News Analysis for the Social Sciences

A DISSERTATION PRESENTED

BY

MIKHAIL BAUTIN

TO

THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

STONY BROOK UNIVERSITY

August 2009

Stony Brook University

The Graduate School

Mikhail Bautin

We, the dissertation committee for the above candidate for
the Doctor of Philosophy degree,
hereby recommend acceptance of this dissertation.

Professor Steven Skiena, Advisor
Computer Science Department

Professor Amanda Stent, Chairperson of Defense
Computer Science Department

Professor I. V. Ramakrishnan
Computer Science Department

Professor Matthew Lebo
Department of Political Science

This dissertation is accepted by the Graduate School.

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

News Analysis for the Social Sciences

by

Mikhail Bautin

Doctor of Philosophy

in

Computer Science

Stony Brook University

2009

Advisor: Professor Steven Skiena

The Lydia system analyzes spatial, temporal, and linguistic statistics of named entity occurrences in news text. As a result, it provides the user with a bird's-eye view of the media coverage for a specific entity. This system can be a valuable tool for social sciences such as political science, history, and economics. Our time series data on the evolution of associations of an entity with other entities and traits is helpful in testing hypotheses in political science and history. Our maps showing the spatial distribution of entity popularity, sentiment polarity, and subjectivity towards that entity in the country are useful in reasoning about the level of success of a political campaign. Sentiment time series can be incorporated into studies of media influence and used to study voter behavior in response to local or national media coverage.

Processing large-volume historical news datasets is computationally intensive. To avoid the bottleneck (which our legacy Lydia system had) of a single relational database and to dramatically increase the scale of our news analysis, we have designed a new data aggregation and processing architecture that encompasses components such as text and derived statistics processing phases, an on-demand data retrieval server, a user interface web application, and experimental components aimed at validation and improvement of individual analysis phases.

This new Lydia architecture (code-named "Freedonia") which is based on the Hadoop open-source map-reduce framework, is fully scalable and capable of analyzing statistics on over 74 million entities in more than 100 million U.S. daily news articles in a few hours on a 18-node cluster. It provides a 10-20x performance improvement over the old Lydia system even on datasets that the old system can still scale up to. The new Lydia architecture contains scalable versions of duplicate article detection, entity sentiment time series calculation, cross-document co-referential entity name identification, and statistic aggregation across groups of co-referential entities and other types of groups.

The data is accessible to social scientists through a web interface, and advanced users can access our data services programmatically through an appropriate API. The new Lydia system was used to provide media coverage data of the 2008 presidential election for the National Annenberg Election Survey, the largest academic public opinion survey conducted during presidential elections.

With the help of the new Lydia system we study the differences in news coverage of various cultural/ethnic/linguistic groups in the U.S. news. In particular, we examine the frequency and sentiment volume time series of persons of various ethnicities and explore the geographic biases of ethnic group coverage.

Finally, we include two studies aimed at the validation and improvement of different aspects of our named entity-centered text analysis system. Firstly, we generalize the Lydia sentiment analysis approach to languages other than English. We utilize state-of-the-art machine translation technology and perform sentiment analysis on the English translation of foreign language text. Our experiments indicate that entity sentiment scores obtained by our method are significantly correlated across nine languages of news sources and five languages of a parallel corpus and can be used to perform meaningful cross-cultural comparisons.

Secondly, we consider the problem of finding the relevant named entities in a text corpus in response to a free-text search query. We analyze the AOL search query logs to assess the significance of this problem. Then, we describe and evaluate our implementation of a concordance-based entity search engine retrieving entity results based on entity occurrence contexts in the corpus.

Contents

List of Tables

List of Figures

Acknowledgments

Firstly I would like to thank Steven Skiena, my advisor. Without his guidance, understanding, and encouragement much of this work would never have been completed.

I thank Charles Ward and Dmytro Molkov for all the good times I have had working closely with them on the core of our new system during the past year, and all the interesting discussions we have had. I thank Akshay Patil for his tremendous effort developing our new user interface and for a walking tour of Philadelphia.

I thank all other Applied Algorithms / News and Blog Analysis Lab students I got to know over the years, including Levon Lloyd, Dimitris Papamichail, Andrew Mehler, Mohammad Sajjad Hossain, Manjunath Srinivasaiah, Namrata Godbole, Prachi Kaulgud, Jae Hong Kil, Alex Turner, Ashwin Subrahmanya, Anand Mallangada, Jai Balani, Sandesh Devaraju, Sushma Devendrappa, Lohit Vijayarenu, Gayathri Ravichandran, Paavan Shanbhag, Jahangir Mohammed, Swapna Reddy, Anurag Ambekar, Shrikant Shanbhag, Shashank Naik, and Sagar Pilania. I wish John Rizzo, Karthik Balaji, Shrijeet Paliwaal, Girish Kathalagiri, and Wenbin Zhang much success in carrying on with the project.

I thank the General Sentiment team for their effort in commercializing the TextMap technology and the National Annenberg Election Survey team for their valuable feedback on our system usability. I also thank Abe Ittycheriah and Salim Roukos of IBM Research for their help with IBM WebSphere Translation Server.

Most importantly, I would like to thank my parents, relatives, and friends, whose support and advice kept me going forward even during the most challenging times of my studies.

The work in this thesis was supported with funding from The Research Foundation of SUNY, Renaissance Fellowship, and National Science Foundation.

Chapter 1

Introduction

The Lydia system [17,19,47,56,61–63,66] analyzes spatial and temporal statistics of named entity occurrences in news text. As a result, it provides the user with a bird's-eye view of the media coverage of a specific entity. This system can be a valuable tool for social science research, in fields such as political science, history, and economics. Our time series data on the evolution of associations of an entity with other entities and traits is helpful in testing hypotheses in political science and history. Our maps showing the spatial distribution of entity popularity, sentiment polarity, and subjectivity towards that entity in the country are useful in reasoning about the level of success of a political campaign. Sentiment time series can be incorporated into studies of media influence and used to study voter behavior in response to local or national media coverage.

I have worked on several projects related to news and text analysis starting from Spring 2006, and have co-authored the following publications: [17] and the corresponding journal article [18], [19], and [92] [1]. But my largest project to date and the main topic of this thesis is the new scalable Lydia architecture codenamed "Freedonia", which is described in detail in Chapters 2 and 3.

Making Lydia viable as a source of high-quality scientific data to feed into social science models requires substantial technical effort. Instead of trying to re-engineer Lydia's four-year-old code written by several generations of graduate students and work around its performance bottlenecks, we decided to build a new scalable text

[1]Another publication that I co-authored at Stony Brook University, unrelated to the Lydia project, is [16].

processing engine utilizing modern distributed computation technologies such as the Hadoop [11] implementation of the Map-Reduce [39] model. My role in this process included the design and implementation of our new data aggregation and processing architecture, providing a framework for interactions between various components of the new system such as text and derived statistics processing phases. I also developed an on-demand data retrieval server and performed experiments aimed at improvement of individual analysis phases. Finally, I was responsible for establishing and upholding code quality, design, implementation, and testing standards for our new system.

The design of this distributed system is significantly more than a mere re-implementation of the previous Lydia system. Our new system is a data mining platform prepared to deal with a two-order of magnitude greater text volume and offers an extensible data extraction framework that will make many new exciting types of study possible. One such study is given in Chapter 4, which examines the differences between the news coverage of various cultural/ethnic groups.

The new Lydia system was brought online and up-to-date with the news in March 2009, and is now tracking about 1000 U.S. and foreign English-language newspapers on a daily basis. This system is also being successfully used to analyze the news coverage data of the 2008 presidential campaign for the National Annenberg Election Survey, the largest academic election survey in the country.

1.1 Lydia System Overview

The Lydia system [17, 19, 47, 56, 61–63, 66] recognizes named entities in text and extracts their temporal and spatial distributions. Text sources are spidered daily by customized website scrapers that convert articles to a standard format and store them in an archive. The articles are then run through a pipeline that performs part-of-speech tagging, named entity identification and categorization, geographic normalization, intradocument coreference resolution, extraction of entity descriptions and relations between entities, and per-occurrence sentiment score calculation. An example of Lydia NLP markup is given in Figure 1.1. The entities are then inserted into a database, and cross-document coreference resolution, entity juxtaposition

Hong Kong's government ordered Wednesday that all kindergartens and primary schools be closed for two weeks amid a flu outbreak, shutting down classes for more than a half million students.

$\longrightarrow$

<p> <pn category = "CITY"> Hong/NT Kong,/NT HKG/NT </pn> 's/POS government/NN ordered/VBN <embedded_date> Wednesday/NNP </embedded_date> that/IN all/DT kindergartens/NNS and/CC primary/JJ schools/NNS be/VB closed/VBN for/IN <pn category = "WEEK_PERIOD"> 2/CD weeks/NNS </pn> amid/IN a/DT <pn category = "DISEASE"> flu/NN </pn> outbreak/NN ,/, shutting/VBG down/IN classes/NNS for/IN more/RBR than/IN a/DT half/NN <num type = "CARDINAL"> 1000000/CD </num> students/NNS ./. </p>

Figure 1.1: An example of Lydia NLP markup for one paragraph.

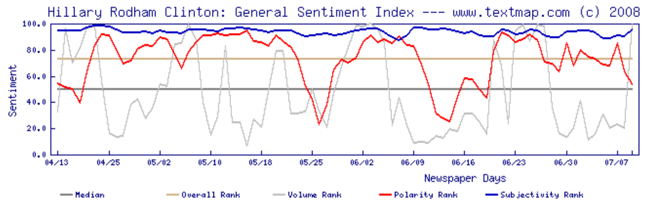


Figure 1.2: Sentiment time series for Hillary Clinton.

scoring, and per-entity sentiment score calculation take place. The resulting statistics are presented visually at `http://www.textmap.com` to a general internet audience.

In the following sections we give two examples of components of the Lydia system that are applicable to the social sciences.

1.1.1 Sentiment Analysis

Sentiment analysis is the area of natural language processing that aims to determine positive and negative opinions expressed in text. The Lydia system is capable of determining the sentiment orientation of individual references to entities in news text and aggregating them into *sentiment score* time series. Quantifying opinion as expressed in blogs and newspapers opens opportunities for studying media influence on voters, so sentiment analysis is particularly important to us in our collaboration with political scientists. The Lydia sentiment analysis system assigns scores indicative of positive or negative opinion to each distinct entity in a text corpus [19, 47]. Positive and negative sentiment words are identified through navigating WordNet [69] synonym and antonym links starting from specific seed sets corresponding to such categories as general, business, crime, health, politics, sports and media sentiment. Sentiment analysis can be extended beyond people to include events, terminology, and other inanimate objects (e.g. roadside bombs). Our sentiment analysis methods prove surprisingly effective, even though they rely on relatively crude natural language processing techniques. Sentiment analysis has become a large area with a substantial literature [49, 75, 96, 97, 100].

Our results described in Chapter 5 and [19] show that the state-of-the-art in machine translation is capable of capturing accurate notions of sentiment when aggregated over many documents. We have set up spiders and translators for hundreds of newspapers in Spanish, French, German, Italian, Chinese, Japanese, Korean, and Arabic. Through comparative statistical analysis, we can accurately identify international biases and trends. This study was done using the previous version of Lydia [62]. Our new scalable Lydia architecture creates new possibilities for evaluating this international sentiment analysis methodology.

1.1.2 Entity Juxtapositions

A lot of information about an entity can be discerned from the list of other entities it is related to. The Lydia system provides a method for extracting pairs of entities that occur close to each other in an overrepresented way in a text corpus. Suppose n_a and n_b are numbers of sentences containing entities a and b respectively, F is the number of sentences containing both a and b, and N is the total number of sentences in the corpus. Then, according to [62], the probability of the observed number of occurrences under the assumption that these two entities are independent is not more than

$$P_{bound}(n_a, n_b, F, N) = \left(\frac{e^{\frac{FN}{n_a n_b} - 1}}{\left(\frac{FN}{n_a n_b}\right)^{\frac{FN}{n_a n_b}}} \right)^{\frac{n_a n_b}{N}} \tag{1.1}$$

We call $-\log P_{bound}$ the *juxtaposition score* of entities a and b.

The top juxtapositions for an entity are often unmistakably related to the entity itself. For example, the top juxtapositions for John McCain in the U.S. news over the year of 2008 include Barack Obama, Republican, Mitt Romney, Sarah Palin, GOP, and Bush. More interesting observations can be drawn from changes in the set of juxtapositions of an entity. For example, the top juxtapositions for Eliot Spitzer calculated over March 2008 indicate entities suggestive of the scandal that led to his resignation, such as Joe Francis, Emperors Club VIP, and "Client 9". At the same time, the juxtapositions calculated from all history available to us list politicians and businessmen as Eliot Spitzer's top juxtapositions. Thus, by observing changes in the set of top juxtapositions of an entity, we can detect events happening to that entity and bring them to the user's attention. Studying how the relationship strength between a specific pair of entities develops over time can also provide insight to political scientists interested in their interaction. Both of these capabilities are implemented on a large scale in the new Lydia data extraction and retrieval architecture.

1.2 Thesis Overview

The remainder of this thesis is organized as follows. Chapter 2 covers our text analysis system at a conceptual level and discusses its social science applications. We survey previous systems that have been developed to aid social science research. Then, we describe the features our system provides that might be of interest to social scientists, such as entity popularity and sentiment time series, juxtapositions, and full-text article search. We explain how these types of data are obtained from the unstructured text. Finally, we provide a use case of our system's user interface.

In Chapter 3 we go through the technical challenges we have encountered while building our scalable text stream analysis infrastructure in more detail. We discuss previous systems relevant to processing large amounts of text on a computer cluster. We describe the data organization and workflow management framework we have developed to simplify building and updating our statistical datasets derived from unstructured text. We also go through the details of certain analysis phases built on top of this framework, such as duplicate removal, entity co-reference resolution and group statistic aggregation. Finally, we present performance evaluation results.

In Chapter 4 we provide a case study of applying our system to a social science question: what are the differences between the news coverage of various cultural and ethnic groups? We describe a new method for nationality detection for news entities (people), and use it in conjunction with the name ethnicity classification method from [5] to identify interesting temporal, geospatial, and association trends in the news with respect to 13 distinct cultural/ethnic/linguistic (CEL) groups. In particular, we examine issues of ethnic and geocentric sentiment/coverage bias in newspapers.

In Chapter 5 we validate another essential component of our text analysis tool for social scientists: the sentiment scoring subsystem. Our experiments indicate that (a) entity sentiment scores obtained by our method are statistically significantly correlated across nine languages of news sources and five languages of a parallel corpus; (b) the quality of our sentiment analysis method is largely translator independent; (c) after applying certain normalization techniques, our entity sentiment scores can be used to perform meaningful cross-cultural comparisons.

Chapter 6 deals with a problem highly relevant to building a data mining system centered around named entities: finding a named entity via a free-text query based on an archive of unstructured text. Our entity search engine creates a concordance document for each entity consisting of all the sentences in the corpus containing that entity. We evaluate our system by comparing the results of each query to the list of entities that have the highest juxtaposition scores with the queried entity. The results show excellent performance, particularly over well-characterized classes of entities such as people.

Chapter 7 presents concluding remarks and future directions for this research.

Chapter 2

News and Blog Analysis for the Social Sciences

2.1 Introduction

The Lydia system [19, 47, 56, 62, 63, 66] analyzes spatial, temporal, and linguistic statistics of named entity occurrences in text corpora. It provides the user with a view of the media coverage of a specific entity. Once a text corpus has been processed through our system, the user can examine data centered around any named entity, either automatically recognized or user-defined.

Figure 2.1 is an example of the types of analyses Lydia makes possible. It shows sentiment subjectivity and polarity score graphs for the entity *World Trade Center* generated by our system from the New York Times 1981-2007 corpus. Subjectivity reflects the volume of strongly positive or negative words associated with the entity and polarity reflects the difference between positive and negative shares of those words. It is clear from Figure 2.1 that after both the 1993 World Trade Center bombings and the September 11th, 2001 attacks the news coverage of *World Trade Center* became much more subjective with prevalently negative sentiment.

As another example, Table 2.1 shows the top entities juxtaposed with Barack Obama over two four-month periods from May to December 2008. The entities

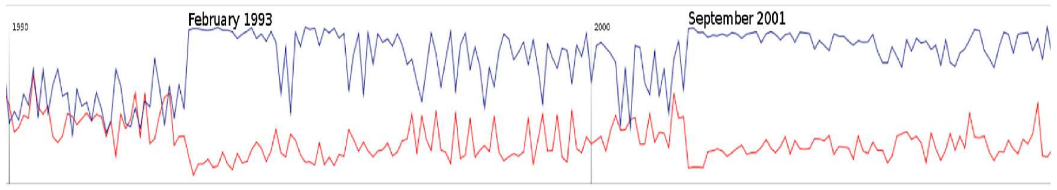


Figure 2.1: Sentiment subjectivity (top/blue) and polarity (bottom/red) score for the World Trade Center.

	May, Jun 2008		Jul, Aug 2008		Sep, Oct 2008		Nov, Dec 2008	
Rank	**Entity Name**	**Count**	**Entity Name**	**Count**	**Entity Name**	**Count**	**Entity Name**	**Count**
1	Hillary Clinton	313316	John McCain	358676	John McCain	563000	Democrat	201034
2	Democrat	346028	Democrat	330643	Democrat	342109	John McCain	178190
3	John McCain	238767	presidential	169928	Republican	176279	election	121648
4	presidential	150699	candidate	127159	presidential	173176	Republican	92358
5	Republican	105005	Republican	123940	candidate	120136	presidential	82209
6	candidate	98933	Hillary Clinton	103319	Sarah Palin	112729	White House	67026
7	senator	81783	Iraq	74484	voters	86988	senate	57876
8	primary	75139	voters	65255	debate	80073	Hillary Clinton	55913
9	voters	75124	Joe Biden	64957	Joe Biden	70325	voters	55145
10	superdelegate	50045	senator	64285	senator	51129	senator	32575

Table 2.1: Top juxtapositions for Barack Obama for four two-month periods and the corresponding co-occurrence counts.

used for this experiment were taken from the list of phrases of interest to the National Annenberg Election Survey. The counts in Table 2.1 are numbers of sentences in which Barack Obama appeared with the respective entity from the table. Entities having the highest juxtaposition frequencies with a given entity can be thought of as those "most talked about with" that entity. From Table 2.1 we notice that Hillary Clinton was the most associated entity with Barack Obama until she left the presidential race in June 2008, after which John McCain became his top juxtaposition. We also notice that vice-presidential candidates Joe Biden and Sarah Palin appear in Barack Obama's top juxtaposition list around the time they are announced. The word "debate" ranks fifth in the period immediately preceding the election. Finally, the White House ranks sixth on Barack Obama's top juxtaposition list in the post-election November-December 2008 period, and his top juxtaposition becomes "Democrat" instead of John McCain once the presidential race is over.

Both of these types of studies (sentiment time series for a large historical corpus and time-based juxtaposition analysis for a custom entity set) are examples of what has only become possible with our new scalable Lydia architecture, compared to the previous version of the Lydia system described in [62]. The performance improvement is approximately 20-fold. The new Lydia system can fully process our four-year archive of over 100 million U.S. articles in less than six weeks on our 18-node Hadoop [11] cluster. The old Lydia ran on a single machine, but even if it could scale linearly, it would take 2.5 years to process the same dataset, according to the 250 articles per hour per machine performance estimate reported in [60].

The main contribution of this work is an innovative news analysis system designed to provide social scientists with an in-depth picture of the news coverage of a named entity or a group of entities. The new version of the Lydia system was developed as the news analysis engine for the National Annenberg Election Survey, but it is applicable to a wider range of studies in political science, history, and global studies.

The remainder of this chapter is organized as follows. Section 2.2 examines related text analysis systems developed for social scientists, as well as tools used in implementation of our system. Section 2.3 discusses the motivating social science applications of our system. In Section 2.4 we describe the processing phases our system takes to produce an archive of entity statistics from raw text. Section 2.5 describes the internal organization of our statistics archive. Section 2.6 covers the algorithms and data structures we use to efficiently collect entity statistics. Section 2.7 describes our approach to searching article text including some non-standard features such as searching by sentiment. In Section 2.8 we outline the features of our system's web frontend and give a use case. Section 2.9 describes the dataset we analyzed for the National Annenberg Election Survey, and Section 2.10 concludes the chapter.

2.2 Related Work

The related work can be separated into text analysis systems useful in social sciences, such as Cornell's Web Lab Collaboration Server [93], TABARI [85], and LexisNexis Academic [59], tools useful in implementation of such systems, such as Hadoop [11], and studies that have been already done using the data provided by the new Lydia system.

2.2.1 Text Analysis Systems for the Social Sciences

This section examines existing systems for text analysis in the social sciences, largely different than Lydia in their structure and capabilities.

2.2.1.1 Large-scale web data analysis for the social sciences

Weigel et al. [93] propose a platform called "Web Lab Collaboration Server" to simplify large-scale web data analysis tasks for non-technical users. They try to automate tasks such as extraction of structured datasets, cleaning and formatting them. They expose a high-level interface to the user, which allows to construct extraction and analysis workflow through an intuitive GUI. The user is given primitives to construct an analysis workflow from: set operations, relational algebra, shallow text analysis (word count, TF-IDF), and simple graph algorithms. A user-created analysis task is converted to a logical algebra language expression, which is then compiled into map-reduce Java code and executed on the cluster. The authors use the Internet Archive data in their experiments.

The differences between Web Lab Collaboration Server and our scalable Lydia architecture are as follows. Firstly, Web Lab Collaboration Server is designed for analyzing somewhat structured data, such as social network graphs or online bookstore user pages, while Lydia is centered around named-entity analysis in unstructured text. Secondly, Web Lab Collaboration Server lets users create their own analysis workflows, while the Lydia system provides a pre-existing workflow producing a wide variety of statistics. And finally, Web Lab Collaboration Server requires users to create and initiate analysis tasks, whereas Lydia does all its large-scale analysis in the background and provides pre-computed entity statistics updated every day.

2.2.1.2 Automated coding of news sources

A substantial amount of work has been done in the area of automated "coding" of news sources, a task traditionally performed by undergraduate and graduate students in political science departments. News text “coding” involves finding and marking up events in news text involving pre-defined entities, such as countries or politicians. The best known systems of this kind, KEDS (Kansas Events Data System) and its successor TABARI (Textual Analysis by Augmented Replacement Instructions) [85], developed at the University of Kansas, use a sparse parsing approach to extract event data from news text. To parse a sentence, the system marks up nouns and verb phrases in it and attempts to match them with a dictionary. TABARI is driven by a list of manually created dictionaries containing proper nouns (actors), common nouns (agents)—such as “French”, verbs/verb phrases, and pronouns. Their political event coding scheme follows that of the World Event/Interaction Survey (WEIS) [65], which was later extended and superseded by the IDEA framework [23]. For example, the sentence “The United States and Egypt approved of efforts by Israel and Jordan” would result in the following TABARI output:

```
USA <APPROVED> ISR
USA <APPROVED> JOR
UAR <APPROVED> ISR
UAR <APPROVED> JOR
```

The TABARI and KEDS systems have been mostly used for international relations research primarily focused on the Middle East.

