In *HLT/EMNLP'05: Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 363–370, Vancouver, Canada, Oct. 2005.
- [43] GeoNames. GeoNames. URL <http://geonames.org>. Accessed 25 Aug 2010.
- [44] K. Goerte. Paris girl will ride in children's parade. URL <http://www.theparisnews.com/story.lasso?ewcd=76e62af33f955f49>. Accessed 16 Jan 2010.
- [45] Google. Google news. URL <http://news.google.com>. Accessed 26 Aug 2010.
- [46] C. Gouvêa, S. Loh, L. F. F. Garcia, E. B. da Fonseca, and I. Wendt. Discovering location indicators of toponyms from news to improve gazetteer-based geo-referencing. In *GEOINFO'08: Proceedings of the 10th Brazilian Symposium on GeoInformatics*, Rio de Janeiro, Brazil, Dec. 2008.



- [47] S. Gupta, G. E. Kaiser, P. Grimm, M. F. Chiang, and J. Starren. Automating content extraction of HTML documents. *World Wide Web*, 8(2):179–224, June 2005.
- [48] J. Gustavson. All grown up: 4 seniors lead Alta. URL [http://www.sltrib.com/ci\\_13856744](http://www.sltrib.com/ci_13856744). Accessed 16 Jan 2010.
- [49] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, June 2009.
- [50] M. Henzinger, B.-W. Chang, B. Milch, and S. Brin. Query-free news search. *World Wide Web*, 8(2):101–126, June 2005.
- [51] E. Hetzner. A simple method for citation metadata extraction using hidden markov models. In *JCDL’08: Proceedings of the 8th ACM/IEEE Joint Conference on Digital Libraries*, pages 280–284, Pittsburgh, PA, June 2008.
- [52] Y.-H. Hu and L. Ge. A supervised machine learning approach to toponym disambiguation. In A. Scharl and K. Tochtermann, editors, *The Geospatial Web*, pages 117–128. Springer, London, 2007.
- [53] X. Huang and C. S. Jensen. Towards a streams-based framework for defining location-based queries. In *STDBM’04: Proceedings of the 2nd International Workshop on Spatio-Temporal Database Management*, pages 57–64, Toronto, Canada, Aug. 2004.
- [54] HUD USER. Huduser home page. URL <http://www.huduser.org>. Accessed 10 Jan 2012.
- [55] International Society for Infectious Diseases. Main ProMED-mail. URL <http://www.promedmail.org>. Accessed 02 Sep 2010.
- [56] U. Irmak and R. Kraft. A scalable machine-learning approach for semi-structured named entity recognition. In *WWW’10: Proceedings of the 19th International World Wide Web Conference*, pages 461–470, Raleigh, NC, Apr. 2010.
- [57] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. In *EuroSys’07: Proceedings of the 2007 European Conference on Computer Systems*, pages 59–72, Lisbon, Portugal, Mar. 2007.
- [58] A. Jaafar. Mayor of Batman sues WB, Nolan. URL <http://www.variety.com/article/VR1117995653>. Accessed 14 Jan 2011.
- [59] F. T. Johnsen, T. Hafsøe, C. Griwodz, and P. Halvorsen. Workload characterization for news-on-demand streaming services. In *IPCCC’07: Proceedings of the 26th IEEE International Performance Computing and Communications Conference*, pages 314–323, New Orleans, LA, Apr. 2007.