The TABARI system has been released as open source but does not contain any framework for data aggregation or parallel processing. The scale of data in studies based on TABARI such as [45] is on the order of hundreds of thousands of news reports, while our new Lydia architecture is capable of handling hundreds of millions of articles.

2.2.1.3 News search for the academia

LexisNexis Academic [59] searches news, business, and legal content and is available to researchers and students at academic institutions. Its text sources include newspapers, broadcast transcripts, blogs, SEC filings, company profiles, law reviews, case law, and statutes. LexisNexis Academic provides flexible search capabilities subdivided into two major search modes:

- Natural Language search. This provides traditional keyword search functionality, and is recommended for quick access to pertinent results on a given topic, especially when the information need is broad or the user is not sure what words to use to express the query.

- Terms and Connectors search. This provides Boolean search functionality and is used when the precise combination of words and phrases to be included or not included into the document of interest is known. It is possible to specify complex nested Boolean queries and restrict word distances between term occurrences.

These two types of search can be accessed using the "easy search" feature that automatically identifies which search method to use based on the type of the input phrase, or using the "power search" feature that allows for precise configuration of the search. The possible configuration options include date restrictions, index terms (document tags indicative of its topic, e.g. "Missile Defense Systems" or "International Relations"), source restrictions (major world publications, weblogs, etc.), and searching specific sections of the document (e.g. headline or body).

The flexibility of LexisNexis search options makes it popular within the social science community. However, it offers no dedicated features for exploring a text corpus's coverage of a named entity and provides no time series or maps visualization to quantify this coverage. This is where our Lydia system complements LexisNexis in a social scientist's arsenal of tools for looking at the media. The Lydia system also provides article search functionality, although more limited than that of LexisNexis, but with some unique features such as finding articles with positive or negative sentiment towards a given entity (see Section 2.7).

2.2.2 Implementation Tools

We use the Hadoop [11] implementation of Google's Map-Reduce [39] distributed computation model to simplify distributing the various computation phases of Lydia text processing across a cluster of computers. I presented a case study of Lydia as a Hadoop application at the Hadoop Summit 2009 in Santa Clara, CA.

2.3 News Analysis in the Social Sciences

We consider several social science fields in which our system will clearly be useful. It has also been shown in [101] that our system can be applied to finance.

2.3.1 Applications in Political Science

Political science is the field that stands to most obviously benefit from using our news analysis system, as it is primarily concerned with current events involving entities widely covered in the media. We have collaborated with political scientists from Stony Brook University and University of Pennsylvania. The general direction of this collaboration is studying the influence of media coverage on electorate opinion, and our system quantifies the media part of this connection. The subsections below outline some questions in political science where our data is likely to be useful.

2.3.1.1 Media effects on elections

It is not well understood how media coverage over the course of primaries and elections affects candidate popularity. According to [58] there are two good reasons for this lack of understanding:

- The speed with which electoral fortunes change over the course of a campaign.
- The lack of available data to measure these quickly changing events.

Public opinion surveys are now being collected often enough that we can study public opinion change on a daily basis. The National Annenberg Election Survey [81] started collecting individual-level survey data in early November 2007 and continued with daily samples of voters all the way until the Election Day 2008. We have worked in collaboration with them to provide the media coverage data for their analysis, and the NAES researchers are currently using the new Lydia system through its interactive web interface and exported databases.

The Lydia project offers the opportunity to closely examine the relationship between campaign events, public opinion, and the media. Many previous studies of media content such as [25, 26] have explored an extremely narrow range of news sources. With our new Lydia infrastructure we are now able to analyze roughly 1000 online news sources with an archive spanning four years, starting from November 2004, comprising over 50 million different articles. The Lydia system is able to navigate and slice the data over

- the spatial dimension (e.g. Iowa newspapers, New Hampshire newspapers or the United States as a whole), and
- the temporal dimension (on a daily, weekly, monthly, quarterly, and other scales).

Given those capabilities, the political scientists using our system in conjunction with public opinion poll data will be able to answer more easily questions such as:

- Do Iowa voters respond to sentiment expressed locally or nationally?
- Does daily media sentiment around the Iraq war influence public opinion on the matter?
- Did media coverage of the Iraq war change after the 2004 election when Republican support for the war began to wane?

2.3.1.2 Media fairness

The perception of media fairness is subjective: the supporters of a particular candidate are likely to attribute the suffering popularity of that candidate to negative media coverage but when things are going well, describe the media coverage of the candidate as fair [70]. For example, during the 2008 presidential campaign Sen. Obama's supporters' perceptions of media fairness declined as the controversy broke about comments made by his former pastor, as did Sen. Clinton's supporters' views of media fairness after her comments about her landing under sniper fire in Bosnia proved false. With our system's ability to objectively measure a candidate's sentiment in the news, it became possible to elucidate the influence of news coverage on a candidate's popularity and, through specific news story case studies, quantify press fairness in a manner less biased by pollees' preferences.

2.3.2 Applications in Global Studies

Scholars working in the field of global studies are concerned with how "global" a certain entity is. They have traditionally based their measures of globality on such attributes as population, capital/non-capital status, etc. Introducing an entity globality measure based on news text in the Lydia system will create a useful tool for global studies [84]. Defining traits of global entities include geographical range and a state of connectedness to a wide variety of other entities [83]. A possible globality measure could be based on:

- The importance of an entity on its own. A general approximation for this is entity frequency in the news.
- The geographical reach of the entities the given entity is connected to according to our juxtaposition score criterion [62]. Here we consider entities that have associated geographic information, such as cities, countries, or people for whom we can identify the country they belong to. This measure will differentiate between entities connected to a wide range of other entities pertaining to various countries or states, and entities local to a specific country or state. For instance, according to this measure we expect New York, NY to be a more global entity than San Antonio, TX, while Geneva, Switzerland should prove more global than Phoenix, AZ.
- The geographical span of the news sources in which the entity is mentioned. With our current data dominated by U.S. daily newspapers this will be mostly useful to distinguish entities of nation-wide importance from those of only local significance, but as we start to acquire larger archives of international (and possibly multi-language) news, this measure will more closely reflect entity globality as it is understood in the literature on the subject.

Entity globality measurement using the Lydia system is an ongoing collaboration between our research group and the Center for Global and Local History at Stony Brook University.

2.3.3 Applications in History

Global history is a recent field of historiography with a plethora of unsolved and complex problems such as periodization and measurement of globality [82]. Extending the concepts of Section 2.3.2 to the time domain, the Lydia system is also likely to be useful in tracking the history of how the globality of an entity changes over time. This is highly relevant to the problem of periodization of global history, especially of its most recent part for which there exist timed text datasets suitable for analysis by our system. By applying our system to the New York Times archive starting from 1851 which has recently been made available, we can extract interesting and subtle trends of historical entity globality variations. Another interesting question one could try to answer in a historical perspective using the Lydia system is how the overall sentiment of the relationship between two countries changes over time.

2.3.4 Applications in Sociology

Chapter 4 (also appearing as [92]) is a study of news coverage of cultural/ethnic/linguistic (CEL) groups and their interactions using the data obtained from the new Lydia system. It proposes a method for entity nationality detection using juxtaposition data, performs geographic news analysis of cultural groups, examines time series trends in CEL group frequency and sentiment, and quantifies interactions and sentiment between these groups.

2.3.5 Applications in Finance

Zhang and Skiena [101] study how company frequency and sentiment data obtained from the new Lydia system reflects the company's stock trading volumes and financial returns. They confirm that the news data is highly informative and propose a news-based market-neutral trading strategy which gives consistently favorable low volatility results over a four-year period.

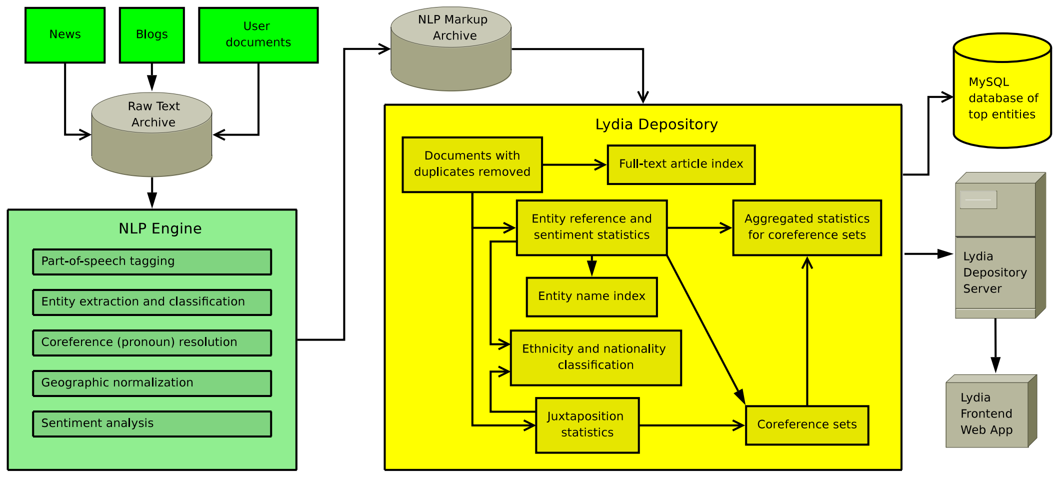


Figure 2.2: High-level Lydia architecture diagram

2.4 Processing Flow

Figure 2.2 shows a high-level layout of the new Lydia system. The basic processing flow can be subdivided into the following parts:

- *Input document collection* – Source documents may come from Lydia news or blog spiders, RSS feeds, or users who need to process a specific corpus of documents.
- *NLP pipeline* – Documents are passed through the Lydia NLP pipeline [60, 62], which marks up and classifies named entities, resolves pronouns, normalizes geographical location names, and identifies sentiment polarity of each entity occurrence. In the current configuration, the NLP pipeline runs as a Hadoop [11] Streaming job, and its output gets stored in the HDFS (Hadoop Distributed File System). This text corpus with named entity markup is the main input for our new scalable text analysis architecture.
- *Depository construction or update* – Documents are processed through a series of jobs (most often map-reduce jobs) and the results are stored in a persistent data structure that we call a *depository*. A Lydia depository includes:
  - All input documents, with duplicates removed;

  - Entity reference statistics;
  - Entity juxtaposition statistics;
  - A full-text index of articles;
  - An index of entity names;
  - Co-referential groups of entities;
  - Entity classification by ethnicity and country association;
  - Entity and juxtaposition statistics aggregated over co-reference sets and ethnic groups.

- *Database export* – We still use a relational database to provide more flexibility of data exploration, but we only export statistics for the most frequent entities into the database.

2.5 Data Organization

The statistics and indexes extracted from a text corpus are placed in a persistent data structure we call a *depository*, which is stored in a directory hierarchy in Hadoop Distributed File System. A depository comprises a number of named homogeneous datasets, such as entity references, juxtaposition statistics, or entity/article indexes. We call these datasets, typically produced with one or more map-reduce jobs, *artifacts*. Each of the interconnected items shown within the "Lydia Depository" part of Figure 2.2 corresponds to an artifact or a group of artifacts.

2.5.1 Artifact Dependencies

Only a few artifacts in the Lydia depository are built based directly on the marked-up text that comes out of our NLP pipeline. Most artifacts are built using other artifacts as inputs. Figure 2.3 shows some dependencies between artifacts containing article, entity, and juxtaposition statistics in the Lydia depository.

Most artifacts in the Lydia depository are constructed using map-reduce jobs. Every block in Figure 2.3 corresponds to an artifact, and each of them, except the source, has an associated type of map-reduce job used to build it. The input of a

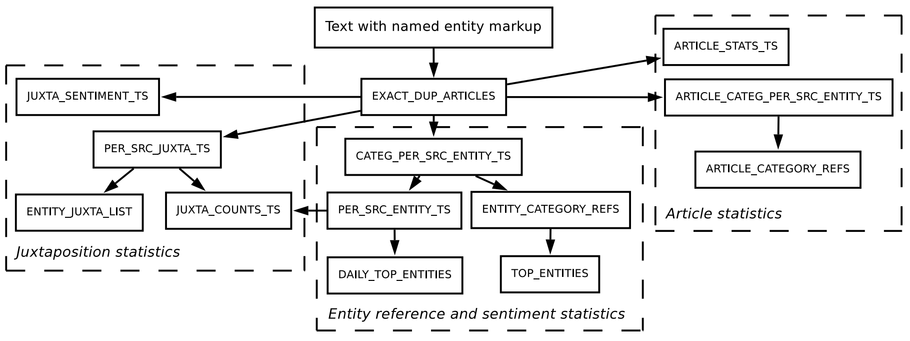


Figure 2.3: Artifact dependencies in the entity statistics folder of the Lydia depository.

map-reduce job that builds or updates an artifact is selected as described in Section 3.5.2.3.

The most important artifacts in the Lydia depository are listed below. We also include the size of each artifact in the U.S. daily news depository. In depositories built from other corpora the same artifacts can be appropriately larger or smaller.

- Entity reference and sentiment statistics.
  - EXACT_DUP_ARTICLES (177 GB) stores a text corpus from which the depository was built with exact duplicate articles removed. It is a key-value artifact with key being a normalized article text hash code and value being an article XML document including metadata about all original sources and dates where this article appeared. All text is compressed with gzip. Our approach to removing exact duplicate articles is described in Section 3.5.1.
  - CATEG_PER_SRC_ENTITY_TS (8.3 GB) is a key-value artifact mapping entity names to per-category per-source time series of their occurrence frequencies and sentiment scores. The data structure used for storing these multi-level time series is discussed in Section 2.6.1. This artifact can be used to retrieve time series of occurrence or sentiment for a particular entity with a certain categorization, e.g. person, in a given source or group of sources.
  - PER_SRC_ENTITY_TS (8.2 GB) is a key-value artifact mapping

entity names to per-source frequency/sentiment time series. Unlike CATEG_PER_SRC_ENTITY_TS, it does not distinguish entities by category, e.g. it would not distinguish between "Jaguar" as an animal and "Jaguar" as a car. It is calculated based on CATEG_PER_SRC_ENTITY_TS by summing up its data across all categories for each entity.

- ENTITY_CATEGORY_REFS (1.9 GB) provides a different condensed view of the categorized per-source time series artifact CATEG_PER_SRC_ENTITY_TS. It maps entity names to vectors of their frequencies of appearances as various categories. For example, for the entity "New York" it would contain the frequencies of this entity occurrences categorized as a city and as a state.
- TOP_ENTITIES (1 GB) contains entities in a descending order of their frequency of occurrence in the whole text corpus. It is an example of a "sequence artifact" which does not allow random access, but only retrieval of the top N items.
- DAILY_TOP_ENTITIES (10 MB) maps a date to a list of the top-frequency entities for that date. It is calculated based on the artifact containing per-source time series for each entity. By default no more than 1000 top entities are stored for each day, which explains why this artifact is much smaller than TOP_ENTITIES.

• Juxtaposition statistics.

  - PER_SRC_JUXTA_TS (9.2 GB) is a key-value artifact that maps an entity pair (E_1, E_2) to a per-source time series of the juxtaposition frequency of this pair of entities. Due to the high number of entity juxtapositions in text corpora, on large corpora we use a randomly-selected subset of sentences to construct these time series. However, this method of reducing the juxtaposition dataset volume sometimes results in a loss of meaningful juxtapositions for low-frequency entities.
  - JUXTA_COUNTS_TS (11.2 GB) is a key-value artifact that for each pair of juxtaposed entities (E_1, E_2) contains per-source time series of

tuples (n_1, n_2, n_{12}) where n_1, n_2, and n_{12} are the numbers of sentences in which E_1, E_2, and both E_1 and E_2 occurred, respectively. It is computed using both PER_SRC_JUXTA_TS and PER_SRC_ENTITY_TS by performing a two-map-reduce job join operation as described in Section 2.6.2. It is used to calculate juxtaposition significance statistics that depend both on the joint occurrence frequency and individual entity occurrence frequencies, such as our juxtaposition score measure (Section 2.1).

  - JUXTA_SENTIMENT_TS (13.7 GB) maps entity pairs to per-source time series of sentiment scores of the entity pair juxtaposition. The juxtaposition sentiment score is calculated based on the sentiment words that occur in the same sentence with the pair of entities.
  - ENTITY_JUXTA_LIST (5.5 GB) maps each entity to a list of top-frequency juxtapositions for that entity and the associated juxtaposition frequencies. It is widely used in our system, particularly in entity co-reference resolution (Section 3.7) and entity geographical association detection (Section 4.3).

- Article statistics.
  - ARTICLE_STATS_TS (6 MB) is a key-value artifact that maps a text source ID to a time series of the number of articles, number of sentences, and number of entity occurrences. It is a summary statistic of news volume in a given source or group of sources on a given time period, which we use for various kinds of time series normalization.
  - ARTICLE_CATEG_PER_SRC_ENTITY_TS (9 GB) is a key-value artifact similar to CATEG_PER_SRC_ENTITY_TS, but uses article categories instead of entity categories. Its values are multi-level maps keyed on article category (e.g. sports, business, entertainment), text source ID, and date. It allows to trace transitions of an entity from one category to another, as shown in Figure 2.4.

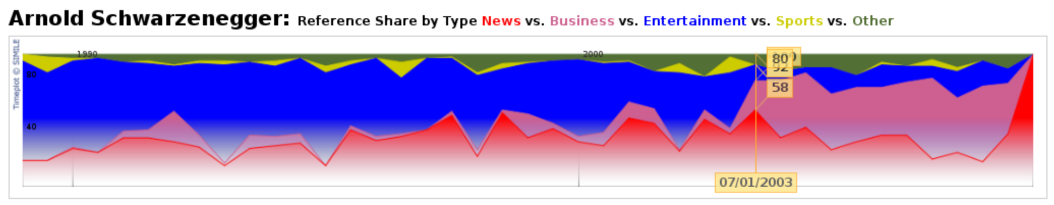


Figure 2.4: The distribution of occurrences of Arnold Schwarzenegger between article categories over time in an archival corpus of U.S. newspapers. The sharp increase in the fraction of "business" articles and decrease in the fraction of "entertainment" articles happen around the time he is elected the Governor of California.

  - ARTICLE_CATEGORY_REFS (1 GB) is similar to ENTITY_CATEGORY_REFS, but contains vectors of entity frequencies in various article categories (e.g. business, sports) instead of entity categories (e.g. person, place).

In the following subsections we examine how some of these artifacts are computed.

2.6 Entity Statistics Collection

Entity statistics collection happens after the EXACT_DUP_ARTICLES artifact is built. The technical details of our approach to duplicate article removal are described in Section 3.5.1.

2.6.1 Entity Time Series

Once duplicate articles have been removed, various artifacts are constructed containing frequency and sentiment time series objects of various types for each entity:

- Per-source entity time series broken up by entity category. The reason the entity category dimension exists is that our NLP pipeline assigns each entity occurrence a category based on the context, and different occurrences of the same entity can have different categories (e.g. car or animal for "Jaguar"). Dimensions: entity category, source ID, date.

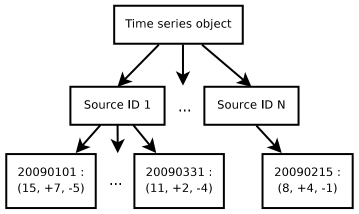


Figure 2.5: A multidimensional map data structure used for time series storage.

- Per-source entity time series broken up by article category (news, sports, business, etc.) in which the entity appeared instead of entity category. Dimensions: article category, source ID, date.
- Per-source entity time series. Dimensions: source ID, date.

All of these objects are represented as sorted multidimensional maps. Their in-memory and serialization format is an array of the form $\{N,\, k_1,\, n_1,\, k_{1,1},\, v_{1,1},\, \ldots,\, k_{1,N},\, v_{1,N},\, k_2,\, \ldots,\, k_3,\, \ldots,\, k_N,\, n_N,\, k_{N,1},\, v_{N,1},\, \ldots k_{N,n_N},\, v_{N,n_N}\}$ (example given for a two-dimensional map). Essentially, it is a pre-ordering depth-first enumeration of a tree such as that shown in Figure 2.5 corresponding to the multidimensional map. Each value $v_{i_1,\ldots,i_m}$ stored in a leaf of such an m-dimensional map is a fixed-length vector containing this entity's frequency and positive/negative sentiment counts.

Using these data structures, entity reference statistics collection is implemented straightforwardly as a map-reduce job:

1. The map phase takes a marked-up article as input. For each entity in the article, it outputs the entity name as a key and a trivial time series object reflecting the references to the entity with the given category, in the given source, and on the given set of dates as a value.
2. The reduce phase of this map-reduce job is `AdditiveReducer`: it uses the addition operation defined on time series objects through the `WritableAdditive` interface to aggregate all individual article-derived time series objects for the same entity into one time series object. The same reducer is used in many map-reduce jobs in our system.

The "additive reduction" used in the entity reference statistic collection job is a common practice in our system. Most time series and other value classes stored

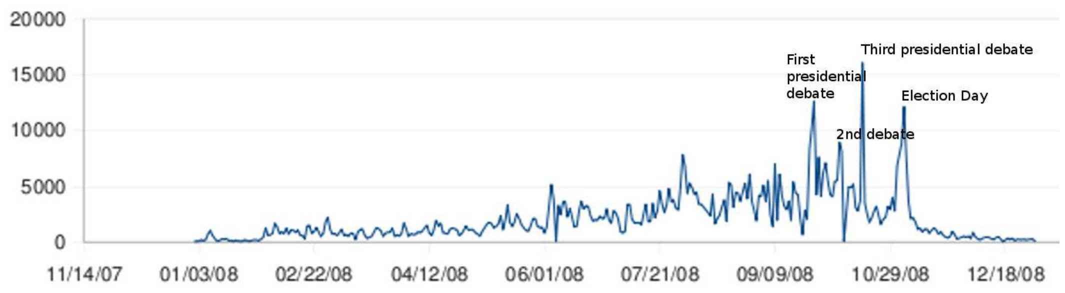


Figure 2.6: Juxtaposition time series for Barack Obama and John McCain

in the Lydia depository implement the `WritableAdditive` interface that allows objects to be added together. Apart from the additive reducer, the additivity property is used in the retrieval part of our system, where data can be aggregated on the fly from multiple map-reduce outputs corresponding to different time periods.

2.6.2 Juxtaposition Time Series

Just as we collect entity frequency and sentiment time series for every entity, we collect co-occurrence (juxtaposition) time series for every pair of entities. Measuring the strength of association between two entities in the news as a function of time can provide useful insights about their interaction. For example, Figure 2.6 shows juxtaposition frequency time series for Barack Obama and John McCain during the 2008 Presidential campaign. By entity or juxtaposition frequency in this section we mean the number of sentences the entity or pair of entities appears in. It is clear that the spikes in the entity co-occurrence frequency are explained by real underlying events involving the two entities.

Juxtaposition time series for each pair of entities are stored in the multidimensional map data structure described in the previous section. The map-reduce job that generates the juxtaposition time series dataset is as follows:

- The map phase takes an article as input, cycles through its list of sentences, and generates a trivial time series object analogous to that described in the previous section for each pair of entities that appear in the sentence. Only the set of entity names appearing in a sentence is taken into account, not the number of times an entity name is repeated in the sentence.

- The reduce phase is an additive reducer that adds up all time series objects originating in all articles for a given pair of entities.