- [60] C. B. Jones, R. S. Purves, P. D. Clough, and H. Joho. Modelling vague places with knowledge from the Web. *IJGIS: International Journal of Geographical Information Science*, 22(10):1045–1065, Oct. 2008.
- [61] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, Upper Saddle River, NJ, 2000.
- [62] W. Kienreich, M. Granitzer, and M. Lux. Geospatial anchoring of encyclopedia articles. In *IV'06: Proceedings of the 10th International Conference on Information Visualization*, pages 211–215, London, July 2006.
- [63] M. Kramer, H. Kaprykowsky, D. Keysers, and T. Breuel. Bibliographic meta-data extraction using probabilistic finite state transducers. In *ICDAR'07: Proceedings of the 9th International Conference on Document Analysis and Recognition*, pages 609–613, Curitiba, Brazil, Sept. 2007.
- [64] M. Krstajic, F. Mansmann, A. Stoffel, M. Atkinson, and D. A. Keim. Processing online news streams for large-scale semantic analysis. In *ICDEW'10: Proceedings of the 26th International Conference on Data Engineering Workshops*, pages 215–220, Long Beach, CA, Mar. 2010.
- [65] V. Kulesh, V. A. Petrushin, and I. K. Sethi. The PERSEUS project: Creating personalized multimedia news portal. In *MDM/KDD'01: Proceedings of the 2nd International Workshop on Multimedia Data Mining*, pages 31–37, San Francisco, Aug. 2001.
- [66] J. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML'01: Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, MA, June 2001.
- [67] N. Leavitt. Will NoSQL databases live up to their promise? *Computer*, 43(2):12–14, 2010.
- [68] J. L. Leidner. An evaluation dataset for the toponym resolution task. *CEUS: Computers, Environment, and Urban Systems*, 30(4):400–417, July 2006.
- [69] J. L. Leidner. *Toponym Resolution in Text: Annotation, Evaluation and Applications of Spatial Grounding of Place Names*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, 2007.
- [70] J. L. Leidner, G. Sinclair, and B. Webber. Grounding spatial named entities for information extraction and question answering. In *Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References*, pages 31–38, Edmonton, Canada, May 2003.
- [71] J. Leveling and S. Hartrumpf. On metonymy recognition for geographic information retrieval. *IJGIS: International Journal of Geographical Information Science*, 22(3): 289–299, Mar. 2008.

- [72] H. Li, R. K. Srihari, C. Niu, and W. Li. InfoXtract location normalization: a hybrid approach to geographic references in information extraction. In *Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References*, pages 39–44, Edmonton, Canada, May 2003.
- [73] Y. Li. Probabilistic toponym resolution and geographic indexing and querying. Master’s thesis, University of Melbourne, Melbourne, Australia, Sept. 2007.
- [74] E. Liarou, R. Goncalves, and S. Idreos. Exploiting the power of relational databases for efficient stream processing. In *EDBT’09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 323–334, St. Petersburg, Russia, Mar. 2009.
- [75] M. D. Lieberman and J. Lin. You are where you edit: Locating Wikipedia users through edit histories. In *ICWSM’09: Proceedings of the 3rd International AAAI Conference on Weblogs and Social Media*, pages 106–113, San Jose, CA, May 2009.
- [76] M. D. Lieberman and H. Samet. Multifaceted toponym recognition for streaming news. In *SIGIR’11: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 843–852, Beijing, China, July 2011.
- [77] M. D. Lieberman and H. Samet. Adaptive context features for toponym resolution in streaming news. In *SIGIR’12: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 731–740, Portland, OR, Aug. 2012.
- [78] M. D. Lieberman, H. Samet, J. Sankaranarayanan, and J. Sperling. STEWARD: Architecture of a spatio-textual search engine. In *GIS’07: Proceedings of the 15th ACM International Symposium on Geographic Information Systems*, pages 186–193, Seattle, WA, Nov. 2007.
- [79] M. D. Lieberman, J. Sankaranarayanan, H. Samet, and J. Sperling. Augmenting spatio-textual search with an infectious disease ontology. In *IIMAS’08: Proceedings of the Workshop on Information Integration Methods, Architectures, and Systems*, pages 266–269, Cancún, Mexico, Apr. 2008.
- [80] M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *GIR’10: Proceedings of the 6th Workshop on Geographic Information Retrieval*, Zurich, Switzerland, Feb. 2010.
- [81] M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging with local lexicons to build indexes for textually-specified spatial data. In *ICDE’10: Proceedings of the 26th International Conference on Data Engineering*, pages 201–212, Long Beach, CA, Mar. 2010.