The juxtaposition dataset is one of the most space-consuming datasets in a Lydia depository. To reduce its size and still retain the most significant juxtapositions for every entity, we provide an option to examine only a random sample of input sentences in the mapper of the above map-reduce job. The sampling ratio n is a configurable parameter: every sentence is examined with probability $1/n$ and adds a maximum of n to the juxtaposition frequencies of all pairs of entities found in it.

The number of sentences in the corpus in which two entities co-occur is the most significant characteristic of the strength of relationship between two entities that we derive from text. However, other meaningful measures of entity relationship strength may be based on individual frequencies of entities as well, so as to compensate for the fact that a very frequent entity would co-occur a lot with other entities purely by chance. In particular, the expression (2.1) introduced in [62] is an upper bound on the probability that two entities occur together in at least F out of the total N sentences under the assumption of their independence, while these entities occur in n_a and n_b sentences respectively:

$$P_{bound}(n_a, n_b, F, N) = \left(\frac{e^{\frac{FN}{n_a n_b} - 1}}{\left(\frac{FN}{n_a n_b}\right)^{\frac{FN}{n_a n_b}}} \right)^{\frac{n_a n_b}{N}} \tag{2.1}$$

We call $-\log P_{bound}$ the *juxtaposition score* of this pair of entities. To be able to calculate juxtaposition score time series for every pair of co-occurring entities, we must combine the entity frequency time series dataset and the entity pair co-occurrence time series dataset to generate a per-source time series object of (n_a, n_b, n_{ab}) tuples for every entity pair (a, b). n_a and n_b are frequencies of entities a and b respectively, and n_{ab} is the pair frequency in a particular source and on a particular date.

It is nontrivial to combine individual entity and entity pair frequencies using map-reduce, because high-throughput random access to results of previous map-reduce jobs is not available in the Hadoop framework. For example, we could not write a map-reduce job over a dataset keyed on entity pairs (a, b) and fetch individual time series for entities a and b when processing a tuple. To work around this limitation we use two map-reduce jobs.

1. The first map-reduce job is structured as follows:
   - The map phase takes both single entity frequency and entity pair frequency time series datasets and generates entity pairs as intermediate keys and frequency time series as intermediate values. For a single entity E's frequency time series a pair (E, $null$) is generated as a key.
   - Intermediate keys are grouped by the first entity in the input to the reducer. The `setOutputValueGroupingComparator` Hadoop job configuration function is used to achieve this. Each invocation of the reduce function receives all values generated by the mapper for all keys of the form (E, $\dots$) for some entity E which is fixed within a reducer invocation. The single-entity time series object for E corresponding to the key (E, $null$) appears first in this order. The reducer stores this single-entity frequency time series object TS_E and loops through time series objects $TS_{E,E'}$ corresponding to various pairs (E, E') using the value iterator provided to it by the map-reduce framework. For each pair it emits two types of tuples:
     - Key: (E, E'), value: a combination of TS_E and $TS_{E,E'}$ that stores a vector of $(n_E, 0, n_{E,E'})$ at every leaf node of a multidimensional map such as that shown in Figure 2.5.
     - Key: (E', E), value: a multidimensional map that has $(0, n_E, 0)$ at every leaf node.
2. The second map-reduce job is just an identity mapper and an additive reducer. Its output keys are entity pairs (E, E'), and its output values are multidimensional maps from source IDs and dates to tuples of the form $(n_E, n_{E'}, n_{E,E'})$, where n_E, $n_{E'}$, and $n_{E,E'}$ are frequencies of E, E', and the pair (E, E') in a given source and on a given date.

This allows us to calculate the time series of any juxtaposition strength statistic depending both on pair frequency of two entities and their individual frequencies, such as that given by (2.1). The number N in (2.1) could be obtained from a relatively small dataset of numbers of sentences in every source and every day, which is also part of the Lydia depository.

2.7 Full-text Article Search

In addition to entity occurrence and co-occurrence statistics, it is useful to have a full-text search feature. We provide full-text search by using the Lucene [8] open-source search engine and the Katta [3] library that helps to do Lucene search in a distributed way. Through careful use of multiple fields in our article index and by converting the user's search form input into a query utilizing these fields, we are able to handle the following restrictions added to the query:

- *Entity or article sentiment polarity.* We can query articles by their overall sentiment orientation and by the their sentiment towards a given entity. For example, we can request all articles that speak "very positively" about George Bush. One of the results returned for this query in May 2009 was an article titled "Bush jovial and friendly at Calgary eatery prior to today's speech", which has a distinctively positive sentiment.

  This feature is implemented by maintaining a special-purpose `entity_-category` field in the index. For every entity appearing in the article this field contains the entity name followed by its sentiment polarity value in this article broadly quantized as "very positive", "positive", "neutral", "negative", or "very negative". Also, the sum of all entity sentiment polarities in the article quantized in the same way is stored in the `article_sentiment` field that permits to search for articles having the given sentiment orientation as a whole, irrespective of any particular entity.

- *Time period of article appearance.* We also provide a way to search for articles that appear in a particular time period. This is done by adding a `date` field to every article and using Lucene's `RangeQuery` to search for articles in a given date range.

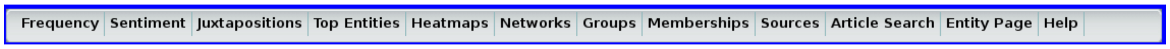


Figure 2.7: The navigation bar of the Lydia web user interface.

- *Entity categories appearing in the article.* The Lydia NLP pipeline marks up entity categories such as person, place, city, and over 100 other categories, assigning a category to each entity occurrence. In our full-text article index we have an `entity_category` field that contains all entity categories that appear in the given article. By searching this field, we are able to handle article set restrictions such as "mentions a religion" placed by the user.

2.8 Web Frontend

The web frontend of the Lydia system provides a way for the user to access and visualize the statistical data stored in Lydia depositories. The frontend web application connects to a number of depository servers that are specified in its configuration, converts user input into depository server API requests, and displays the results of those requests to the user as graphs and tables. This provides the user with time series data of frequency, sentiment score, and relation strength to other entities for any entity or set of entities. The user can also select the source or set of sources he or she is interested in. The yielded data is available for download as a spreadsheet (.csv) as well as in graphical renderings of time series and spatial analysis ("heatmaps") of entity popularity and sentiment for any given period of time. This user interface is available at `http://www.textmap.com/access`.

2.8.1 A Use Case: Political Scientist View

The typical use case for the web frontend of our system, arrived at in collaboration with political scientists from Stony Brook University and the University of Pennsylvania, is as follows. After logging into the system the user can choose

- *The data type to access.* The major data types available through the navigation bar shown in Figure 2.7 are entity frequency time series, entity sentiment

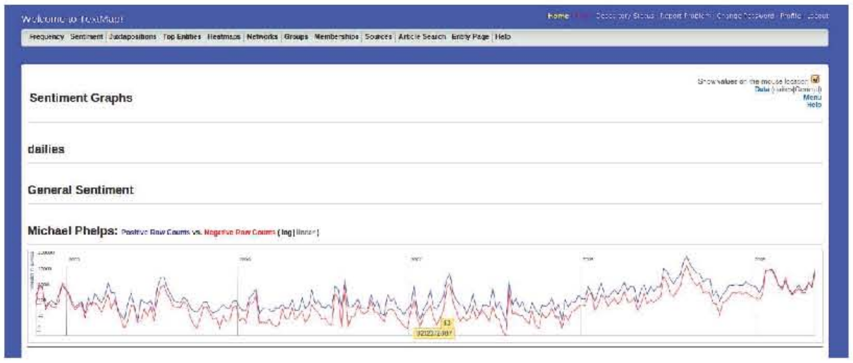


Figure 2.8: Lydia Web frontend: browsing sentiment time series for Michael Phelps.

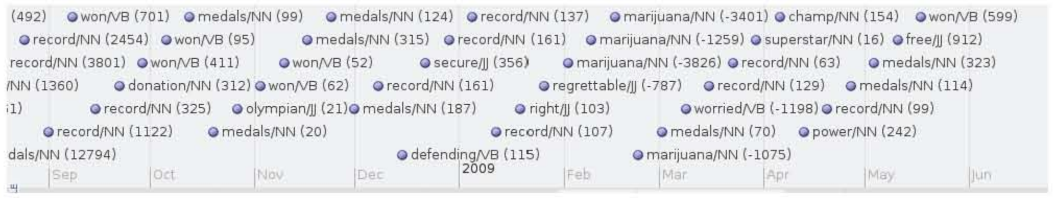


Figure 2.9: A sentiment word timeline for Michael Phelps.

time series, top juxtaposition lists, most popular entities, heatmaps, and entity relation networks.

- *An entity.* Even if the precise entity name entered is not present in the depository, a suggestion list is provided using the depository server's capability to search entity names.
- *A corpus or a source set within a corpus to use.* The user can define his or her own source sets on country, state, and individual source granularity, and save them for later use.
- *A time period* (start and end date).
- *Data aggregation time scale* (daily, weekly, monthly) and the aggregation period length (e.g. 7 to get a weekly time scale instead of daily).

Once these parameters have been identified, the user can do the following, depending on the data access tab chosen:

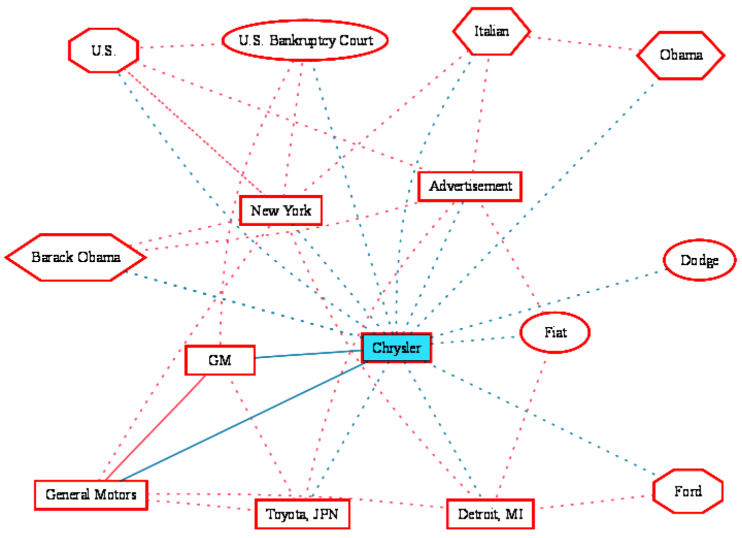


Figure 2.10: An entity relation network for Chrysler in June 2009.

- Browse time series graphs and download these time series as .csv files:
  - Entity reference frequency;
  - Number of articles referencing the entity;
  - Number of sentences referencing the entity;
  - Sentiment counts (positive and negative), scores, and ranks. For example, Figure 2.8 shows a positive and negative sentiment raw count graph for Michael Phelps.
  - A timeline of sentiment words contributing the most to an entity's sentiment score (Figure 2.9). This is useful for understanding the reasons of significant entity sentiment score changes. This feature is not yet available in the production version of the user interface.
- View the top juxtapositions for an entity, and juxtaposition time series for a given pair of entities.
- Browse the entity relation network based on entity juxtapositions, such as that shown in Figure 2.10.

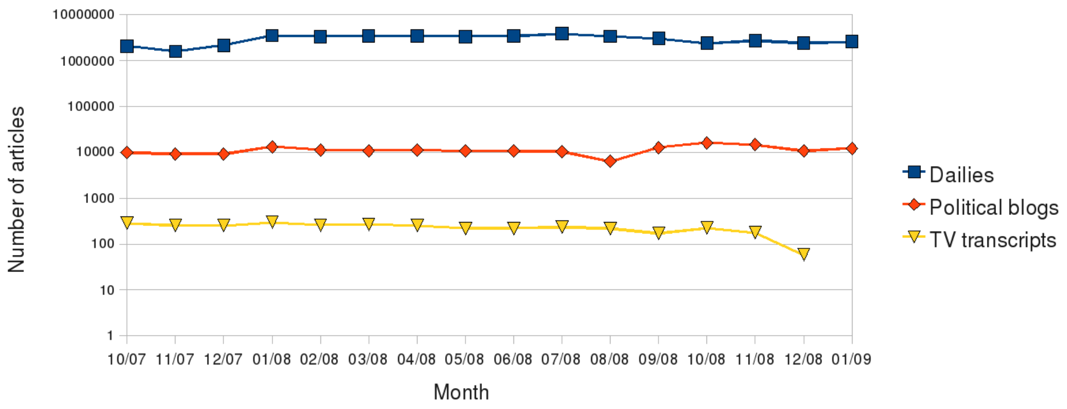


Figure 2.11: Number of articles as a function of time for different types of content in our NAES 2008 text corpus.

2.9 The National Annenberg Election Survey Dataset

We have performed analysis of 16 months of news, political blog, and TV show transcript sources for the National Annenberg Election Survey (NAES), spanning the period from October 2007 to January 2009. This depository was constructed in a somewhat different manner than all our other Lydia system depositories, reflecting the specific needs of the NAES. We used a custom entity list that the NAES team provided us with, because certain words and expressions they cared about (e.g. “inexperienced” or “maverick”) were not considered entities by the legacy Lydia Perl NLP pipeline. Our custom phrase annotation phase is described in Section 2.9.2.

2.9.1 Data Sources

We used documents from three classes of text: the 1000-2000 daily U.S. online newspapers we crawl on a daily basis, 45 political blogs, and transcripts of 13 political TV shows. Figure 2.11 shows monthly numbers of articles from each of these sources on a log scale. Each transcript of a single TV show is represented as one “article” in the system. Table 2.2 shows how many distinct sources those articles came from each month.

Month	Dailies	Political blogs	TV transcripts
10/07	1601	33	12
11/07	1702	32	12
12/07	1849	32	12
01/08	1778	34	12
02/08	1759	33	12
03/08	1764	33	12
04/08	1764	34	11
05/08	1712	33	10
06/08	1719	32	10
07/08	1713	33	10
08/08	1779	33	10
09/08	1683	32	8
10/08	1746	32	9
11/08	922	33	9
12/08	910	33	3
01/09	907	33	0

Table 2.2: Numbers of sources represented in each month of the NAES dataset

2.9.2 Custom Entity Markup

Because the National Annenberg Election Survey team requested statistics on certain phrases that would not be normally considered entities by our NLP pipeline, we had to implement a custom entity markup phase that would recognize entities from a pre-defined list. We were given a list of 626 phrases, which we manually grouped into synonymous phrase groups, one per line, and assigned categories to where appropriate (Figure 2.12).

To mark up these phrases as entities in the text, we replaced the entity markup added by our NLP pipeline with a simple dynamic programming algorithm to select a set of non-overlapping phrases in the text matching the abovementioned predefined phrase dictionary. As the objective function in this dynamic programming algorithm we used the sum of squares of marked-up phrase lengths. This is preferable to using the sum of lengths, because when a phrase and the separate words it comprises are all included in the the custom entity dictionary, preference will be given to marking up the full phrase, as $(a+b)^2 > a^2 + b^2$. An example of a news

```
9-11|September 11
PERSON:50 Cent
ORGANIZATION:AARP
abortion
absentee ballot|absentee ballots
absentee vote|absentee votes|absentee voting
accountability
PERSON:Al Sharpton|Reverend Sharpton
ORGANIZATION:ACORN
```

Figure 2.12: A fragment of the custom entity list.

```
<p> CHARLESTON,/NT WV/NT -/: Polls/NNS are/VBP open/JJ in/IN West/NNP Virginia/NNP ,/, and/CC <pn category="UNKNOWN">election/NN</pn> officials/NNS said/VBD things/NNS are/VBP off/IN to/TO a/DT smooth/JJ start/NN ./.</p> <p> <pn category="PERSON">Hillary/NNP Rodham/NNP Clinton/NNP</pn> is/VBZ expected/VBN to/TO beat/VB rival/JJ <pn category="PERSON">Barack/NNP Obama/NNP</pn> by/IN double/JJ digits/NNS ,/, and/CC score/VB another/DT victory/NN in/IN next/JJ week/NN 's/POS Kentucky/NNP <pn category="UNKNOWN">primary/JJ</pn> ./.</p>
```

Figure 2.13: Custom entity markup example.

text paragraph with these custom entities marked up is provided in Figure 2.13.

2.9.3 Analysis Examples

One example of analyzing the NAES dataset was already given in Table 2.1, showing the top juxtapositions for Barack Obama in four two-month time periods. Table 2.3 offers another view of the same experiment, this time subdivided by three types of sources in our NAES dataset. Unlike in Table 2.1, we did not use co-reference resolution here, which allows us to catch the difference in usage of name variants such as "Hillary Clinton" vs. "Hillary Rodham Clinton". Some differences between topics of political TV shows and other parts of the NAES corpus can be seen from Table 2.3:

- The word "voters" is by far the highest on the TV shows list, indicating that there is more discussion of various groups of voters, including those participating in primaries and caucuses, happening in TV shows than in other types

	Daily News		Political Blogs		TV Transcripts	
1	John McCain	939482	John McCain	9792	Senator	5708
2	Democratic	795907	McCain	6411	John McCain	3961
3	presidential	643408	Hillary Clinton	5753	Hillary Clinton	3842
4	Hillary Rodham Clinton	641815	Democratic	5600	voters	1392
5	Democrat	492442	Hillary	4895	Democratic	1354
6	Hillary Clinton	407742	candidate	3874	Hillary	1237
7	Republican	382197	election	3833	race	1002
8	candidate	333097	voters	3539	McCain	898
9	voters	305741	presidential	3191	Democrats	851
10	Democrats	237960	Democrat	3095	candidate	837
11	race	219677	Democrats	2779	Iraq	761
12	primary	213984	race	2292	presidential	712
13	election	203098	black	2242	debate	636
14	Senator	164169	change	2077	election	630
15	black	155659	Iraq	2063	Iowa	597
16	senator	142834	Senate	1862	change	581
17	White House	141805	Joe Biden	1623	John Edwards	576
18	McCain	139309	Senator	1615	Democrat	555
19	debate	123747	white	1507	Jeremiah Wright	543
20	John Edwards	103155	John Edwards	1158	Pennsylvania	491

Table 2.3: Barack Obama's top juxtapositions in the three sub-corpora of the NAES corpus.

of media.

- TV shows use the word "race" more frequently, and as it turns out looking at the TV transcripts, mostly in the "presidential race" meaning. This is an example of different language usage in different types of media.
- TV shows talk more about controversial issues, as the presence of Jeremiah Wright on the top juxtaposition list indicates.

As another example of analyzing this corpus with the Lydia system, Figure 2.14 shows sentiment polarity time series for Barack Obama taken from the three

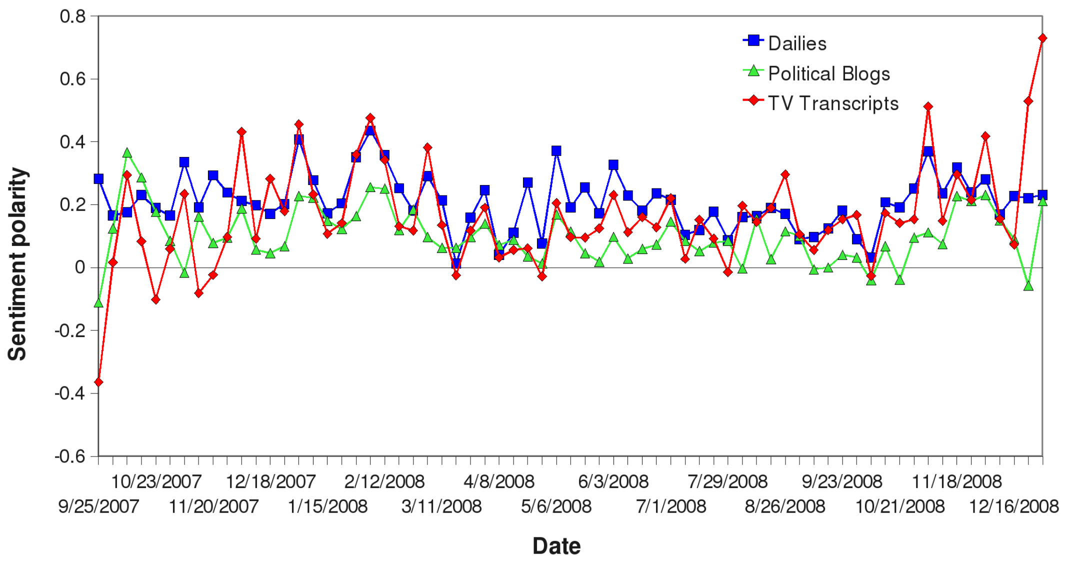


Figure 2.14: Weekly sentiment polarity time series for Barack Obama in the three sub-corpora of the NAES corpus.

different parts of the corpus. The pairwise correlations between these time series are 0.42, 0.49, and 0.37, suggesting that while they are clearly driven by a common trend, there is a difference in how these three different text sources cover events.

2.10 Conclusion

We have described a new version of the Lydia text analysis system that was designed to facilitate efficient data extraction from unstructured text, to satisfy the needs of social scientists. Our system allows a social scientist to obtain statistics about media coverage of a named entity, sentiment of this coverage, entity juxtapositions, and their changes over time. It allows to slice the news statistics by source or group of sources, time period, and time scale. These capabilities are not readily available in previous text analysis systems, which makes our system a valuable addition to a social scientist's toolbox. The new Lydia system was used to provide media coverage data of the 2008 Presidential Election to the National Annenberg Election Survey. As of this writing, the NAES publications on the 2008 U.S. presidential election that use data provided by the Lydia system are still in preparation.

Chapter 3

A Scalable Architecture for Text Stream Analysis

3.1 Introduction

An increasing number of commercial and scientific applications are being developed that must deal with large volumes of text. This includes web-scale and specialized search engines, news and blog aggregation and summarization systems, sentiment analysis systems, question answering systems, and Semantic Web [21,87] entity relation modeling applications. Most of these systems are required to maintain up-to-date versions of their indexes and other pre-computed datasets while the set of documents they have to track is constantly expanding. For certain applications, rebuilding their backend datasets from scratch once every week or month is acceptable, but many applications have to bring their datasets up-to-date on a daily or even hourly basis. For these applications it is necessary to have a scalable and low-overhead method to incrementally update their persistent data structures on arrival of additional documents without disruption of service.

At the same time, the technology for parallel processing of large datasets is rapidly evolving, which changes the way text-processing applications are built. Google's Map-Reduce [39] simplified computation model for large clusters has been implemented in the Hadoop [11] open-source project, and higher-level analysis frameworks based on it such as Pig [72], Hive [43], and Cascading [36] have emerged. A more detailed review of some of these related systems is available in Section 3.2. Open source search libraries such as Lucene [8] and higher-level distributed search servers and frameworks such as Katta [3], Nutch [9], and Solr [10] have appeared. However, standardized and unified approaches to map-reduce workflow management and handling incremental updates to large datasets derived from natural language text are yet to be developed.

In this chapter, we describe our experience of building a workflow management and text stream processing framework for a new scalable version of the Lydia news and blog analysis system [17, 19, 47, 56, 61–63, 66]. The Lydia system performs named entity recognition and pre-computes the statistics necessary to see a picture of news coverage centered around any given entity, including popularity and sentiment time series, juxtapositions between entities, and spatial distributions of news coverage.