- [82] H.-S. Lim, J.-G. Lee, M.-J. Lee, K.-Y. Whang, and I.-Y. Song. Continuous query processing in data streams using duality of data and queries. In *SIGMOD'06: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 313–324, Chicago, June 2006.
- [83] S.-H. Lin and J.-M. Ho. Discovering informative content blocks from Web documents. In *KDD'02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 588–593, Edmonton, Canada, July 2002.
- [84] J. Liu and L. Birnbaum. LocalSavvy: Aggregating local points of view about news issues. In *LocWeb'08: Proceedings of the 1st International Workshop on Location and the Web*, pages 33–40, Beijing, China, Apr. 2008.
- [85] M. Lynch. Summer's long gone, but stars remain. URL <http://www.heraldnet.com/article/20091129/LIVING/711299981>. Accessed 16 Jan 2010.
- [86] T. Maeda. Japan's Obama town overjoyed. URL <http://www.reuters.com/article/idUSTRE4A448W20081105>. Accessed 14 Jan 2011.
- [87] I. Mani, J. Hitzeman, J. Richer, and D. Harris. *ACE 2005 English SpatialML Annotations*. Linguistic Data Consortium, Philadelphia, PA, Jan. 2008. LDC Catalog Number LDC2008T03.
- [88] I. Mani, J. Hitzeman, J. Richer, D. Harris, R. Quimby, and B. Wellner. SpatialML: Annotation scheme, corpora, and tools. In *LREC'08: Proceedings of the 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco, May 2008.
- [89] D. Manov, A. Kiryakov, B. Popov, K. Bontcheva, D. Maynard, and H. Cunningham. Experiments with geographic knowledge for information extraction. In *Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References*, pages 1–9, Edmonton, Canada, May 2003.
- [90] C. Mantratzis, M. Orgun, and S. Cassidy. Separating XHTML content from navigation clutter using DOM-structure block analysis. In *HT'05: Proceedings of the 16th ACM Conference on Hypertext and Hypermedia*, pages 145–147, Salzburg, Austria, Sept. 2005.
- [91] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.
- [92] S. Mardis and J. Burger. Design for an integrated gazetteer database. Technical Report MTR-05B0000085, MITRE Corporation, McLean, VA, Nov. 2005.

- [93] D. M. Mark and A. U. Frank. Concepts of space and spatial language. In *Proceedings of the 9th International Symposium on Computer-Assisted Cartography*, pages 538–556, Baltimore, MD, Apr. 1989.
- [94] D. Martin, H. Wu, and A. Alsaïd. Hidden surveillance by Web sites: Web bugs in contemporary use. *CACM: Communications of the ACM*, 46(12):258–264, Dec. 2003.
- [95] B. Martins, H. Manguinhas, and J. Borbinha. Extracting and exploring the geo-temporal semantics of textual resources. In *ICSC’08: Proceedings of the 2nd IEEE International Conference on Semantic Computing*, pages 1–9, Santa Clara, CA, Aug. 2008.
- [96] B. Martins, H. Manguinhas, J. Borbinha, and W. Siabato. A geo-temporal information extraction service for processing descriptive metadata in digital libraries. *e-Perimtron*, 4(1):25–37, 2009.
- [97] B. Martins, I. Anastácio, and P. Calado. A machine learning approach for resolving place references in text. In *AGILE’10: Proceedings of the 13th AGILE International Conference on Geographic Information Science*, pages 221–236, Guimarães, Portugal, May 2010.
- [98] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS’04: Proceedings of the 18th Annual Conference on Neural Information Processing Systems*, Whistler, Canada, Dec. 2004.
- [99] MetaCarta. MetaCarta GeoTagger v4.5.0, . URL <http://developers.metacarta.com/product/metacarta-geotagger>. Accessed 21 Jan 2011.
- [100] MetaCarta. MetaCarta GeoSearch News, . URL <http://geosearch.metacarta.com>. Accessed 26 Aug 2010.
- [101] Microsoft. Top stories - Bing news. URL <http://www.bing.com/news>. Accessed 26 Aug 2010.
- [102] M. F. Mokbel, X. Xiong, W. G. Aref, S. E. Hambrusch, S. Prabhakar, and M. A. Hammad. Continuous query processing of spatio-temporal data streams in PLACE. In *STDBM’04: Proceedings of the 2nd International Workshop on Spatio-Temporal Database Management*, pages 73–80, Toronto, Canada, Aug. 2004.
- [103] D. R. Montello, M. F. Goodchild, J. Gottsegen, and P. Fohl. Where’s downtown? behavioral methods for determining referents of vague spatial queries. *Spatial Cognition & Computation*, 3(2–3):185–204, Sept. 2003.
- [104] National Center for Biotechnology Information. PubMed home. URL <http://www.ncbi.nlm.nih.gov/pubmed>.