Built on top of the Hadoop [11] map-reduce framework, the Lydia system maintains an archive of entity statistics which we call a *depository* containing datasets we call *artifacts* produced by one or more map-reduce jobs. This depository is read by a *depository server* that exposes a named entity-oriented random access API to its clients, which include our web frontend, specialized tools, and ad-hoc data analysis scripts. The Lydia system is capable of building a depository from scratch as well as updating it incrementally with new data arriving on a daily basis without disruption of service. This software architecture, described in Section 3.3, allowed us to successfully overcome the scalability bottlenecks of the old version of Lydia based on a single MySQL database, and to provide access to a multi-terabyte entity statistics dataset.

On top of our workflow framework we have two particular technical contributions:

- *A large-scale cross-document co-referential entity resolution algorithm* – Described in Section 3.7, this algorithm replaces the centralized co-reference resolution algorithm of the old Lydia system described in [63]. To identify co-referential names among entity names extracted by the legacy natural language processing pipeline described in [62] from a text corpus, we apply several hashing methods. Then, entity names collected into the same groups according to some of these hashes are considered connected with an edge.

In this graph, we identify connected components using the union-find algorithm [37], and then refine these sets using our entity juxtaposition information.

- *A method for aggregating entity statistics across arbitrarily defined entity groups* – Described in 3.8, this allows us to pre-compute and incrementally maintain all our entity statistics for entity co-reference sets. Another type of entity group is entity name classification by ethnicity presented in [5], which allows us to examine differences between entity statistics by ethnicity, as done in Chapter 4. Our entity statistics aggregation method works equally well for small groups such as alternative names for the same person or large groups such as all people of a certain ethnicity.

In Section 3.9 we evaluate the performance and scalability of our system as a function of the amount of data to be processed and the number of nodes in the Hadoop cluster. We also test the system by marking up all WordNet nouns instead of named entities. Section 3.10 presents future directions for Lydia infrastructure development.

3.2 Related Work

3.2.1 Hadoop

Apache Hadoop [11,94] is an open-source implementation of Google's map-reduce [39] computation model and a distributed file system modeled after the Google File System [46]. It is implemented in Java and runs on Linux and Windows clusters. The Hadoop framework manages many traditionally time-consuming and error-prone components of distributed computation systems, such as inter-process communication and failure handling, at the expense of a restricted computation model. Our text analysis architecture uses Hadoop both as a distributed storage system and a distributed computing platform.

3.2.2 Pig

Yahoo's Pig [72] is a data processing environment that implements a query language called Pig Latin combining declarative constructs inspired by SQL and procedural statements. Some Pig Latin primitives include `FOREACH`, `FILTER`, `GROUP`, `COGROUP`, and `JOIN`. Pig is independent of the underlying execution platform, but currently only runs on top of Hadoop. Pig compiles the user's program into a sequence of map-reduce jobs that are run on a Hadoop cluster. It simplifies data analysis tasks compared to programming directly to the map-reduce framework. An innovative feature of Pig is its debugging environment, capable of producing a sandbox dataset for verifying and refining user's programs without incurring the overhead of time-consuming large-scale processing.

Both Pig and our new Lydia architecture are implemented on top of the Hadoop map-reduce platform, but their purposes are orthogonal. Pig is a general-purpose data exploration system designed primarily for batch querying. The Lydia system, on the other hand, builds a specialized statistical dataset from a text corpus, updates it with new data, and provides fast on-demand retrieval of these statistics. We can use Pig to compute certain elements of our statistical datasets in the future.

3.2.3 Hive

Facebook's Hive [43] is a data warehousing infrastructure built on top of Hadoop for analysis of large datasets stored in Hadoop files. It has a query language based on SQL which translates into sequences of map-reduce jobs. Therefore, the minimum possible query response time in Hive is on the order of minutes. Hive provides a way to define a data schema that can be used to organize data stored in a variety of formats (Thrift [88], Hadoop's RecordIO, tab-separated, or user data format). The schema metadata and the way data is partitioned between Hadoop files is stored in a specialized *metastore*.

The difference between Hive and our scalable Lydia architecture is that Hive is a general-purpose data extraction system with its own SQL-derived language. Lydia, on the other hand, is focused on text analysis, and provides some features Hive lacks, such as a flexible map-reduce dataset update logic and on-demand retrieval of data from a historical statistics archive that allows for interactive data exploration.

3.2.4 Sawzall

Google's Sawzall [79] is a specialized parallel data analysis language implemented on top of Google's map-reduce framework. The best application of Sawzall is analyzing large but homogeneous datasets in such a way that the volume of extracted statistics is orders of magnitude smaller than the input data. A Sawzall program is written to handle only one record, and emits values to a number of aggregators (tables) that it defines. Aggregators can be sums, lists of all values, or more complicated structures allowing to compute top values by some weight function, quantiles, most popular values, or an unbiased random sample of emitted values. In addition, aggregators can be indexed, allowing the creation of multidimensional aggregators. Although Sawzall is a batch processing system, the original thinking behind it was a streaming processing approach where a Sawzall program would keep up with the data flow and aggregators would be maintained in an on-line server. That concept was not implemented in Sawzall [79] because the availability of map-reduce made it unnecessary.

Although Sawzall is a much more powerful and general-purpose statistics collection framework than our new Lydia architecture, it lacks some features the Lydia architecture provides, such as a higher-order chaining and dependency management framework, or an on-line server for maintaining and presenting aggregate statistics with updates happening in the background.

3.2.5 BigTable

BigTable [32] is Google's column-oriented distributed database with a simplified non-relational data model and row-only locking. For every row it allows one to store a number of *column families* pre-defined by the schema, each of which can contain any number of columns. Internally, an ordered immutable sorted map file format called *SSTable* is used to store the data. Each table at any moment is split between a number of *tablets* corresponding to the table's row ranges, which are served by a number of tablet servers. Writes to a table initially get stored in a commit log and an in-memory sorted table called *memtable*, but eventually a variety of *compactions* are performed to move data from commit log to SSTables and to consolidate multiple small SSTables into bigger ones.

BigTable powers the backend of a variety of Google's applications, and has inspired multiple open-source column-oriented database implementations including HBase [7], Hypertable [102], and Cassandra [42]. We considered using HBase to store our per-entity statistics but at the time of making this architectural decision the performance overhead introduced by the necessity of insertion of map-reduce job results into HBase was unacceptable, so we switched to using plain Hadoop *MapFiles* (analogous to SSTable) as our storage primitives. As of this writing, many performance issues in HBase have been addressed and we may consider using it as a backend storage engine for the Lydia system in the future.

3.3 Architecture

The previous version of the Lydia system described in [62] was capable of processing roughly 250 articles per hour on a single computer [60]. Around 2.5% of that time was spent during the NLP (natural language processing) phase and 97.5% was spent on post-processing and database-intensive operations. Thus in redesigning the Lydia system we focused on removing the bottleneck of a single relational database and implementing the post-processing and statistics collection functionality on top of the Hadoop [11] map-reduce [39] framework.

3.4 Processing Flow

The processing flow in our new Lydia system contains the following steps:

- *Input document collection* – Source documents come from Lydia news and blog spiders, RSS feeds, or as custom corpora provided by users.

- *NLP pipeline* – Documents are passed through a specialized version of our legacy Lydia NLP pipeline [60, 62], which runs as a Hadoop Streaming job. It marks up and classifies named entities, resolves intra-document coreferences, normalizes geographical location names, and assigns sentiment scores to entity references. The resulting text corpus with marked-up named entities is the input for our scalable text analysis system.

- *Depository construction or update* – Documents are processed through a series of map-reduce jobs and the results are stored in a directory structure in HDFS (Hadoop Distributed File System) that we call a *depository*, which includes:
  - All input documents with exact duplicates removed. Our duplicate removal methods are described in Section 3.5.1.
  - Entity reference statistics: per-source frequency and sentiment time series by entity and article category.
  - Entity juxtaposition statistics: time series of co-occurrence frequency for every pair of entities.
  - A full-text index of articles. We use Katta [3] and Lucene [8] to do distributed full-text search.
  - An index of entity names. We use Lucene [8] to search for entity names.
  - Groups of co-referential entity names. See Section 3.7 for a discussion of our cross-document co-reference resolution algorithm.
  - Entity classification by ethnicity and country association.
  - Entity and juxtaposition statistics aggregated over co-reference sets and ethnic groups. See Section 3.8 for a description of our group statistics aggregation method.
- *Database export* – We still utilize a relational database to provide more flexibility of data exploration. However, we only export a subset of the Lydia depository into the database, which includes statistics for a number (currently around a million) of the most frequent entities and co-reference sets.

3.5 Data Organization

Our initial experience with time series and statistics extraction from news text using Hadoop made clear that a framework was necessary to build, store, and retrieve these statistics and other pieces of data derived from text, and manage dependencies. We call the logical units of data containing a specific type of statistic derived from text *artifacts* and the directory hierarchy in the distributed file system comprising multiple such artifacts for a specific corpus a *depository*.

More precisely, an artifact in our framework is a homogeneous dataset of a well-defined key-value structure that is stored in a specific directory hierarchy in HDFS. There are the following types of artifacts:

- **Map-reduce artifacts.** These artifacts are produced and updated with Hadoop map-reduce jobs. We have three distinct types of map-reduce artifacts:
  - *Readable key-value artifacts.* Examples include frequency and sentiment time series for each entity, co-occurrence time series for each entity pair, or a list of top-frequency entities for each day. These artifacts are stored in HDFS as groups of Hadoop's `MapFiles`. A `MapFile` is a combination of a data part (a sorted key-value file) and an index part (a subset of keys providing an index into the data part), similar to Google's SSTable [32].
  - *Lucene [8] indexes.* Examples include full-text article index and entity name index. These artifacts are used for storing entity name and full-text article indexes. They are still produced using map-reduce jobs, but each output part is a Lucene index instead of a key-value file. These artifacts are used for article search which is discussed in Section 2.7.
  - *Intermediate artifacts.* These artifacts, also having a key-value structure, are used as intermediate results in some computations that require more than one map-reduce job. Therefore, they are not always stored as random-access `MapFiles`, but rather as plain sorted key-value files.
- **Folders.** A folder is a special type of artifact that can contain other artifacts. This gives the Lydia depository a hierarchical structure. Every artifact has a corresponding fully-qualified name of the form "folder_name1.folder_-name2.$\cdots$.artifact_name". Examples of Lydia depository folders are:
  - `exact_dup_stats` – statistics for entity names obtained from a set of articles with exact duplicates removed;

    - `synset_stats` – statistics for co-reference sets;
    - `group_stats` – statistics for ethnic and national groups.

- **Bloom filter artifacts.** These artifacts store bit arrays corresponding to Bloom filters [22] for various entity name sets. An example is the Bloom filter accepting the most frequent 1,000,000 entity names. We use Bloom filters to restrict the number of entities handled by various parts of our system, such as the entities and co-reference sets exported to a MySQL database for convenient data exploration. A great convenience of using Bloom filters with Map-Reduce is that they are small enough so they can be loaded into memory by each mapper, making complicated multi-input joins unnecessary.

Since we need to process large historical text corpora quickly, as well as to keep up-to-date with new documents arriving every day for some corpora such as U.S. dailies, our framework contains means to accommodate these two different modes of processing. All Lydia artifacts, except folders, are time-based. The artifact's directory contains subdirectories corresponding to one or more date ranges for which the data has been generated.

3.5.1 Duplicate Removal

News streams contain duplicate and near-duplicate news articles. This is partially explained by our news spiders erroneously downloading the same article more than once on different days, or by the same article existing under different URLs within a news website. Even if the spiders can be made perfect, we still must deal with issues of news syndication (Reuters, Associated Press) and our historical news archives, so duplicate removal is an essential step we must take prior to any analytical processing.

3.5.1.1 Exact duplicate removal

In Figure 2.3 we can see that the very first job that runs on the text with named entity markup removes exact duplicate articles and produces the EXACT_DUP_ARTICLES artifact. This map-reduce job works as follows:

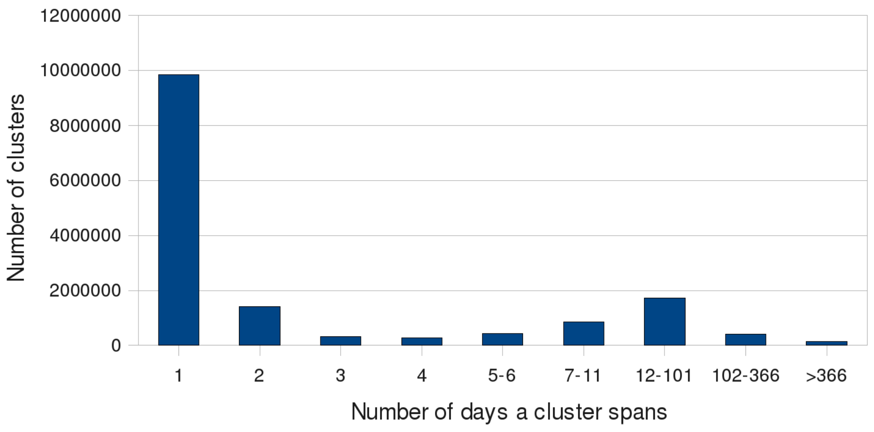


Figure 3.1: Statistics of exact duplicate removal on a four-year U.S. daily news dataset.

- The mapper takes an article and produces the MD5 hash of the article's normalized text (lowercased, punctuation removed, whitespace reduced to single spaces) as a key and the full text of the article as a value.
- The reducer takes articles grouped by the MD5 hash of their normalized contents. For every group of articles with the same hash it produces an article that contains the text of the first article in the group and aggregated source, date, and URL metadata for all occurrences of this duplicated article.

During later phases, when time series are constructed, the occurrence metadata included into every article in the output of the exact duplicate removal job is used to count every repeated article towards all sources in which it appeared. This allows us to reconstruct per-source entity occurrence time series appropriately even in cases the same article happens to occur in multiple sources and/or on multiple dates. This is important for correct visualization of entity frequency and sentiment time series and heatmaps.

To make this exact duplicate removal algorithm function in the scenario of daily updates to our document collection, we add an artifact EXISTING_ARTICLE_HASHES containing MD5 hash codes of all articles to date. This artifact is less than 1% of the size of the article dataset. We use EXISTING_ARTICLE_HASHES as an additional input to the above map-reduce

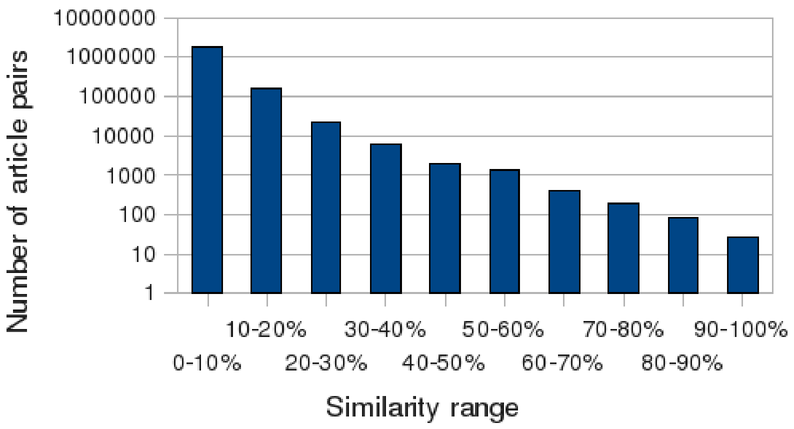


Figure 3.2: A near-duplicate detection experiment using similarity measure (3.1) performed on 78421 U.S. news articles spanning a 10-day interval.

job and thus filter out articles that have appeared in the past.

Figure 3.1 gives some statistics of exact duplicate removal from our four-year U.S. news archive. From the large number of days on which some articles are repeated it is evident that the exact duplicate removal phase managed to compensate for some imperfections of our news spiders. From Figure 3.1 we see that most articles are not repeated, but some articles are repeated across long periods of time. These articles most likely represent stubs or static pages on the news website. The earliest date of appearance is used for these articles in entity time series calculation.

3.5.1.2 Near-duplicate removal

Near-duplicate articles may be present in the news as a result of multiple news outlets using the same syndicated articles but fitting them differently to their publication area. To check for near-duplicate articles in our U.S. daily news corpus, we conjectured that these articles differ in a small fraction of sentences.

We computed a sentence-level similarity between pairs of articles A_1 and A_2 as follows:

$$\frac{|\{s \in A_1 \cap A_2 \mid len(s) \geq l_0\}|}{\max(|\{s \in A_1| \; len(s) \geq l_0\}|, |\{s \in A_2 \mid len(s) \geq l_0\}|)} \tag{3.1}$$

It essentially measures the fraction of sentences of length at least l_0 that are shared between two articles. The sentence length restriction is used to eliminate very frequent short sentences. If one article is significantly longer than the other, the similarity is discounted by the increased denominator.

However, less than 6% of all pairs of articles on our test set of 78421 articles spanning 10 days had a similarity of 0.5 or more according to the measure (3.1) with l_0 equal to 20 characters. Exact duplicates were removed from the test set before conducting the near-duplicate experiment. The distribution of similarities between all article pairs is shown in Figure 3.2. Because there were so few pairs of articles with significant overlaps on the sentence level, we decided not to pursue near-duplicate article removal in the current version of the Lydia system.

3.5.2 Processing Scheduling

The artifacts in a Lydia depository and the dependencies between them form a directed acyclic graph (DAG) that imposes constraints on the order in which the artifacts can be built or updated. Our system has a scheduler that determines the order in which map-reduce jobs used to build or update a depository are run. The scheduler uses the following data to schedule and configure the map-reduce jobs:

- The set of target artifacts to build or update. This can be specified e.g. as an artifact name wildcard on the command line.
- Date ranges existing for each artifact.
- Per-artifact settings specified during depository creation:
  - Input artifacts. These are the artifacts whose data is used as input to build or update the given artifact.
  - "Side input" artifacts. These artifacts are used as additional inputs but do not influence the artifact build order. An example of a side input is the EXISTING_ARTICLE_HASHES artifact used in section 3.5.1.1.
  - Whether the artifact needs "final aggregation", which is typically done via an additional map-reduce job with an identity mapper and an additive reducer.
  - Date range merging strategy.

We use two algorithms for scheduling the map-reduce jobs that update a depository: sequential and parallel. The sequential version is easier to understand, while the parallel version provides improved performance when running on an underutilized Hadoop cluster.

3.5.2.1 Sequential depository build

For a given set of target artifacts to build or update T we identify all of their prerequisites recursively. Let us call this union of T and all their prerequisites D. From this extended set of artifacts of interest, we select the artifacts that are out-of-date. The operation of identifying whether an artifact is out-of-date is polymorphic, i.e. how it is done depends on the artifact type. The default implementation is to check if the date range set of the artifact contains the union of date range sets of its input artifacts. In that case, the input artifacts contain no new data and the artifact in question is up-to-date. Otherwise, the artifact is marked as out-of-date.

Once all out-of-date artifacts are identified, all artifacts in D that have at least one out-of-date prerequisite are marked. These are the artifacts to be built. Finally, these artifacts are arranged in the order of a topological sort of the artifact DAG. This is the order in which artifacts are built or updated.

3.5.2.2 Parallel depository build

A single map-reduce job does not utilize a cluster evenly. There are inevitably some map and reduce tasks that run longer than others, and the job does not complete until all tasks complete. To improve cluster utilization and shorten depository build time we use an algorithm that schedules more than one map-reduce job at a time. There is a configuration parameter specifying the number m of map-reduce jobs that can be run at the same time by a single depository build process.

The first step of a parallel depository build is to determine the set of artifacts to be built. We do this precisely as described in Section 3.5.2.1. The resulting artifact list gives us the initial contents of the list L of artifacts left to be built. Then, the scheduler goes through the following steps in a loop until the list L is empty:

1. All the artifacts in the list L that have not been submitted for building are examined, and as long as there are fewer than m artifacts being built, all

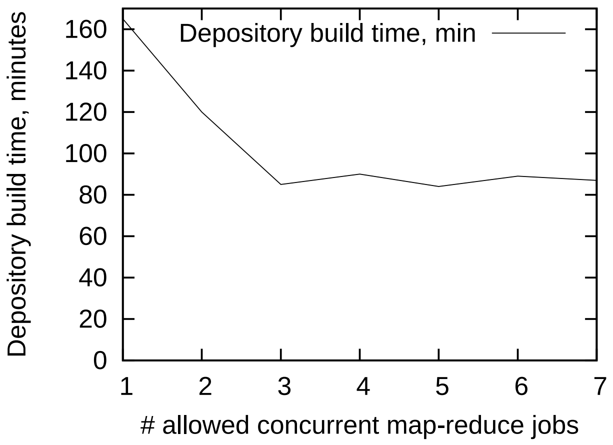


Figure 3.3: Parallel depository build time as a function of the number of allowed concurrent map-reduce jobs.

those that have all of their dependencies successfully built are submitted for building.

2. The next build report is requested from the pool of artifacts currently being built. If no new artifact build operations have completed by this point, the request blocks until the next report is available.

3. The artifact build report is stored in a map under the key of the artifact ID so that the success of the artifact build can be easily checked in step 1. Also, the artifact is removed from the list L.

Once the list L has been cleared, an attempt has been made to build all artifacts whose prerequisites could be successfully built. There can be artifact build failures, but we implement no retry logic beyond that inherently present in the map-reduce framework, because map-reduce job failures usually indicate software or configuration problems that require manual intervention. In more than two months running the Lydia depository build process every day, we have not met with any serious failures during the build process.

We have experimented varying the number m of allowed concurrent map-reduce jobs in this algorithm from one to seven on a one-week subset of 486,249 U.S. newspaper articles. Figure 3.3 shows how the total build time of the entire depository

DAG changes as a function of m. With our depository graph structure the optimal value of m is three, as further increases of m do not improve performance but introduce additional contention overhead. We analyzed our depository build logs with $m = 3$ and found that the number of concurrently built artifacts averaged over time was 2.26, and 59% of the time three artifacts were being built concurrently.

3.5.2.3 Job input selection

The Lydia depository framework automatically assigns input for each map-reduce job it runs. The input of a map-reduce job that builds or updates an artifact is selected from the available date ranges for its input artifacts to satisfy the following conditions:

- The input date ranges taken from a single input artifact should not overlap.
- The input ranges date should not overlap with date ranges already present for the artifact being built.

We satisfy these conditions by selecting a minimum cover of the difference $\cup_i A_i \setminus \cup_i B_i$ with the date range set $\{A_i\}$, where $\{A_i\}$ and $\{B_i\}$ are the date range sets of artifacts A and B respectively, and A is an input artifact for B. Note that in case of valid functioning of daily depository updates this set cover always consists of non-overlapping date ranges and its union is precisely $\cup_i A_i \setminus \cup_i B_i$. Therefore, it represents a *set packing* of the $\cup_i A_i \setminus \cup_i B_i$ with date ranges of $\{A_i\}$. However, we compute these set packings using standard set cover algorithms for intervals. We will therefore use the terms “set cover” and “set packing” interchangeably in the remainder of this section and in the following section.

This rule is applied to compute the date ranges to use from each input artifact A of the artifact B. All of these date range directories combined across all inputs A of B compose the input of the map-reduce job that builds or updates the artifact B. For example, Figure 3.4 shows a selection of two date ranges of artifact A that are chosen by our framework as input to the appropriate map-reduce job to bring artifact B up-to-date. Note that we could also use three non-overlapping date ranges of the same length that give us the same union, but the smallest possible number of input date ranges is chosen.