- [105] National Geospatial-Intelligence Agency. NGA: GNS Home. URL <http://earth-info.nga.mil/gns/html>. Accessed 25 Aug 2010.
- [106] NTS Realty Holdings. NTS Realty Holdings limited partnership announces refinancing of debt on eight multifamily properties. URL <http://www.earthtimes.org/articles/show/nts-realty-holdings-limited-partnership,1062375.shtml>. Accessed 16 Jan 2010.
- [107] OBO Foundry. The open biological and biomedical ontologies. URL <http://www.obofoundry.org>. Accessed 07 Oct 2010.
- [108] L. O’Callaghan, N. Nishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In *ICDE’02: Proceedings of the 18th International Conference on Data Engineering*, pages 685–694, San Jose, CA, Feb. 2002.
- [109] S. E. Overell. *Geographic Information Retrieval: Classification, Disambiguation and Modelling*. PhD thesis, Imperial College London, London, July 2009.
- [110] S. E. Overell and S. Rüger. Using co-occurrence models for placename disambiguation. *IJGIS: International Journal of Geographical Information Science*, 22 (3):265–287, Mar. 2008.
- [111] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL’02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, July 2002.
- [112] L. L. Petersen and B. S. Davie. *Computer Networks: A Systems Approach*. Elsevier, New York, 2007.
- [113] W. Petroski. Amtrak riders break record at Iowa stops. URL <http://www.desmoinesregister.com/article/20091129/NEWS10/911290343>. Accessed 16 Jan 2010.
- [114] R. Plushnick-Masti. Olmsted’s 1895 Pa. steel town seeks green rebirth. URL [http://hosted.ap.org/dynamic/stories/U/US\\_OLMSTEDS\\_GREEN\\_TOWN\\_PAOL-](http://hosted.ap.org/dynamic/stories/U/US_OLMSTEDS_GREEN_TOWN_PAOL-). Accessed 16 Jan 2010.
- [115] M. F. Porter. An algorithm for suffix stripping. In K. S. Jones and P. Willet, editors, *Readings in Information Retrieval*, pages 313–316. Morgan Kaufmann, San Francisco, 1997.
- [116] PostgreSQL Global Development Group. PostgreSQL. URL <http://www.postgresql.org>. Accessed 26 Mar 2012.

- [117] B. Pouliquen, M. Kimler, R. Steinberger, C. Ignat, T. Oellinger, K. Blackler, F. Fuart, W. Zaghoulani, A. Widiger, A.-C. Forslund, and C. Best. Geocoding multilingual texts: Recognition, disambiguation, and visualization. In *LREC'06: Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 53–58, Genoa, Italy, May 2006.
- [118] R. S. Purves, P. Clough, C. B. Jones, A. Arampatzis, B. Bucher, D. Finch, G. Fu, H. Joho, A. K. Syed, S. Vaid, and B. Yang. The design and implementation of SPIRIT: a spatially aware search engine for information retrieval on the Internet. *IJGIS: International Journal of Geographical Information Science*, 21(7):717–745, Aug. 2007.
- [119] T. Qin, R. Xiao, L. Fang, X. Xie, and L. Zhang. An efficient location extraction algorithm by leveraging web contextual information. In *GIS'10: Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 53–60, San Jose, CA, Nov. 2010.
- [120] G. Quercini, H. Samet, J. Sankaranarayanan, and M. D. Lieberman. Determining the spatial reader scopes of news sources using local lexicons. In *GIS'10: Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 43–52, San Jose, CA, Nov. 2010.
- [121] E. Rauch, M. Bukatin, and K. Baker. A confidence-based framework for disambiguating geographic terms. In *Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References*, pages 50–54, Edmonton, Canada, May 2003.
- [122] Research and Markets. Research and markets: Multi utility meter report, database & directory ed 7 2009. URL <http://www.earthtimes.org/articles/show/research-and-markets-multi-utility,1041523.shtml>. Accessed 16 Jan 2010.
- [123] K. Roberts, C. A. Bejan, and S. Harabagiu. Toponym disambiguation using events. In *FLAIRS'10: Proceedings of the 23rd International Florida Artificial Intelligence Research Society Conference*, pages 271–276, Daytona Beach, FL, May 2010.
- [124] C. Sallaberry, M. Gaio, J. Lesbegueries, and P. Loustau. A semantic approach for geospatial information extraction from unstructured documents. In A. Scharl and K. Tochtermann, editors, *The Geospatial Web*, pages 93–104. Springer, London, 2007.
- [125] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *IPM: Information Processing & Management*, 24(5):513–523, 1988.
- [126] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *CACM: Communications of the ACM*, 18(11):613–620, Nov. 1975.
- [127] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, 2006.