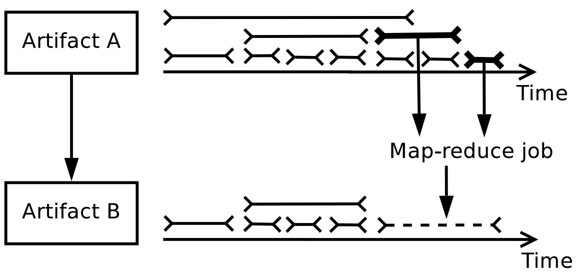


Figure 3.4: Prerequisite date range selection for map-reduce job input used to bring an artifact up-to-date.

We also have a notion of a “side input” of an artifact defined as follows: if A is a side input of B, then the entire set cover of date ranges of the artifact A is included into the input of each map-reduce job that builds B, ignoring the rule above. This feature is used in our incremental exact duplicate removal method described in Section 3.5.1.1.

3.5.3 Date Range Directory Merging

When depository updates are run every day over a period of time, the artifact data becomes scattered across many single-day and multiple-day date ranges. Most updates are single-day, but multiple-day updates happen when processing gets skipped for one or more days, e.g. due to power outage. This slows down data retrieval, because the requested key has to be looked for in multiple date range-named directories. To keep the number of date ranges necessary to read an artifact small, we merge a subset of date ranges for an artifact after it has been updated. The merging operation is usually an `AdditiveReducer` run on a subset of date range directories, producing a directory named as a smallest encompassing range of the merged set. We have multiple strategies for identifying the set of date ranges to merge.

To formalize our date range merging problem, suppose we have a set $A = \{[a_1, b_1], \ldots, [a_n, b_n]\}$ of intervals, such that $a_i < b_i$ and $b_i \leq a_{i+1}$, corresponding to the date ranges of data successively added to an artifact (e.g. if $b_i = a_i + 1$, $[a_i, b_i]$ is a single-day date range). A merging strategy is then a function f mapping a set of date-range intervals $\{[a'_1, b'_1], \ldots, [a'_m, b'_m]\}$ existing in the artifact at a certain point to its subset $\{[a'_{i_1}, b'_{i_1}], \ldots, [a'_{i_l}, b'_{i_l}]\}$ that has to be merged. The merging strategy is successively applied to the sequence of daily updates A, producing intermediate sets of date ranges after each number j of updates: $B_1 = \{[a_1, b_1]\}$ and $B_j = B_{j-1} \cup \{[a_j, b_j]\} \cup \left(\bigcup_{[a',b'] \in f(B_{j-1} \cup \{[a_j,b_j]\})} [a', b']\right) \setminus f(B_{j-1} \cup \{[a_j, b_j]\})$ for $j > 1$. We are interested in the trade-off between two performance measures of a date range merging strategy:

- Merging overhead. For a single update, e.g. the j'th ($j > 1$), this overhead is defined as $\mu_j = \sum_{[a',b'] \in f(B_{j-1} \cup \{[a_j,b_j]\})} (b' - a')$. It represents the time spent to merge date ranges $f(B_{j-1} \cup \{[a_j, b_j]\})$ chosen by the merging strategy under the assumption that the merging operation is linear in the total amount of input data and the same amount of data arrives every day. We also consider the amortized per-day merging overhead calculated as $\sum_{j=1}^{n} \mu_j / \sum_{j=1}^{n} (b_j - a_j)$.

- Random access overhead. This is calculated as the number of date ranges in the minimum set cover of the intermediate set of date ranges B_j after j updates and merging operations. We are interested in how this depends on the total number of days in all updates $[a_1, b_1], \ldots, [a_j, b_j]$.

3.5.3.1 Total merging strategy

All date ranges are merged into one range after each artifact update, i.e. $f(A) = A$. This is suitable for relatively small and frequently accessed artifacts such as those containing total frequencies for all entities and mappings from entities to co-reference sets and vice versa. This strategy has a merging overhead proportional to the total amount of data at every update, which is quite wasteful. However, it has the lowest possible random access overhead, because all data is available through only one date range after every update.

3.5.3.2 Binary merging strategy

This strategy aims to maintain an $O(\log n)$ number of date ranges in the minimum set cover of an artifact's set of date ranges after n days of updates to keep the random access overhead manageable. Suppose an update happens every day, starting from day one, generating another single-day date range. If every day i is represented by

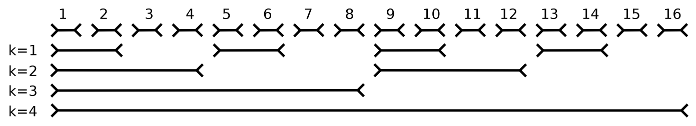


Figure 3.5: Date ranges merged by the binary merging strategy during daily updates over a 16-day period.

a unit-length interval $[i-1,i]$, we can merge date ranges of the form

$$[2^{k+1}j, 2^{k+1}j + 2^k] \tag{3.2}$$

where $k \geq 1$ and $j \geq 0$, as shown in Figure 3.5.

For example, on the day 13 in Figure 3.5 the minimum cover of the date range set for the artifact will contain ranges $[0,8]$, $[8,12]$, and $[12,13]$ as a result of this merging strategy. As another example, suppose daily updates start happening on May 1, 2009. On May 2, we merge data from May 1 and May 2, producing a single-range set cover. On May 3, we don't merge anything, and have a two range set cover (May 1-2 and May 3). On May 4 we merge the ranges for May 1-2, May 3, and May 4, and again obtain a single-range set cover. On May 5 we don't merge anything and have a two-range set cover, May 1-4 and May 5. On May 6, we merge May 5 and May 6 data and obtain a two-range set cover (May 1-4 and May 5-6), and so on.

Suppose that the time complexity of a merging operation for a group of date ranges of total length of n days is $O(n)$. Assuming that an update happens every day, it is clear from Figure 3.5 that the total time complexity of all merge operations happening in $n = 2^K$ days is $O(2^{K-1}(K-1)) + O(2^K) = O(2^{K-1}(K+1)) = O(n \log n)$. This gives an amortized daily merging overhead of $O(\log n)$. This is confirmed by our simulations for numbers of days n different than a power of two.

To prove the $O(\log n)$ random access overhead bound, suppose we want to retrieve data from the artifact on the m'th day of daily updates, $m < n$. For the retrieval result to be complete, we must use set of non-overlapping date ranges that cover the full range of days 1 to m. If the binary representation of m is $m = 2^{i_1} + \cdots + 2^{i_s}$, $0 \leq i_1 < \cdots < i_s < k$, the artifact will have a date range set packing containing ranges of lengths $2^{i_1}, \ldots, 2^{i_s}$ at that time. Therefore, no more than $O(\log m)$ date ranges will have to be opened to perform the retrieval.

```
Input: A set $S = \{[a_1, b_1], \dots, [a_n, b_n]\}$ of date ranges satisfying the
       containment property.
Output: A subset $M \subset S$ of date ranges to merge such that adding the
        encompassing date range of $M$ to $S$ does not violate the
        containment property.
1  $b \leftarrow \max_i b_i$
2  $a \leftarrow$ the minimum $a_i$ such that $b_i = b$
3  $M \leftarrow \{[a, b]\}$
4  while $\exists\, i \in \{1, \dots, n\}$ such that $b_i = a$ do
5      $a' \leftarrow$ the minimum $a_i$ such that $b_i = a$
6      if $a - a' \leq b - a$ then
7          $M \leftarrow M \cup \{[a', a]\}$
8          $a \leftarrow a'$
9      else
10         break
11     end
12 end
13 if M contains only one date range then
14     $M \leftarrow \emptyset$
15 end
16 return $M$
```

Algorithm 1: The binary date range merging algorithm.

This scheme only deals with the case when all added date ranges are one day in length. But our binary merging algorithm must be able to handle multiple-day updates, for example, when the system misses a day of processing due to a power outage and produces a two-day update when it is brought up. The generalized version of the merging algorithm presented below relies on and maintains the property of "containment": every two date ranges of the same artifact either have a zero-length intersection, or one of them is contained in the other. The special-case merging algorithm described by (3.2) and Figure 3.5 obviously maintains this property. This property is important for reading the artifact because it guarantees that in the minimum packing set of date ranges every pair of date ranges will also have a

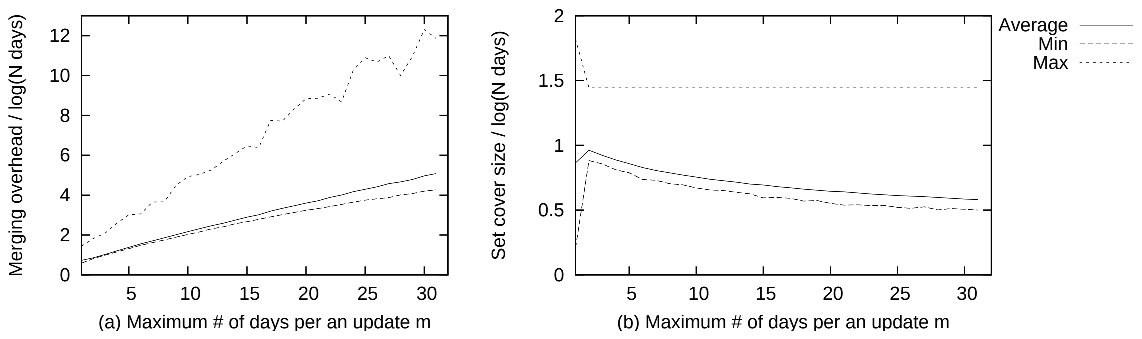


Figure 3.6: The ratio of the merging overhead (a) and of the minimum set packing size (b) of Algorithm 1 to the logarithm of the number of days depending on the maximum update date range length.

zero-length intersection.

The general-case Algorithm 1 takes a set of date ranges and produces its subset that should be merged. It is easy to see that if a single-day date range is generated every day, Algorithm 1 merges exactly the date ranges described by (3.2) and shown in Figure 3.5.

To see why Algorithm 1 does not break the containment property of the artifact's date range set S, suppose it outputs $M = \{[a_{i_1}, b_{i_1}], \ldots, [a_{i_m}, b_{i_m}]\}$, where $b_{i_1} = a_{i_2}, \ldots, b_{i_{m-1}} = a_{i_m}$. The only way the set $S' = S \cup [a_{i_1}, b_{i_m}]$ can violate the containment property is when there is a date range $[a_l, b_l] \notin S'$ such that $a_l < a_{i_1}$ and $a_{i_1} < b_l < b_{i_m}$. If $b_l \notin \{a_{i_2}, \ldots, a_{i_m}\}$, then $b_l \in (a_{i_j}, b_{i_j})$ for some j and S itself violates the containment property. If, on the other hand, $b_l = a_j = b_{j-1}$ for some $j \in \{2, \ldots, m\}$, then a_l should have been selected as a' instead of a_{j-1} at line 1. Therefore, the new set S' of date ranges does not violate the containment property.

We conducted simulations to calculate the per-day merging overhead and the average random access overhead of Algorithm 1 with the number of days per update distributed uniformly from 1 to m, where $m = 1, \ldots, 31$, as a function of the total number of days n in all updates. Figure 3.6 (a) shows that the per-day merging overhead of Algorithm 1 is $O(m \log n)$. However, in our real use cases m does not exceed 4, so the per-day merging overhead of Algorithm 1 for our purposes is effectively $O(\log n)$ even in case of a variable number of days per update. The performance of Algorithm 1 in terms of the random access overhead (minimum set cover size) is even better: Figure 3.6 (b) shows that for any number of days per update the number of date ranges in the minimum set cover stays $O(\log n)$.

3.6 Depository Server

To provide the web frontend (see Section 2.8) and research scripts access to the collected statistics, our system includes a *depository server*. The depository server provides access to one or more Lydia depositories via a Thrift [88] API. Currently the depository server for a given depository or group of depositories runs on a single machine. This is acceptable given our current workload, because we can serve different depositories on different machines, and the data structures necessary to access the largest of our depositories (U.S. daily newspapers) still fit into one machine's memory.

The API exposed by the depository server allows access to various artifacts present in the underlying depository. Here is a representative sample of API functions provided by the depository server. The `IdFilter` type is used to specify source ID sets to return statistics for, and whether the results should be aggregated across source IDs.

- `list<ArtifactDesc> getArtifacts()` – returns a list of structures describing every artifact handled by this depository server, for all available depositories.
- `list<DepositoryDesc> getDepositories()` – returns a list of structures describing every depository handled by this depository server.
- `list<SourceDesc> getSources(string depository_name)` – returns a list of structures describing all document sources in the given depository.
- `list<EntitySearchResult> searchEntities(string full_artifact_name, string query)` – searches the given entity index artifact and returns a list of entity name results along with their frequencies.

- `list<ArticleSearchResult> searchArticles(string full_artifact_name, string query, IdFilter source_filter, i32 start_index, i32 num_results)` – uses Katta [3] to search the given article index artifact and returns the given number of results starting from the given starting position.

- `map<i32, IntTimeSeries> getPerSourceEntityTimeSeries(string full_artifact_name, string entity_name, TimeFilter time_filter, IdFilter source_filter, i32 counter_id_mask)` – retrieves frequency and/or sentiment time series for the given entity from the given time series artifact with the specified filtering by source and time (see Section 3.6.3). The bit mask of types of time series to retrieve (e.g. frequency, positive and negative sentiment counts) is specified by `counter_id_mask`.

- `JuxtaResult getPerSourceEntityPairJuxtaTimeSeries(string full_artifact_name, string entity1_name, string entity2_name, TimeFilter time_filter, IdFilter source_filter)` – retrieves juxtaposition time series for the given pair of entities from the given artifact of the appropriate type with the specified filtering by source and time.

- `list<JuxtaResult> getPerSourceJuxtaTimeSeries(string full_artifact_name, string entity_name, i32 top_count, bool include_score, bool sort_by_score, TimeFilter time_filter, IdFilter source_filter)` – retrieves the top juxtapositions for the given entity, optionally including juxtaposition score (2.1), in the order of either decreasing frequency or decreasing juxtaposition score. Each result entry contains a juxtaposed entity name and a juxtaposition time series filtered by source and time according to `time_filter` and `source_filter`.

The depository server API includes two types of functions: those that return information about the contents of the server as a whole (`getArtifacts()`, `getDepositories()`, `getSources()`) and those that query specific artifacts. The majority of the depository server API functions are of the latter type. They accept a "full artifact name" as their first argument, which is an artifact name with all enclosing folders prepended by the depository name, such as dailies#exact_dup_stats.PER_SRC_ENTITY_TS. Other arguments of artifact-querying API functions usually include an entity name, a pair of entity names, or a search query, a specification of the source set to retrieve the data from (`source_filter`), and a specification of a date range to retrieve time series data for and the granularity of its aggregation (`time_filter`).

3.6.1 Depository Server Implementation Overview

The primary components of the depository server are shown in Figure 3.7. The multi-depository handler dispatches API requests to the correct single-depository handler based on the depository name included in the full artifact name provided. Each single-depository handler has an artifact reader instance associated with each readable artifact of the corresponding depository.

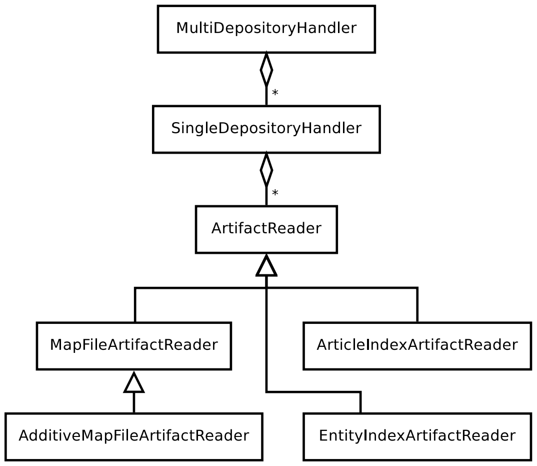


Figure 3.7: A class diagram of the Lydia depository server.

3.6.2 Artifact Readers

The artifact reader base class is responsible for maintaining the reader finite state machine (“not opened”, “opening”, “working”, “updating”, and “closing”) and providing general caching functionality. The specific artifact reader subclasses are responsible for handling the following types of artifacts mentioned in Section 3.5:

- *Readable key-value artifacts.* These artifacts are handled by `MapFileArtifactReader`. This type of reader maintains a set of opened Hadoop `MapFiles` for a set packing of date ranges of the artifact. Inside of every date range directory there is a group of `MapFiles` named `part-0` to `part-`$(n-1)$, each corresponding to the output of one reduce task of the map-reduce job that produced the date range directory. For all date ranges in the covering set, or in the subset of the covering set that is sufficient to answer the current request, the reader uses the same hash function by which keys were originally split between reduce tasks to identify the part containing the given key. Then the appropriate `part-`i `MapFile` is queried with the given key (such as entity name), yielding a list of values, at most one per date range. For most artifacts `AdditiveMapFileArtifactReader`’s functionality is then invoked which sums up those retrieved values using the polymorphic addition operation introduced in Section 2.6.1, and returns a single value. In other cases, e.g. for the EXACT_DUP_ARTICLES artifact, the first valid value from the list is chosen and returned, because the result list is not expected to contain more than one value.

- *Entity name Lucene indexes.* These artifacts are handled by `EntityIndexArtifactReader`. Entity indexes are relatively small (only 4 GB for our 74-million entity U.S. daily news dataset). Therefore, we download an entity index from HDFS to the local file system every time it is updated and access it using the standard Lucene [8] API.

- *Full-text article Lucene indexes.* This type of artifacts is handled by `ArticleIndexArtifactReader`. An artifact of this type corresponds to a full-text Lucene index of all articles present in the depository with the additional metadata to facilitate advanced search capabilities described in Section 2.7. The reader class handles queries by forwarding them to servers of the Katta [3] distributed Lucene infrastructure library running on our cluster.

3.6.3 Time Series Filtering by Source and Time

The Lydia depository server is capable of returning time series aggregated on a variety of time scales and document source granularities. The depository server API includes two frequently used structures to specify these parameters:

- *Time filter.* This structure contains:
  - Time scale: daily, monthly, or yearly.
  - Time period specified by its first and last time point (day, month, or year, depending on time scale).
  - Aggregation period length in the units of the given time scale. For example, to obtain weekly time series, a daily filter with a period length of 7 is used.
- *Source filter.* This structure contains:
  - A boolean flag indicating whether to return statistics for all document sources.
  - A set of document source IDs of interest. Used if the above “all sources” flag is false.
  - A boolean flag indicating whether to aggregate the returned data across all sources included. If this is false, detailed data is returned for each source.

3.7 Large Scale Entity Co-reference Resolution

An important component of the Lydia system is a named entity co-reference resolution engine. An entity may be referred to using many different names in a text corpus. For example, George W. Bush may be variously referred to as "George W. Bush", "George Bush", "Bush", or even "Dubya." While NLP techniques can be used to perform co-reference resolution within the scope of a news article, associating an entity with its synonymous names is a difficult problem. This is particularly true for text corpora in which the number of entities makes pairwise similarity comparison infeasible.

As part of the new Lydia analysis architecture, we have created a co-reference resolution engine capable of computing likely co-referential entities in an unsupervised way. The previous version of the Lydia system used a co-referential entity identification algorithm based on morphological and contextual similarity and clustering [63]. This algorithm, however, was designed to run on a single machine and was not immediately parallelizable. To handle over 74 million entity names in our U.S. daily news corpus, we implemented co-reference resolution as a sequence of map-reduce jobs on top of the Lydia depository framework. This engine also allows the seamless introduction of known co-referential entities hand-annotated by a human user.

The new Lydia co-reference resolution engine works in three main steps.

- Entities are hashed into groups using logic dependent on their type, as determined by earlier phases of the Lydia system.
- Groups of entities with compatible hash types are then merged.
- The resulting larger groups of entities are refined into smaller groups, and finalized.

We will discuss each of these phases in turn.

3.7.1 Entity Hashing

The primary work done in the co-reference resolution system consists of a series of entity type-dependent hashes. That is, if an entity is a person, there are a very different set of morphological traits which imply co-reference than if the entity is an organization. Each hash type attempts to reduce the entity to a core form which should be common across a set of possibly co-referential entities. While in principle extensible to a variety of other types, there are four primary types of hashes implemented in the Lydia co-reference resolution system.

3.7.1.1 Default morphological hashing

All entities are hashed by removing case, whitespace, and punctuation. This hash is simply designed to catch variants of an entity where we cannot apply more specific type-based knowledge to aid us. In general, all other hashes perform these simple reductions in addition to their more specific hashing logic.

3.7.1.2 Person hashing

This type of hashing is performed for entities which the Lydia system has classified as a person. The hash optionally performs a variety of simplifications to the name, such as removing suffixes (e.g., III, Jr., Sr.), removing middle names, and a variety of steps to eliminate likely entity markup failures (e.g., trailing punctuation and possessives). Thus, the resulting hash for the set of entities “George Bush,” “George W. Bush,” and “George H.W. Bush” should be “georgebush.”

The hash also reduces nicknames (based on a predefined list) to their base form (e.g., “Ted” becomes “Edward”). There are a significant number of cases in which nicknames may be ambiguous. For example, the name “Alex” may refer to “Alexandra” or “Alexander.” In these cases, all possible hashes are emitted. In a similar vein, the system may also optionally generate abbreviated forms of the name for inclusion in the list of hashed elements (these will be later removed if they do not appear in the corpus). For example, “George Bush” might be abbreviated “G. Bush” or simply “Bush.”

We note that both of these manipulations can result in an entity being present in many different hash buckets. However, as discussed later, we will later merge buckets and perform a contextual refinement of groupings, eliminating groupings of person entities which are only morphologically similar. Thus, while it is possible for an entity to be determined to be associated with multiple distinct entities, these entities will eventually be merged together and then refined into smaller groups. Of course, like most behaviors in the co-reference resolution engine, abbreviation generation and nickname resolution are optional.

3.7.1.3 Company hashing

This type of hashing is performing for entities which the system has classified as companies. The hash primarily removes suffixes associated with companies. That is, “Wal-mart”, “Wal-Mart Incorporated”, and “Wal-Mart Inc.” will all be hashed to “walmart”.

3.7.1.4 Metaphone hashing

Metaphone [77] is a “sound-alike” hash system which generates a hash based on the pronunciation of a word. Hence, two words will have the same metaphone hash if they “sound alike.” We directly use the Double-Metaphone implementation provided by the Apache Commons library, and apply the hash only to entities classified as people. The reason for this is simply that two arbitrary entities with similar pronunciations are not likely to be similar, while person entities undergo a separate later stage of refinement wherein erroneous relationships will be purged. The benefit of this stage, however, is that names such as “Rachmaninoff” and “Rachmaninov” will be placed together in a metaphone hash, while this similarity will be entirely missed by other types of hashes.

3.7.2 Merging of Hashed Groups

The results of the various hashing stages described above is a set of entity groupings. We can now combine these groupings into larger groups, where appropriate, by application of a disjoint-forest merge algorithm. This algorithm is implemented as a serial program in C++, for reasons both of efficiency and the theoretical difficulty of implementing such an algorithm as a Map-Reduce job.

In our current implementation, the default morphological hash is combined with the company hash, while the metaphone hash is combined with the person hash. The result of these merges is two sets of entity groupings, one consisting of person entities and the other all remaining entities. For non-person entities, the processing is now effectively complete. However, we perform further refinement on the large groupings of similar names.

3.7.3 Refining Entity Clusters Using Juxtapositions

We now have the problem of examining a large set of names and determining which are likely coreferential entities, and which are not. Such a group may include entities which are largely similar, such as “George Bush” and “George W. Bush,” and entities which can be largely dissimilar, such as “Edward M. Kennedy” and “Theodore R. Kennedy” (here, for example, linked because of the shared nickname “Ted”).

To refine these relatively large groups of potentially co-referential entities into smaller likely co-referential sets, we examine the juxtapositions associated with these entities. In order to compare the similarity between two juxtaposition lists, we essentially treat the set of juxtapositions for each entity as a unit vector, and apply some measure of vector similarity, such as cosine distance. However, a number of different measures can be used to determine the similarity between two vectors, and in practice we use a simple, if ad-hoc, metric which places significant weight on the overall size of the set of mutually juxtaposed entities.

With these similarity measures computed, we perform an iterative clustering of entities, examining each pair of entities in order of similarity. Each entity in this pair is associated with some cluster of entities (initially only itself), and we decide whether to merge these clusters based on the (1) overall similarity between all pairs of entities in the two clusters, and (2) the morphological compatibility of elements within the two clusters. By (2) we mean that we require significantly more similar juxtaposition lists to merge entities such as “George W. Bush” and “George H. W. Bush” than to merge “George Bush” and “George W. Bush.” The overall similarity required for these merges to occur is a user-defined parameter, tunable to provide more or less aggressive (and hence less or more accurate) clustering behavior.

3.7.4 Hand-annotated Co-referential Entities

As no unsupervised entity co-reference system will ever be completely accurate, the system must be able to easily incorporate hand-annotated co-referential entities. The system described above allows for such annotations by allowing the user to specify groups of entities which should appear as co-referential in the output. The system does this by forcibly merging the sets containing these entities during the hash group merging phase (and additionally, overriding the type of the entity, if it was misclassified). Furthermore, during the refinement phase, pairs of entities which were hand-annotated as co-referential are treated as having perfect similarity for the purpose of clustering. Thus, the resulting clustering will necessarily have all hand-annotated co-referential entities, but also other entities determined by the system to be co-referential. For example, if the user specifies that "Al Gore" and "Gore" should be co-referential, the system may still determine that "Albert Gore" should also be in this cluster.

3.8 Group Statistics Aggregation

The ability to efficiently aggregate statistics over a large number of potentially large groups of entities is an important capability for the Lydia system. The most common usage of this is that statistics for co-referential entities (e.g. George Bush, George W. Bush, Bush) should be aggregated to allow for combined analysis. Additionally, this ability enables large scale analyzes of groups, such as the distinctions in news coverage based on Cultural/Ethnic/Linguistic groups discussed in Chapter 4 and [92].

However, implementing efficient group aggregation in a Map-Reduce system is somewhat non-intuitive, due to the inability to perform random access queries to determine group membership. That is, although it is trivial to implement group aggregation given that each mapper has access to the complete random-access group membership information for every entity, this assumption is not scalable. The size of our U.S. dailies dataset alone makes it infeasible to store such a table locally to each mapper, while distributed solutions such as *memcached* proved relatively slow (as well as a drain on clusterwide memory resources). Thus, we must aggregate statistics across groups without relying on the ability to make any membership queries directly, but only as part of the key-value inputs to a Map-Reduce job.

We will first discuss the case in which we wish to aggregate key-value pairs of the form ENTITY_NAME : STATISTIC (we make no assumptions as to the content of the statistic, but that it can be meaningfully aggregated in a commutative and associative way). We will then discuss the more interesting situation that occurs when we wish to aggregate statistics keyed to pairs of entities, such as with juxtaposition time series.

3.8.1 Aggregation of Single-keyed Entity Statistics

Group aggregation of an artifact keyed by a single entity can be performed as two Map-Reduce jobs, the second simply being an additive reduction. To begin, there are two types of input key-values given as input to the job: (1) groups, with an entity name as the key and a list of groups the entity belongs to as the value, and (2) statistics, with the entity name as the key and a commutatively additive statistic as the value. Note that if the groups are given as group to entity mappings (rather than entity to group) we can easily invert the mapping in a single job. The map phase of the job behaves as follows:

- ENTITY : $Group_1, \ldots, Group_n \rightarrow$
  ENTITY : $(null,\ Group_1, \ldots, Group_n)$
- ENTITY : STATISTIC $\rightarrow$
  ENTITY : (STATISTIC, $null$)

Then, the value set of each key sent to the reduce phase will consist of a set of groups and a set of statistics. For each, the reducer will sum the set of statistics, and for each group emit GROUPNAME : SUMMEDSTATISTIC. A final additive reduction will then merge the emitted statistics for each group, resulting in a fully aggregated artifact.

3.8.2 Aggregation of Pair-keyed Entity Statistics

Aggregation across groups in which the statistic corresponds to a pair of entities is considerably more involved than the single entity case. First, let us consider what we mean by aggregation in this situation. Suppose that we have two entities, E_1 and E_2, such that E_1 belongs to groups G_1 and G_2, while E_2 belongs to groups G_2 and G_3. Correct aggregation of a statistic over the pair E_1 and E_2 will attribute the statistic to all group pairs (G_1, G_2), (G_1, G_3), (G_2, G_2), and (G_2, G_3) (and, of course, their symmetrical counterparts). Another consideration of the system is whether entities which do not appear in any group can appear as one or both entries in the pairs of the resulting output. Finally, one technical consideration is whether the input includes both orderings for pairs, and whether we wish the final output to include data for both orderings. As we shall see, both of these considerations can be easily dealt with.

In order to perform this aggregation, we use three Map-Reduce jobs, one of which is an additive reduction. The first job proceeds as follows:

Map phase #1:

```
• ENTITY : $Group_1, \ldots, Group_n \rightarrow$
        (ENTITY, $null$) : ($null$, $null$, $Group_1, \ldots, Group_n$)

• (ENTITY$_1$, ENTITY$_2$) : STATISTIC $\rightarrow$
  IF ENTITY$_1 \leq$ ENTITY$_2$
        (ENTITY$_1$, ENTITY$_2$) : (STATISTIC, $null$, $null$)
        (ENTITY$_1$, $null$) : ($null$, ENTITY$_2$, $null$)
  IF ENTITY$_1 >$ ENTITY$_2$
        (ENTITY$_2$, ENTITY$_1$) : (STATISTIC, $null$, $null$)
        (ENTITY$_2$, $null$) : ($null$, ENTITY$_1$, $null$)
        (ENTITY$_1$, $null$) : ($null$, MARKER, ENTITY$_1$)
        (ENTITY$_2$, $null$) : ($null$, MARKER, ENTITY$_2$)
```

Reduce phase #1:

```
• (ENTITY$_1$, ENTITY$_2$) : (STATISTIC, $null$, $null$) $\rightarrow$
        (ENTITY$_1$, ENTITY$_2$) : (STATISTIC, $null$, $null$)

• (ENTITY$_1$, $null$) : ($null$, ENTITYLIST, GROUPLIST) $\rightarrow$
  FOR EACH ENTITY$_2$ IN ENTITYLIST
        (ENTITY$_1$, ENTITY$_2$) : ($null$, $null$, GROUPLIST)
        (ENTITY$_2$, ENTITY$_1$) : ($null$, $null$, GROUPLIST)
```

In essence, this first job works to merge the individual group data from each entity with the set of pairs that each entity has a paired statistic with. This phase also generates stand-in groups for entities which do not appear in any groups, so that their pairwise data can be preserved later, if desired. The output of this job is then additively reduced, resulting in each keyed pair containing a data structure with all relevant information about the set of groups each entity is in, as well as the statistic. The final map-reduce job simply emits statistics based on pairs of groups listed for each pair of entities, based on the policy regarding single-entity groups and desired pair-wise orderings, and then additively reduces to obtain the desired result.

3.9 Performance

3.9.1 Comparison with the Old Lydia System

The new Lydia architecture is significantly more scalable than the old database-driven version of the Lydia system [62]. For example, in our experiment processing one day of U.S. daily news data using the old Lydia system took 9 hours 12 minutes on one machine. Processing the same corpus from scratch with the new Lydia system on an 18-node cluster takes 1 hour 55 minutes, representing a speedup of 4.75, or an efficiency of 26% (where 100% would correspond to linear scalability). However, scaled up up to a four-day corpus, the old system met with significant bottlenecks in the DBMS and took 7 days and 33 minutes to complete the processing, while the new system took only 7 hours and 38 minutes on our 18-node cluster to perform NLP processing and build a depository. This already demonstrates a speedup of over 22, and the advantage of our new system over the old one is even greater on larger corpora. Note that the functionality of the new Lydia architecture does not completely replicate the architecture, but offers additional features not present in the original system.

3.9.2 Scalability

Figure 3.8 (left) shows the total amount of time taken by the Lydia depository build process for a 160 MB compressed input data set split between four days to simulate daily updates to the depository. For such a small amount of data it is unsurprising to see that with more than 8 nodes the communication overhead starts to outweigh the benefit of additional computation power. A similar pattern holds for a slightly larger 1.2 GB corpus in Figure 3.8 (right). This shows that it is important to select

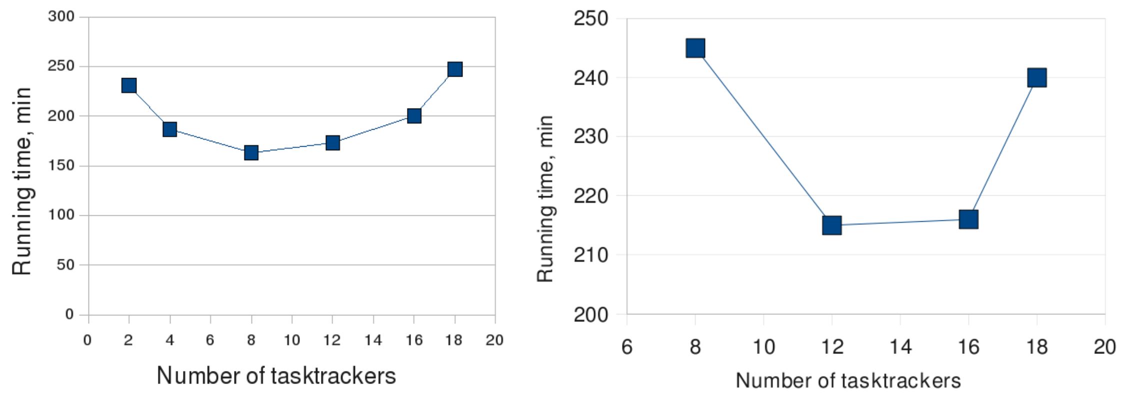


Figure 3.8: The amount of time taken to build a depository depending on the number of nodes used. The input data size is 160 MB (left) and 1.2 GB (right), split between four daily updates.

Number of nodes	Time taken, min
8	849
17	489

Table 3.1: Amount of time taken to build a depository on a 10 GB dataset with varying number of nodes.

the optimal number of nodes for map-reduce workloads that run on small datasets such as daily updates to a Lydia depository.

For large corpora, however, an increase in cluster size translates clearly into a performance increase. As Table 3.1 shows, for a corpus approximately 10 GB in compressed size there was a speedup of 1.73 going from 8 to 17 nodes, corresponding to an efficiency of 81.6%. This confirms our explanation of the non-monotonicity of the running time as a function of number of nodes on small corpora.

Depository update times heavily depend on the existing depository size because of the artifacts that require full merging on every update, such as the CATEGORY_REFS artifact containing entity names with associated categories and frequencies. Figure 3.9 shows incremental update times on a small depository as it grows from two to nine days of U.S. daily news data. Figure 3.10 shows the sum of

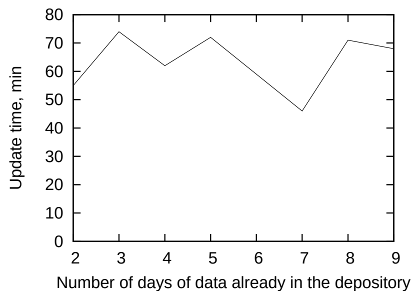


Figure 3.9: Time taken by incremental updates to a U.S. daily newspaper depository with a small number of days.

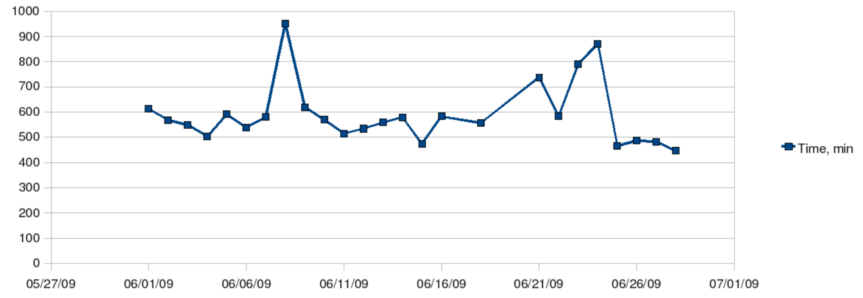


Figure 3.10: Time taken by incremental updates to a four-year U.S. daily newspaper depository.

all artifact build times obtained through analysis of our June 2009 production depository update logs of the four-year U.S. daily newspapers depository. The spikes in Figure 3.10 can be explained by manual changes caused by system maintenance. Even though these update times are still manageable with our current data scale and computational resources, it is clear there is an opportunity for improvement of the update times for large depositories, perhaps by adopting a column-oriented database such as HBase [7] as backend storage.

3.9.3 Alternative Entity Markup

As a test for our architecture we have built a depository on one month of U.S. daily news (May 2009) in a mode where all nouns and noun phrases appearing in WordNet are marked up as entities and the original entity markup from the legacy

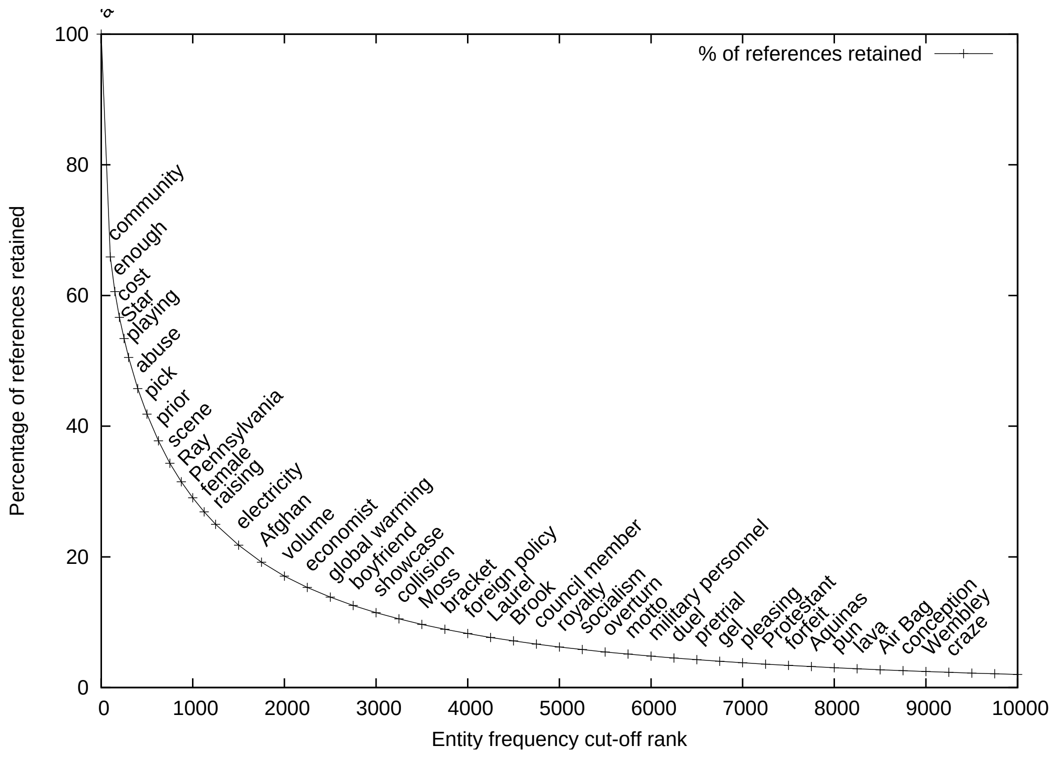


Figure 3.11: The fraction of WordNet noun references in the U.S. daily corpus retained as a function of the number of most frequent nouns ignored. Example words corresponding to various frequency ranks are shown.

	Lydia markup	All WordNet nouns
Time to build, min	282	387
Number of entities	5639436	105340
Example top entities	U.S., flu, Barack Obama, Web, Canada	can, May, people, year, time
Top entity frequency	155814	1108203
Depository size	38.8 GB	22.8 GB

Table 3.2: Amount of time taken to build a depository on one month of U.S. daily news data using normal Lydia entity markup and all WordNet nouns.

Lydia NLP pipeline is discarded. As Table 3.2 shows, this increased processing time by 37% and reduced the number of resulting entities down to a number comparable to the fixed size of the WordNet dictionary. We explain the performance decrease by the extremely high frequency of some of the common nouns, while most of the entities marked up by the NLP pipeline are very infrequent. This is confirmed by the "top entity frequency" line of Table 3.2.

Looking at the top entities in each depository in Table 3.2 we see that the named entities marked up by the Lydia NLP pipeline are much more explanatory of the events, e.g. flu is on the list due to the swine flu epidemic. The top WordNet nouns, however, include May, which is the month when the news articles were collected. The WordNet nouns are a valuable addition to the Lydia entity markup, but the most frequent of them are of little interest. The top 11 of them are even included into a standard list of English stop-words. To identify the best number of top-frequency nouns to ignore, we have plotted the fraction of all WordNet noun references in the news remaining after ignoring all nouns higher than a certain frequency rank (Figure 3.11). Using this graph and the example words, we decided to discard the 1000 most frequent nouns and retain 32% of noun references. Our system allows marking up custom phrase lists such as this while keeping the Lydia NLP pipeline entity markup intact.

This experiment demonstrates that we can switch to alternative notions of what to consider an entity with a relatively low overhead in terms of performance and disk space.

3.10 Conclusion

The new scalable version of our Lydia text analysis system is implemented on top of the Hadoop map-reduce library. The Lydia system takes a text corpus and builds a statistical dataset we call a depository, containing statistics of entity occurrences, juxtapositions, entity and article search indexes. We have implemented a custom workflow management framework on top of Hadoop that allows us to automatically track dependencies between various statistical datasets (artifacts) derived from text contained in a Lydia depository, and unify batch processing of a new corpus with an incremental update of a corpus that has already been processed. Our system also supports scalable cross-document co-reference resolution and entity statistics aggregation across co-reference sets and other entity groups. The results of text processing and entity statistics extraction are available for random access by the web frontend and ad-hoc analysis scripts through our depository server.

The directions for future improvement of our text analysis architecture include:

- Using a column-oriented database such as HBase to store entity statistics. This will allow to eliminate the scalability bottleneck our depository server currently represents.
- Addition of a functionality permitting the user to define custom entities after the depository has been built.
- Further optimization of multiple-map-reduce job workflow scheduling.

Chapter 4

Differences in News Coverage of Cultural/Ethnic Groups [1]

4.1 Introduction

Interactions between distinct ethnic/cultural groups comprise one of the dominant social forces shaping our world. Accurately quantifying the nature of these interactions provides essential data for social science research, in fields as diverse as history, political science, and international relations.

In this chapter, we report on a serious effort to use computational analysis of long-scale text streams (newspapers) to measure differences in the reference frequency and sentiment associated with distinct ethnic/cultural groups. Our methodology is as follows. Using two distinct and orthogonal classification methods (ethnic name analysis and co-reference association), we identify likely ethnicities and nationalities for each person mentioned in the news. By aggregating reference/sentiment counts across all members of a cultural/ethnic group, we obtain strong-enough signals to detect and measure interesting trends.

In particular, we report on the following contributions:

- *Methods for Nationality Detection* – We develop a method to associate primary nationalities for distinct named entities (people) based on a statistical

[1] This chapter is drawn from "Identifying Differences in News Coverage Between Cultural/Ethnic Groups" [92].

analysis of geographic co-locations. Here we validate how this nationality detection methodology and the name ethnicity detection methodology described in [5] can work together to measure ethnic divisions in regional and national contexts.

- *Geographic News Analysis of Cultural Groups* – We demonstrate that our methods coupling ethnicity/nationality detection with large-scale news analysis produce striking and insightful distributions of ethnicity and nationality. In particular, we produce a series of cartograms (skewed data maps) reporting the ethnic distribution of 13 cultural/ethnic (CEL) groups across the world. We present accurate frequency distribution and sentiment maps [66] for CEL groups within the United States, and demonstrate that they accurately reflect survey data collected by the U.S. Census in 2000.

- *Time-Series Trends in CEL Group Frequency and Sentiment* – Through analysis of two extensive news corpora, we detect interesting trends and periodicities in news coverage. Our "dailies" corpus covers roughly 500 U.S. newspapers on a daily basis over the past four years, while our analysis of *The New York Times* spans 27 years, from 1981 to 2008. Interesting trends of where different CEL groups appear within sections of the newspaper (e.g. business, sports, entrainment) are revealed. Our sentiment analysis measures the half-life of major events such as 9/11 in public discourse, as well as the ebbs and flows of smaller events.

- *Inter-Group Interactions and Sentiment* – Our methods generalize to measure the frequency and sentiment co-locations spanning CEL groups, with natural geographic and political associations being revealed through the news data. We can quantify the tenor of interactions between different groups and how they change over time.

Of course, our methods can be applied to text streams such as blogs as well, but we limit our attention here to news data, because of its applicability over greater time spans and the better precision of geographic location information we have for news sources. We anticipate that our data/analysis will be of intense interest to social scientists.

This chapter is organized as follows. We begin with a quick review of related studies of ethnic bias in news coverage, and the *Lydia* text analysis system used to perform named entity recognition and sentiment analysis. We then present our method for nationality detection with evaluation results. Finally, we report on our observed temporal, geospatial, and association trends for all CEL groups.

4.2 Previous Work

4.2.1 Name Ethnicity Detection

We use a method for ethnicity detection based on surname / given name frequencies and k-mer analysis, which is presented in [5]. This method is implemented as a custom processing phase in the new Lydia architecture described in Chapter 3.

4.2.2 Ethnic Biases in Newspaper Coverage

Media bias is of considerable interest within the social sciences. Representative studies include [91], who examined the *Los Angeles Times* coverage of 1241 homicides over a 5-year period, and found that neither the ethnicity of perpetrator nor the victim was associated with the nature of news coverage. However, in related work reviewing the same news coverage, [89] found that the ethnicity of the victim did affect the volume of news coverage. Dixon, et al. found that Whites were over-represented in news coverage of crime, both as perpetrators and as victims [41]. In a study of U.S. newspapers, Johnson [54] found that the coverage of Mexico is influenced by the ethnic makeup of the newspaper's circulation region.

4.2.3 News Analysis Infrastructure

The *Lydia* text analysis system [62] performs named entity recognition and analysis over text corpora. Named entity analysis over large news corpora is extremely useful for applications in a variety of areas, ranging from market research to social science. Particularly interesting is our sentiment analysis [19, 48], measuring the degree to which entities are regarded positively or negatively. In the course of our research, we have collected nearly a terabyte of raw news text spanning more than a thousand U.S. and International newspapers over the course of more than four years.