- [128] H. Samet, M. D. Lieberman, J. Sankaranarayanan, and J. Sperling. STEWARD: Demo of spatio-textual extraction on the web aiding the retrieval of documents. In *DG.O'07: Proceedings of the 8th Annual International Conference on Digital Government Research, Bridging Disciplines & Domains*, pages 300–301, Philadelphia, PA, May 2007.
- [129] H. Samet, B. E. Teitler, M. D. Adelfio, and M. D. Lieberman. Adapting a map query interface for a gesturing touch screen interface. In *WWW'11: Proceedings of the 20th International World Wide Web Conference*, pages 257–260, Hyderabad, India, Mar. 2011.
- [130] J. Sankaranarayanan, H. Samet, B. Teitler, M. D. Lieberman, and J. Sperling. TwitterStand: News in tweets. In *GIS'09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 42–51, Seattle, WA, Nov. 2009.
- [131] C. Scanlan. *Reporting and Writing: Basics for the 21st Century*, chapter 5. Oxford University Press, 2000.
- [132] F. Schilder, Y. Versley, and C. Habel. Extracting spatial information: grounding, classifying and linking spatial expressions. In *GIR'04: Proceedings of the SIGIR 2004 Workshop on Geographic Information Retrieval*, Sheffield, UK, July 2004.
- [133] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 154–164, Manchester, UK, Sept. 1994.
- [134] M. J. Silva, B. Martins, M. Chaves, A. P. Afonso, and N. Cardoso. Adding geographic scopes to Web resources. *CEUS: Computers, Environment, and Urban Systems*, 30(4):378–399, July 2006.
- [135] D. A. Smith and G. Crane. Disambiguating geographic names in a historical digital library. In *ECDL'01: Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, pages 127–136, Darmstadt, Germany, Sept. 2001.
- [136] N. V. Sobhana, A. Barua, M. Das, P. Mitra, and S. K. Ghosh. Co-occurrence based place name disambiguation and its application to retrieval of geological text. In *Proceedings of NeCoM 2010, WiMoN 2010, and WeST 2010*, pages 543–552, Chennai, India, July 2010.
- [137] Social Security Administration. Popular baby names. URL <http://www.ssa.gov/OACT/babynames>. Accessed 08 Oct 2010.
- [138] J. Squires. Fire chief: Boy, 4, sparked fire that gutted Boulder Creek house. URL [http://www.santacruzsentinel.com/ci\\_13867925](http://www.santacruzsentinel.com/ci_13867925). Accessed 16 Jan 2010.



- [139] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical Report 00-034, University of Minnesota, Minneapolis, MN, May 2000.
- [140] N. Stokes, Y. Li, A. Moffat, and J. Rong. An empirical study of the effects of NLP components on Geographic IR performance. *IJGIS: International Journal of Geographical Information Science*, 22(3):247–264, Mar. 2008.
- [141] J. Strötgen, M. Gertz, and P. Popov. Extraction and exploration of spatio-temporal information in documents. In *GIR'10: Proceedings of the 6th Workshop on Geographic Information Retrieval*, Zurich, Switzerland, Feb. 2010.
- [142] F. Supek. fast-random-forest: An efficient implementation of the Random Forest classifier for Java. URL <http://code.google.com/p/fast-random-forest>. Accessed 27 Mar 2012.
- [143] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A new view on news. In *GIS'08: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 144–153, Irvine, CA, Nov. 2008.
- [144] B. E. Teitler, J. Sankaranarayanan, and H. Samet. Online document clustering using the GPU. Technical Report CS-TR-4970, University of Maryland, College Park, MD, Aug. 2010.
- [145] The Hindu. ‘Kites’ soaring high. URL <http://www.thehindu.com/mp/2009/11/28/stories/2009112853180400.htm>. Accessed 16 Jan 2010.
- [146] The Paris News. Have a Grand time, help restore the marquee. URL <http://theparisnews.com/story.lasso?ewcd=09b627849e99d1fa>. Accessed 07 Jun 2010.
- [147] Thomson Reuters. OpenCalais. URL <http://opencalais.com>. Accessed 30 Dec 2010.
- [148] R. Tobin, C. Grover, K. Byrne, J. Reid, and J. Walsh. Evaluation of georeferencing. In *GIR'10: Proceedings of the 6th Workshop on Geographic Information Retrieval*, Zurich, Switzerland, Feb. 2010.
- [149] H. Tolentino and R. Kamadjeu. Epispider ai mashup 2.0. URL <http://www.epispider.org>. Accessed 27 Mar 2012.
- [150] H. Tolentino, R. Kamadjeu, P. Fontelo, F. Liu, M. Matters, M. Pollack, and L. Madoff. Scanning the emerging infectious diseases horizon - visualizing ProMED emails using EpiSPIDER. *Advances in Disease Surveillance*, 2(4):169, 2007.
- [151] UPI. Serena, Jankovic win at French Open. URL [http://www.upi.com/Sports\\_News/2009/05/30/Serena-Jankovic-win-at-French-Open/UPI-42361243724411](http://www.upi.com/Sports_News/2009/05/30/Serena-Jankovic-win-at-French-Open/UPI-42361243724411). Accessed 05 Jan 2011.