Large scale studies of newspaper data pose challenging implementation problems for any data mining system. Performing analysis of our one-terabyte newspaper corpus requires a system which is fast and scalable. With the newly implemented Map-Reduce-based framework for statistical analysis, discussed in Chapter 3, the Lydia system is now capable of generating interesting statistics for over 74 million named entities spanning this corpus using an 18-node cluster in 7 days. Among the results of the this process are reference time series, sentiment analysis, and juxtaposition relationships.

For the purposes of this chapter, the Lydia system has been extended to efficiently compute statistics over large groups of entities such as ethnic groups and co-reference sets [63]. This extension is discussed in detail in Section 3.8. Like the remainder of the new system, this aggregation is done using Map-Reduce. As a result, we are able to aggregate hundreds of gigabytes of statistics in 10 hours on the same 18-node cluster.

4.3 Entity Geographical Associations

Beyond the groups formed by our ethnicity classification, another interesting type of culturally defined entity group can be composed from those entities which are closely associated with some particular country. That is, we would like to automatically classify news entities into groups by national identity. In this section we discuss both how we do this, and the results obtained using our method.

4.3.1 Geographic Association

A juxtaposition relationship occurs between a pair of entities which co-appear in sentences within news articles. The strength of a juxtaposition relationship can be assessed simply by the frequency of appearance (juxtaposition count), or by a normalized juxtaposition score based on the relative frequency of both entities. Given two entities in an N sentence corpus which appear in N_1 and N_2 sentences and co-appear in F sentences, the juxtaposition score for these two entities is calculated

Vladimir Putin		Nicolas Sarkozy	
Russia	652,743	France	322,273
Russian	335,563	French	265,513
Moscow, RUS	234,006	Paris, FRA	126,910
U.S.	100,705	China	51,744
Iran	98,324	Russia	53,238
Ukraine	83,574	Iran	47,706
Georgia	70,550	Afghanistan	44,505

Table 4.1: Top geographic juxtapositions (with scores) for two important world leaders.

as:

$$-\log\left(\left(\frac{e^{\frac{FN}{N_1N_2}-1}}{\left(\frac{FN}{N_1N_2}\right)^{\frac{FN}{N_1N_2}}}\right)^{\frac{N_1N_2}{N}}\right)$$

In order to associate an entity with a country, we consider the list and strength of juxtaposition relationships associated with the entity, as reported in [62].

For convenience, the set of interesting geographic juxtapositions we use for this task are the set of countries, national adjectives (e.g., Russian), and international cities already recognized by the geographical normalization engine in the *Lydia* entity recognition pipeline. Table 4.1 shows two world leaders and their highest ranking geographical juxtapositions by juxtaposition score. Because the sources in our corpus are predominantly U.S. daily newspapers (with some additional major Canadian and British sources), we expect a large skew towards associations with places within the United States. To compensate for this, we first compute the juxtaposition rate to each country's set of geographic juxtapositions and use this for normalization. Thus, out of the set of all geographic juxtapositions for a given entity, the nationality with the most statistically improbable set of juxtapositions will be assigned.

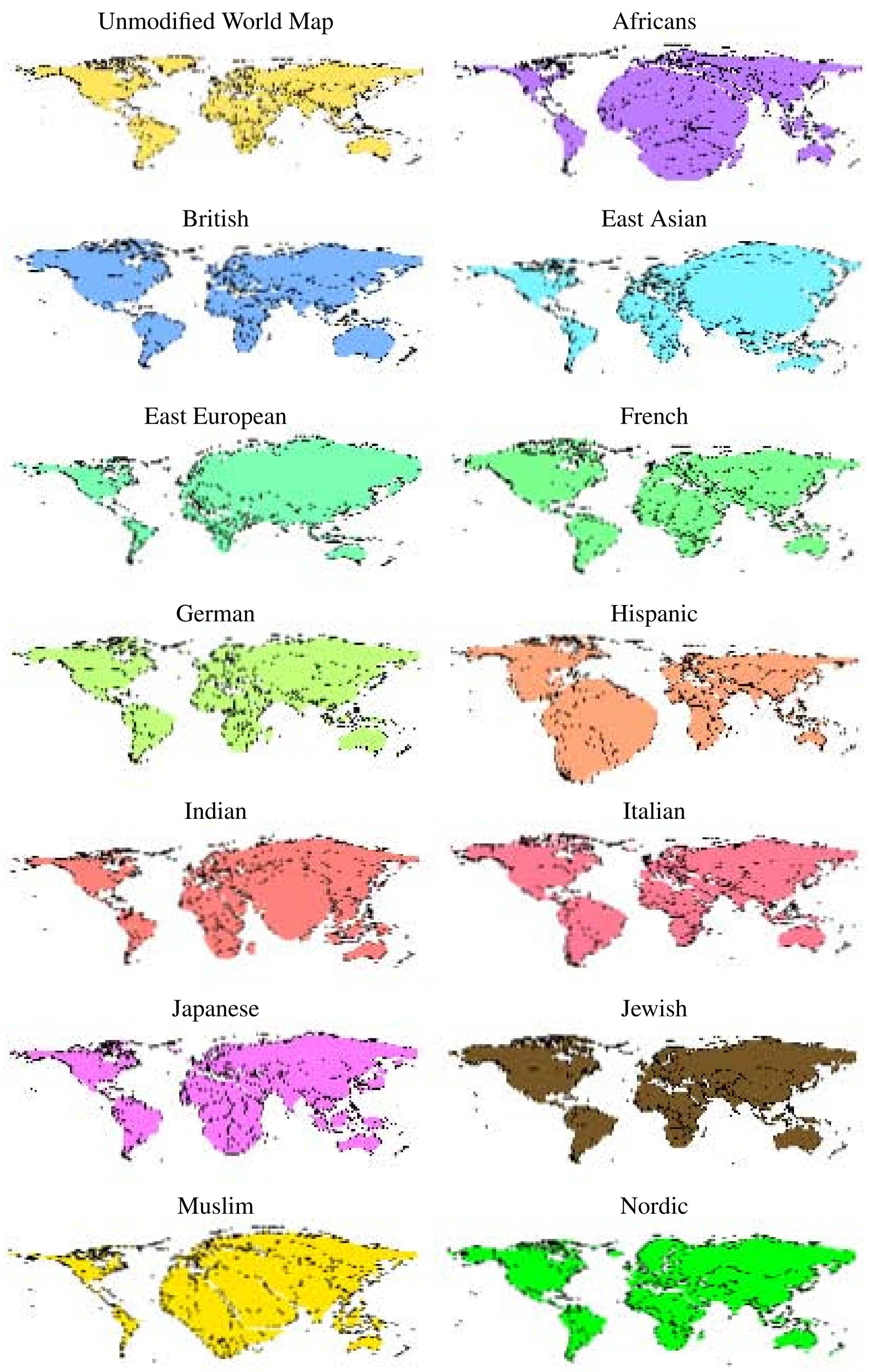


Figure 4.1: CEL groups by nationality cartograms. Enlarged countries reflect higher concentrations of the given CEL group.

	Heads of State	Heads of Gov.
Total	177	192
Correct	116	112
Incorrect (Total)	36	32
(US)	5	2
(Same Region)	16	10
No data	25	48
Precision	0.76	0.78
Recall	0.66	0.58

Table 4.2: Agreement between nationality and dominant geographic association for heads of state/government.

4.3.2 Results and Comparison with Ethnicity Data

Even following normalization, the dominant country of association for 80% of all entities is the United States, which should be expected in the analysis of U.S. newspapers. As validation that the remaining entities are associated correctly, we perform two assessments. First, we determine the country association of the Head of State and the Head of Government for 192 countries. Table 4.2 gives the results of this classification, establishing a high precision and recall.

As a second method of validating the utility of our geographically-associated entities, we analyze their ethnicities using the name-based classifier described in [5]. Figure 4.1 shows cartograms for various CEL groups where the cartogram density of each country is determined by the percentage of associated entities classified as within the ethnic group.

4.4 Trends in Group Coverage

With entity statistics now aggregated by ethnic group, we can examine trends in news volume and sentiment across various ethnic groups. This section provides charts detailing four primary methods of examining CEL group news coverage:

- *Reference volume time series* – the fraction of sentences which contained a person entity which belonged to this ethnic group.

- *Sentiment score time series* – the aggregate sentiment score for all entities of the ethnic group. Sentiment score is a measure of how positively or negatively entities are being discussed, and is based on predefined dictionaries of positive and negative sentiment words. The score for a given entity is simply calculated as:

$$\frac{(pos\_sent\_words - neg\_sent\_words)}{(pos\_sent\_words + neg\_sent\_words)}$$

- *Geographic distribution* – the distribution of news volume of a CEL group by geographic region.

- *Juxtaposition relationships* – the aggregate juxtaposition score between two CEL groups, as computed by the formula given in the section on Geographic Associations.

We will discuss each of these in turn.

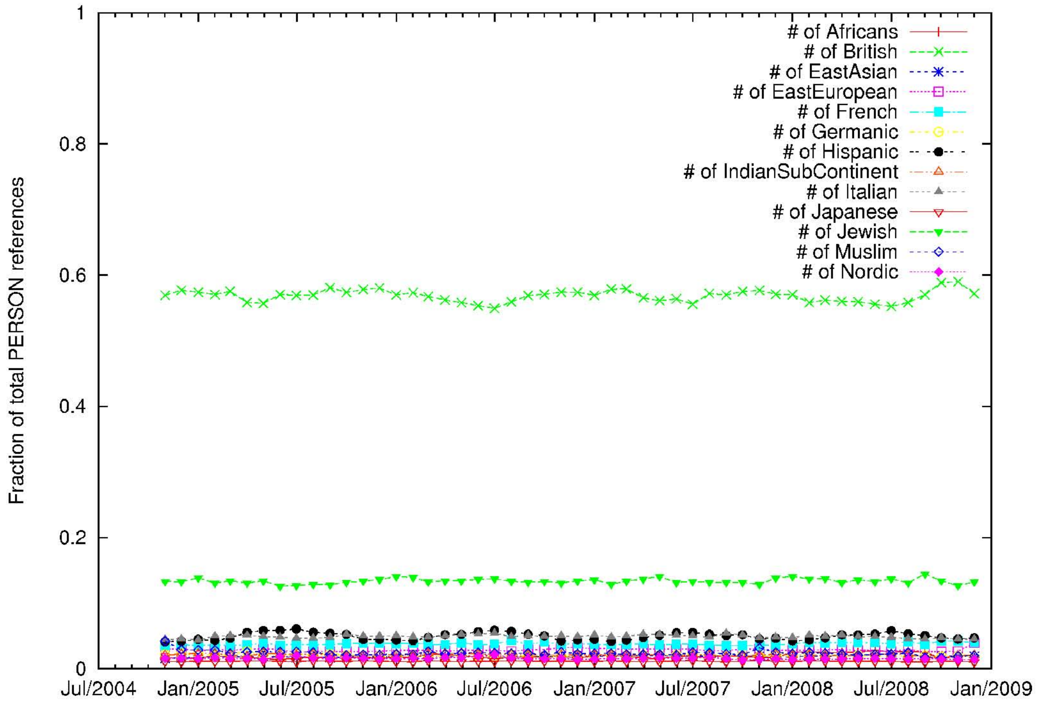
4.4.1 News Volume

Figures 4.2 and 4.3 show aggregate news volume for all CEL groups across four years of U.S. daily newspaper coverage and the *New York Times* (NYT) coverage from 1981–2008, respectively. Interesting features include:

- Coverage of Muslim entities spikes dramatically during the first Gulf War and after September 11th.

- Coverage of Italian entities is significantly larger in the NYT than the national average, consistent with New York State's large Italian population.

- Coverage of Hispanics in the New York Times rose by only a third from 1980 to 2007, while over the same time period the country's Hispanic population roughly doubled as a share of the U.S. population. The Hispanic share of U.S. daily news coverage actually *fell* during the past four years by 14%.



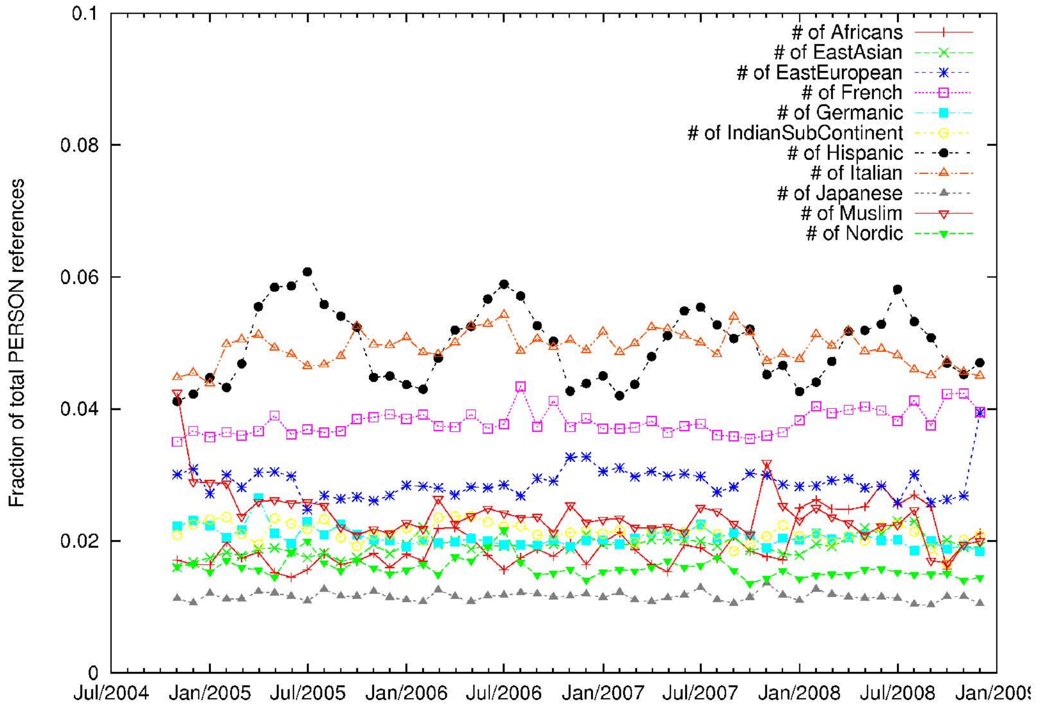
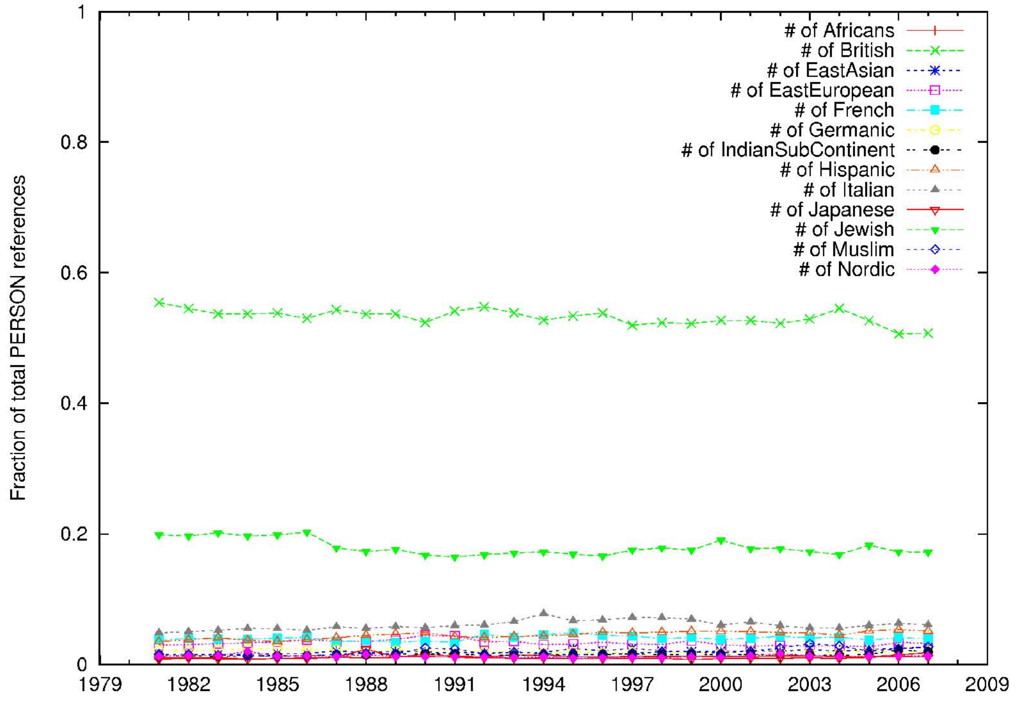


Figure 4.2: Newspaper references to CEL groups in U.S. daily newspapers



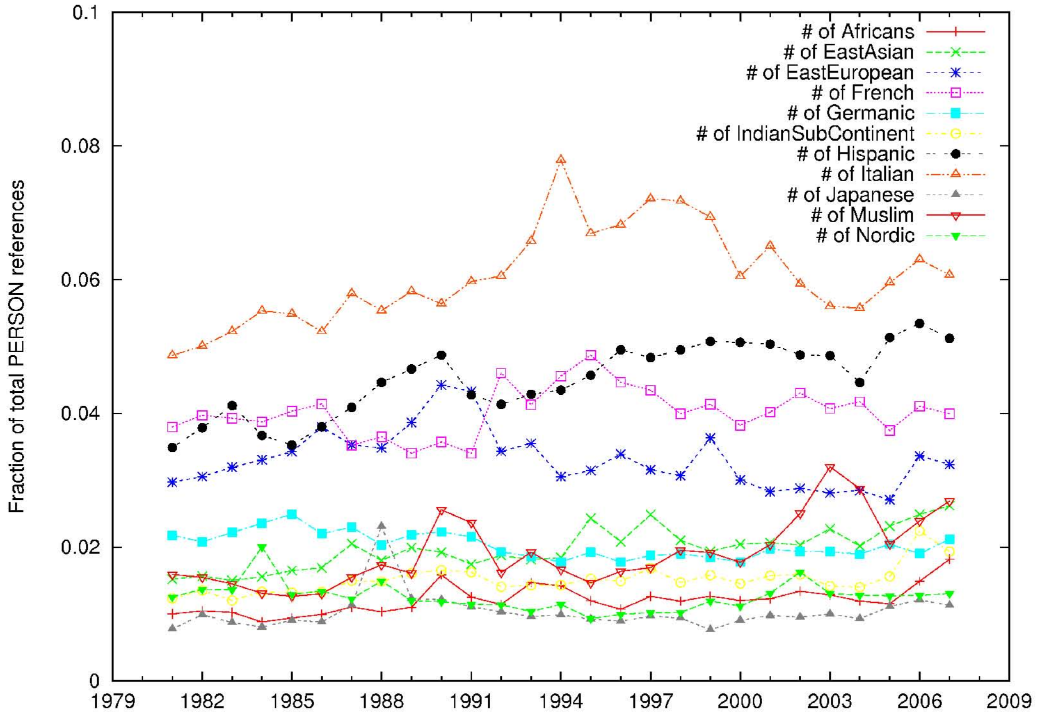


Figure 4.3: Newspaper references to CEL groups in the New York Times

CEL group	Afri.	Brit.	E. As.	E. Eur.	Fren.	Germ.	Hisp.	Ital.	Ind.	Jap.	Jew.	Mus.	Nor.
News	17.4	18.5	17.3	16.5	16.4	17.3	20.6	19.3	17.1	14.7	18.9	17.7	16.8
Business	36.8	19.0	28.4	28.7	23.0	18.4	19.1	20.9	28.5	36.7	20.6	48.3	17.6
Entertainment	19.3	28.3	29.0	25.9	31.5	34.9	21.3	27.7	23.7	24.5	34.1	16.0	31.0
Sports	16.3	23.4	14.7	20.2	19.4	18.8	30.1	20.2	18.7	14.0	13.2	8.5	23.4
Other	10.1	10.9	10.6	8.6	9.7	10.7	8.9	11.8	11.9	10.1	13.2	9.4	11.1

Table 4.3: Type of news coverage by ethnicity

One interesting trend immediately visible from the dailies data is the cyclic nature of coverage of Hispanic entities. The volume of coverage of Hispanics cycles between approximately 4% and 6% of total coverage of entities classified as people, peaking around July and falling to its lowest ebb around December. The reason becomes clear when we break down news volume by news article type and aggregate by month. The seasonal trend in Hispanic news volume results from the disproportionate number of baseball players who are Hispanic.

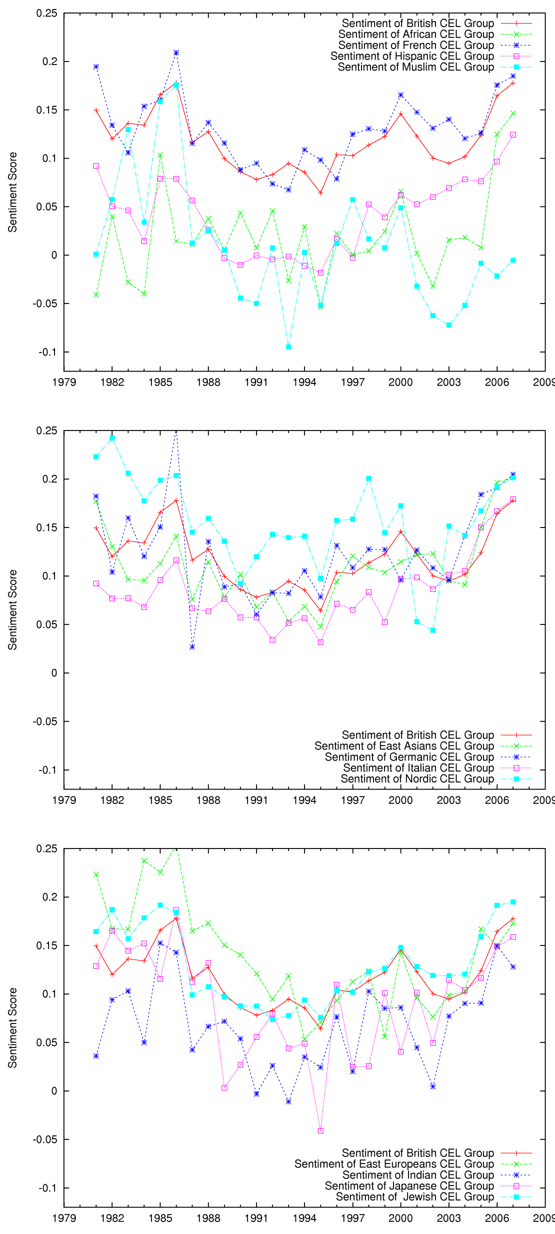
Table 4.3 presents a breakdown by article type for each CEL group. Several ethnicities (e.g. Muslims, Jews, and East Asians) are dramatically underrepresented in sports, with Muslims and Africans underrepresented in entertainment. Many groups show significantly different representation between articles classified as news and business.