- [152] U.S. Geological Survey. BGN: Domestic names. URL <http://geonames.usgs.gov/domestic>. Accessed 25 Aug 2010.
- [153] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
- [154] R. Volz, J. Kleb, and W. Mueller. Towards ontology-based disambiguation of geographical identifiers. In *I3'07: Proceedings of the WWW 2007 Workshop on I3: Identity, Identifiers, Identification, Entity-Centric Approaches to Information and Knowledge Management on the Web*, Banff, Canada, May 2007.
- [155] C. Wang, X. Xie, L. Wang, Y. Lu, and W.-Y. Ma. Web resource geographic location classification and detection. In *WWW'05: Proceedings of the 14th International World Wide Web Conference*, pages 1138–1139, Chiba, Japan, May 2005. Poster session.
- [156] A. Weichselbraun. A utility centered approach for evaluating and optimizing geo-tagging. In *KDIR'09: Proceedings of the 1st International Conference on Knowledge Discovery and Information Retrieval*, Madeira, Portugal, Oct. 2009.
- [157] Wikipedia. List of infectious diseases. URL [http://en.wikipedia.org/wiki/List\\_of\\_infectious\\_diseases](http://en.wikipedia.org/wiki/List_of_infectious_diseases). Accessed 06 Oct 2010.
- [158] R. H. Wilson and M. Guajardo. Capacity building and governance in El Cenizo. *Cityscape*, 5(1):101–123, 2000.
- [159] B. P. Wing and J. Baldridge. Simple supervised document geolocation with geodesic grids. In *ACL'11: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 955–964, Portland, OR, June 2011.
- [160] A. G. Woodruff and C. Plaunt. GIPSY: Automated geographic indexing of text documents. *JASIS: Journal of the American Society for Information Science*, 45(9): 645–655, Oct. 1994.
- [161] E. Wyatt. F.C.C. likely to open new airwaves to wireless. URL <http://www.nytimes.com/2010/09/13/technology/13wifi.html>. Accessed 12 Jan 2012.
- [162] Yahoo! Yahoo! Placemaker, . URL <http://developer.yahoo.com/geo/placemaker>. Accessed 26 Aug 2010.
- [163] Yahoo! The top news headlines on current events from Yahoo! News, . URL <http://news.yahoo.com>. Accessed 26 Aug 2010.
- [164] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL'95: Proceedings of the 33rd annual meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA, June 1994.
- [165] N. Yasuda, T. Hirao, J. Suzuki, and H. Isozaki. Identifying bloggers' residential areas. In *Proceedings of the 2006 AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*, Palo Alto, CA, Mar. 2006.

- [166] P. Yin, M. Zhang, Z. Deng, and D. Yang. Metadata extraction from bibliographies using bigram HMM. In *ICADL'04: Proceedings of the 7th International Conference of Asian Digital Libraries*, pages 310–319, Shanghai, China, Dec. 2004.
- [167] W. Zong, D. Wu, A. Sun, E.-P. Lim, and D. H.-L. Goh. On assigning place names to geography related web pages. In *JCDL'05: Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries*, pages 354–362, Denver, CO, June 2005.