4.4.2 News Sentiment

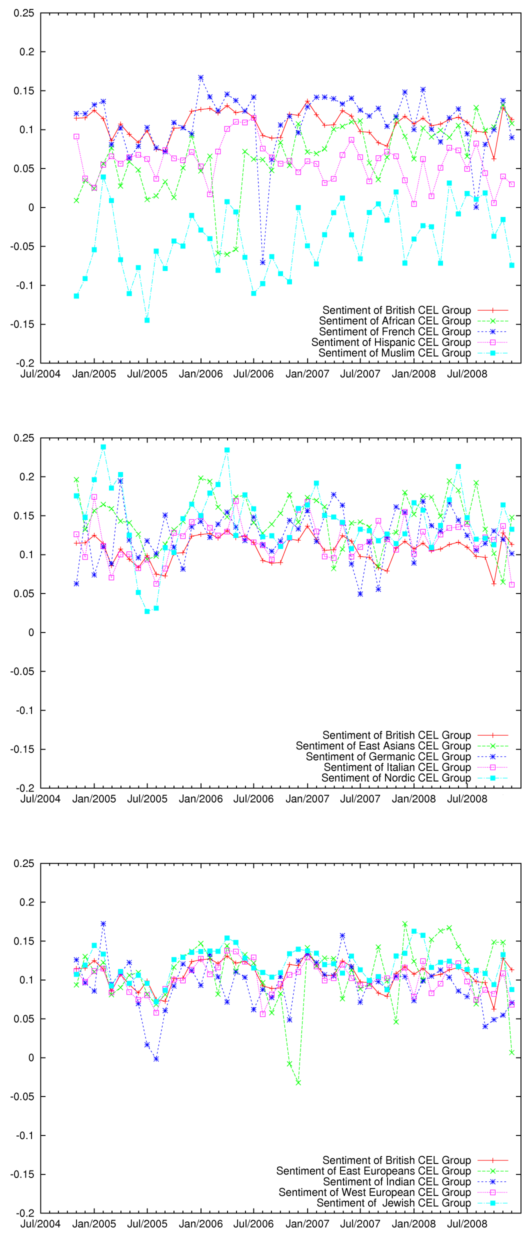
Sentiment analysis broadly measures the tone of text pertaining to news entities. The *Lydia* sentiment analysis system is described in [48]. Figure 4.4 shows time series of sentiment scores for each CEL group as computed by the system. Several interesting trends emerge:

- The majority of ethnic groups do not greatly differ from the baseline sentiment of the British CEL group. Over the 27 year dataset of the NYT, the variance from this baseline tends to narrow – perhaps reflecting improving sensitivities.

- The sentiment of Muslim entities is by far the lowest of any CEL group, with Muslim sentiment being particularly low during the Gulf War, after the World Trade Center Bombing, and for two full years following the September



(a) *New York Times*, 1981–2008.



(b) US dailies, 2004-2008.

Figure 4.4: Sentiment of CEL groups

11th attacks in 2001. Of all CEL groups, only the coverage of the Muslim CEL group is *more negative than positive*, a trend which shows no sign of reversing.

- Hispanic and African sentiment scores are also significantly lower than the baseline. This gap seems to narrow for the African group, though this is substantially due in 2008 to the favorable coverage for Barack Obama. The gap remains relatively constant for Hispanics.

4.4.3 Geographic Biases in News Coverage

Figure 4.6 illustrate the U.S. geographic biases in news volume and sentiment (respectively) for all CEL groups. In general, the size of the local CEL group population heavily influences the news volume of entities from that group. The frequency maps in Figure 4.6 correlate extremely well with maps of ethnic ancestry generated by the U.S. Census [29], particularly with respect to Hispanics. To summarize:

- Large Hispanic populations throughout the Southwest and Florida generate large volumes of local news coverage, which is disproportionally of negative sentiment.
- French populations emerge along the border with Quebec and historically French Louisiana.
- Scandinavian populations throughout Minnesota, Wisconsin, the Dakotas, Montana, and Utah are all reflected.

4.4.4 Juxtaposition Relationships between CEL Groups

Table 4.4 reports the normalized strength of association between pairs of CEL groups derived from news analysis. Under the assumption of independence, the number of collocations between two groups should be proportional to the product of the group sizes. These values have been normalized so 1.0 corresponds to statistical independence. We note that most groups exhibit strong inter-group coherence, with the notable exception of the British CEL group, which presumably reflects both the

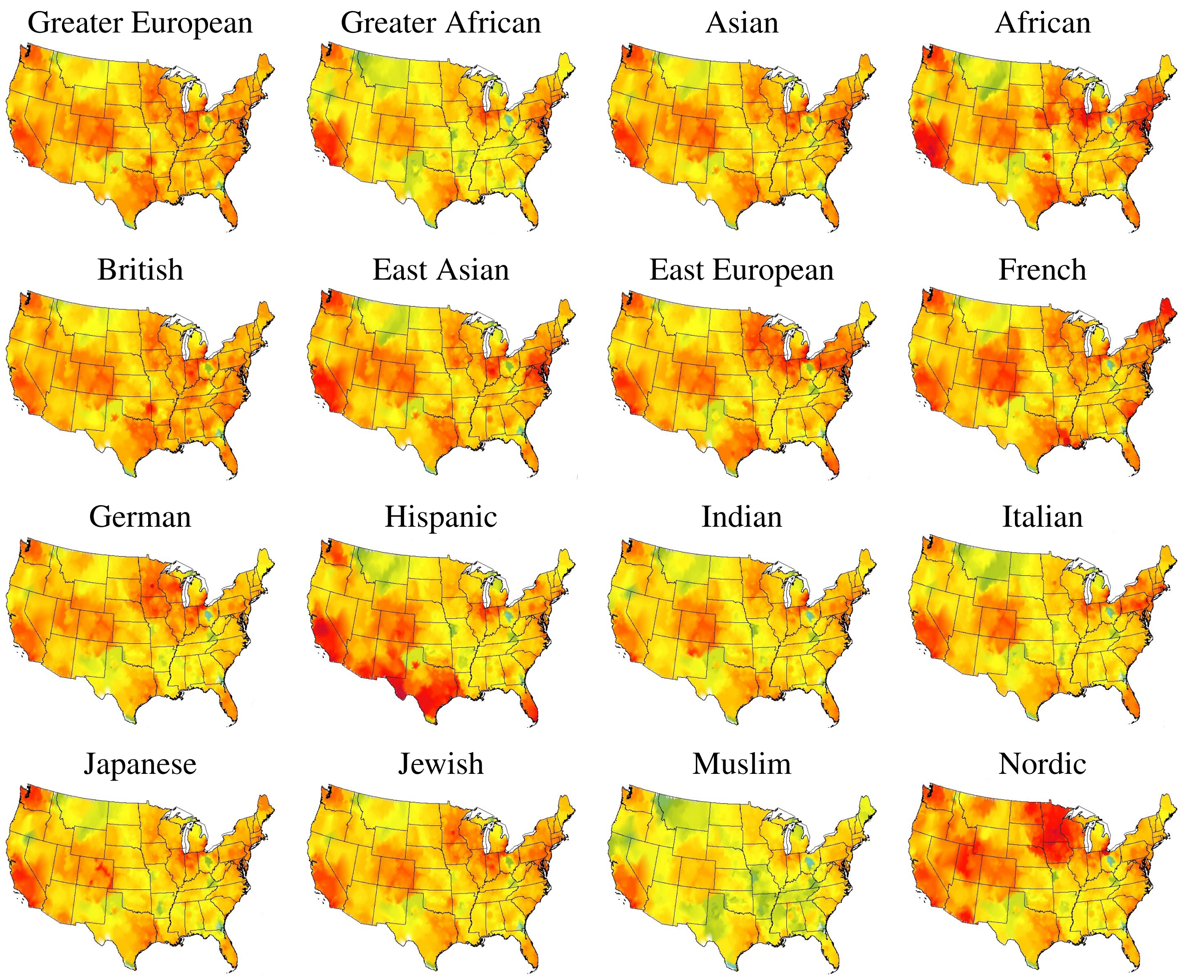


Figure 4.5: Frequency maps for CEL groups within the United States

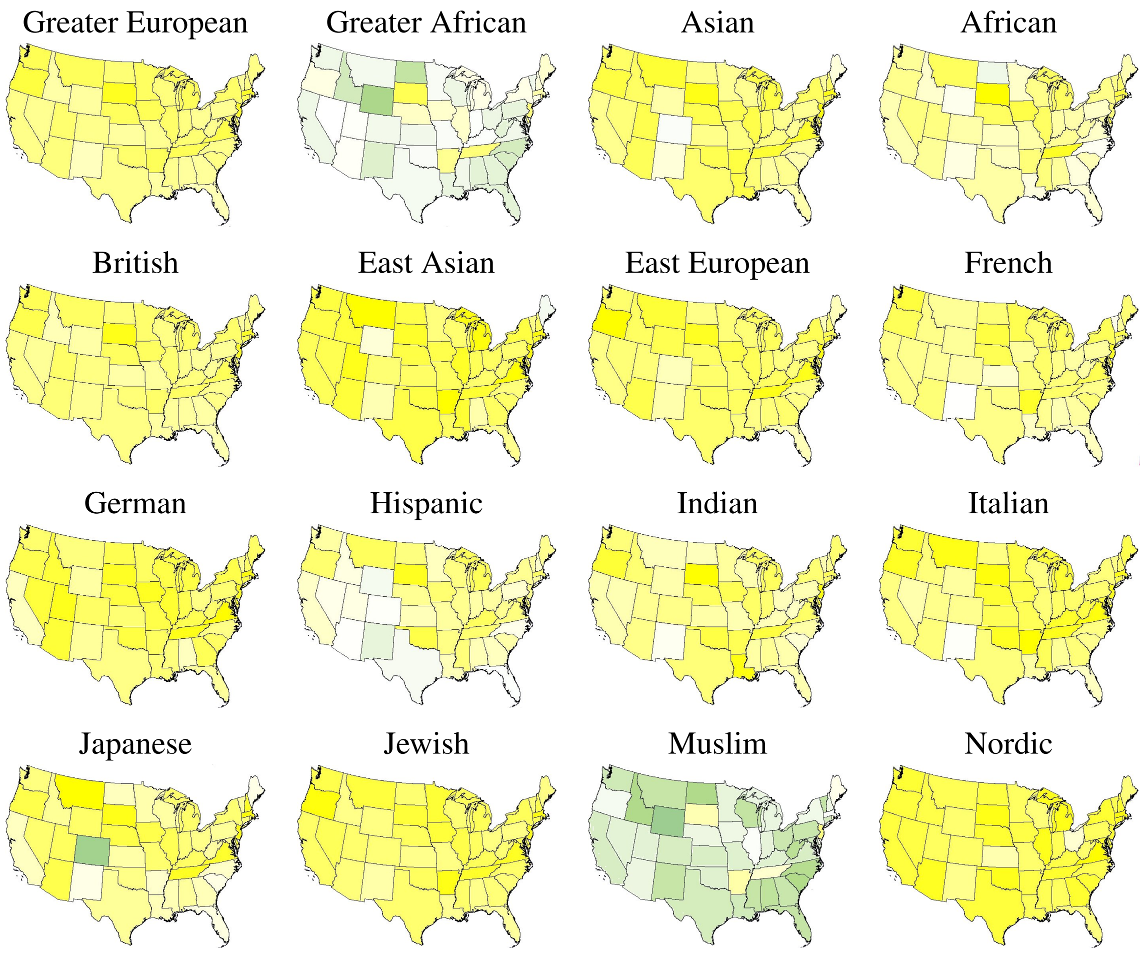


Figure 4.6: Sentiment Maps for CEL groups within the United States

CEL group	Afri.	Brit.	E. As.	E. Eur.	Fren.	Germ.	Hisp.	Ind.	Ital.	Jap.	Jew.	Mus.	Nor.
African	**3.01**	*0.75*	*0.75*	0.84	*0.74*	*0.71*	1.08	1.2	*0.79*	0.89	*0.73*	**1.6**	*0.7*
British	*0.75*	*0.64*	0.88	0.8	0.93	0.96	0.87	0.87	0.9	0.89	0.98	*0.64*	1.07
East Asian	*0.75*	0.88	**3.48**	*0.71*	0.8	*0.75*	0.86	1.13	0.82	**1.26**	0.83	*0.63*	0.9
East Euro.	0.84	0.8	*0.71*	**1.83**	1.1	0.99	*0.79*	0.81	0.8	0.85	0.89	*0.67*	1.16
French	*0.74*	0.93	0.8	1.1	**1.48**	1.15	1	0.96	1.19	*0.79*	0.94	*0.67*	1.18
German	*0.71*	0.96	*0.75*	0.99	1.15	**1.88**	1.01	0.86	0.93	*0.73*	1.17	*0.69*	**1.26**
Hispanic	1.08	0.87	0.86	*0.79*	1	1.01	**2.5**	0.87	**1.28**	1.1	*0.76*	*0.71*	1.09
Indian	1.2	0.87	1.13	0.81	0.96	0.86	0.87	**2.18**	*0.77*	0.99	0.91	**1.46**	1.05
Italian	*0.79*	0.9	0.82	0.8	1.19	0.93	**1.28**	*0.77*	1.16	0.84	0.95	*0.57*	0.97
Japanese	0.89	0.89	**1.26**	0.85	*0.79*	*0.73*	1.1	0.99	0.84	**3.15**	*0.77*	0.85	1.1
Jewish	*0.73*	0.98	0.83	0.89	0.94	1.17	*0.76*	0.91	0.95	*0.77*	0.85	*0.68*	1.07
Muslim	**1.6**	*0.64*	*0.63*	*0.67*	*0.67*	*0.69*	*0.71*	**1.46**	*0.57*	0.85	*0.68*	**2.98**	*0.58*
Nordic	*0.7*	1.07	0.9	1.16	1.18	**1.26**	1.09	1.05	0.97	1.1	1.07	*0.58*	**2.47**

Table 4.4: Strength of juxtaposition relationships between CEL groups. Underrepresented juxtapositions (less than 0.8) are underlined, and overrepresented juxtapositions (greater than 1.2) are boldfaced.

geographically widespread use of English names and the relatively high frequency of other CEL groups in the United States. The Muslim CEL group shows strikingly low rates of interactions with all but the African and Indian communities.

4.5 Conclusions

We have demonstrated that subtle spatial, temporal, and associative trends can be distinguished between cultural/ethnic groups on the basis of aggregate news analysis. We believe our results illustrate the power of our analytic techniques for serious research in several of the social sciences.

We are now working to expand our analysis to a variety of other text corpora and group phenomena (such as influence of gender). Of particular interest is the blogosphere (which should prove even more representative of local sentiment if the postings can be accurately geocoded) and longer-term news archives starting from the 1850's (providing insight into historical trends and cultural forces). Further work in computational methods will revolve around improvements to our ethnicity/nationality classifiers and other group identification techniques (e.g. [67]).

Chapter 5

International Sentiment Analysis for News and Blogs[1]

5.1 Introduction

There is considerable and rapidly-growing interest in using sentiment analysis methods to mine opinions from news and blogs [57, 74, 75, 95, 99, 100]. Applications include product reviews, market research, public relations, and financial modeling.

Almost all existing sentiment analysis systems are designed to work in a single language, usually English. But effectively mining international sentiment requires text analysis in a variety of local languages. Although in principle sentiment analysis systems specific to each language can be built, such approaches are inherently labor intensive and complicated by the lack of linguistic resources comparable to WordNet for many languages.

An attractive alternative to this approach uses existing translation programs and simply translates source documents to English before passing them to a sentiment analysis system. The primary difficulty here concerns the loss of nuance incurred during the translation process. Even state-of-the-art language translation programs fail to translate substantial amounts of text, make serious errors on what they do

[1] This chapter is drawn from our paper "International Sentiment Analysis for News and Blogs" [19].

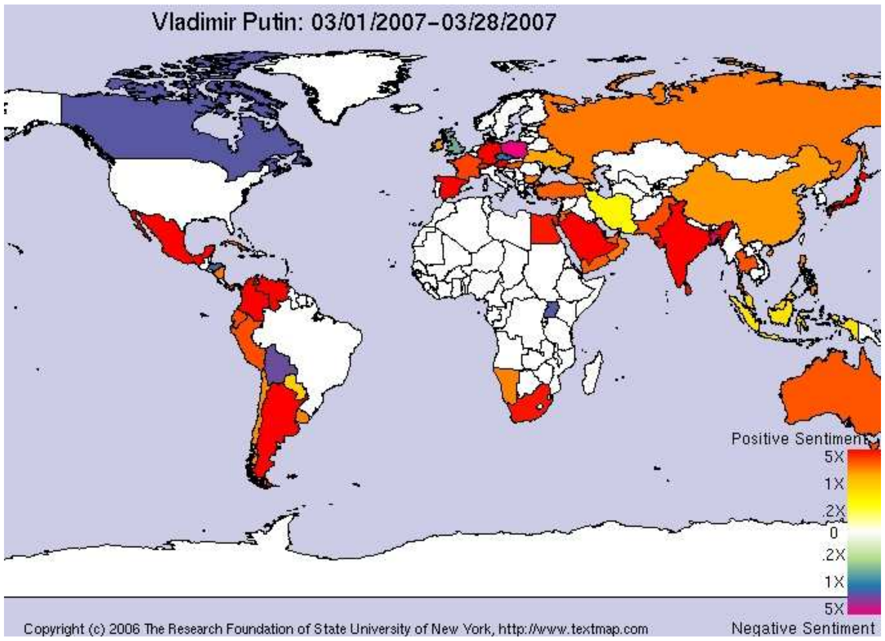

Vladimir Putin: 03/01/2007-03/28/2007
Positive Sentiment
5X
1X
.2X
0
.2X
1X
5X
Negative Sentiment
Copyright (c) 2006 The Research Foundation of State University of New York, http://www.textmap.com

Figure 5.1: An international sentiment map for Vladimir Putin

translate, and reduce well-formed texts to sentence fragments.

Still, we believe that translated texts are sufficient to accurately capture sentiment, particularly in sentiment analysis systems (such as ours) which aggregate sentiment from multiple documents. In particular, we have generalized the Lydia sentiment analysis system to monitor international opinion on a country-by-country basis by aggregating daily news data from roughly 200 international English-language papers and over 400 sources partitioned among eight other languages. Maps illustrating the results of our analysis are shown in Figures 5.1 and 5.2. From these maps we see that (at the time of our research in Spring 2007) George Bush was mentioned the most positively in newspapers from Australia, France and Germany, and negatively in most other sources. Vladimir Putin, on the other hand, had positive sentiment in most countries, except Canada and Bolivia.

Such maps are interesting to study and quite provocative, but beg the question of how meaningful the results are. Here we provide a rigorous and careful analysis of the extent to which sentiment survives the brutal process of automatic translation.

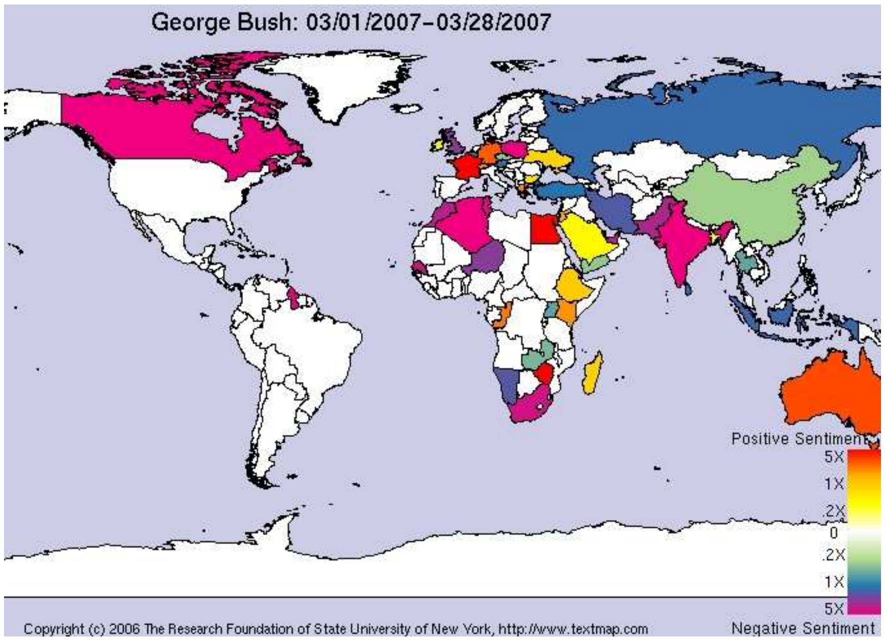

George Bush: 03/01/2007-03/28/2007
Positive Sentiment
5X
1X
.2X
0
.2X
1X
5X
Negative Sentiment
Copyright (c) 2006 The Research Foundation of State University of New York, http://www.textmap.com

Figure 5.2: An international sentiment map for George W. Bush

Our assessment is complicated by the lack of a "gold standard" for international news sentiment. Instead, we rely on measuring the *consistency* of sentiment scores for given entities across different language sources. Previous work [47] has demonstrated that the Lydia sentiment analysis system accurately captures notions of sentiment in English. The degree to which these judgments correlate with opinions originating from related foreign-language sources will either validate or reject our translation approach to sentiment analysis.

In this chapter we provide:

- *Cross-language analysis across news streams* – We demonstrate that statistically significant entity sentiment analysis can be performed using as little as ten days of newspapers for each of the eight foreign languages we studied (Arabic, Chinese, French, German, Italian, Japanese, Korean, and Spanish).

- *Cross-language analysis across parallel corpora* – Some of difference in observed entity sentiment across news sources reflect the effects of differing content and opinion instead of interpretation error. To isolate the effects of news source variance, we performed translation analysis on a parallel corpus of European Union law. As expected, these show greater entity frequency conservation than variable sources. One does not expect impassioned sentiment to be revealed in legal codes, yet these results also show meaningful sentiment consistency.

- *Analysis of translation system-specific artifacts* – The quality of our sentiment analysis will depend on the quality of the language translation software, but how strongly? We compare the sentiment results on the same source text corpus across two distinct Spanish translation systems. Aggregated entity frequency, sentiment polarity, and sentiment subjectivity were highly correlated across both translation systems, with results statistically significant beyond $p < 0.001$. We conclude that the success of our methods is largely (but not completely) translation system independent.

- *Normalizing for cross-cultural language effects* – Translation system/language effects complicate the problem of comparing entity sentiment across distinct language sources. Certain languages (e.g. Chinese and Korean) appear to produce substantially higher sentiment scores than others (e.g. Italian). We present techniques to correct for such bias, and present an interesting cross-cultural comparison of country sentiment by language.

This chapter is organized as follows. We review previous work on foreign language sentiment analysis in Section 5.2, where we also provide an overview of the Lydia sentiment analysis system. The experimental methodology underlying our work is presented in Section 5.3. Sections 5.4-5.6 present our analysis on the consistency of sentiment over corpora designed to isolate the effects of news variance, language variance, and translation system variance respectively. Issues associated with comparing sentiment across languages are presented in Section 5.7. Our conclusions follow in Section 5.8.

5.2 Related Work

There has been a wide research effort in analyzing sentiment in languages other than English by applying bilingual resources and machine translation techniques to employ the sentiment analysis approaches existing for English. We survey that literature below. Subsequently, we describe the approach to sentiment analysis implemented by the Lydia system which we are using for our experiments.

5.2.1 Cross-language Sentiment Analysis

The approach taken in [51] uses machine translation technology to develop a high-precision sentiment analysis system for Japanese at a low cost. Sentiment unit polarity extraction precision of 89% is reported.

Mihalcea et al. [68] discuss methods to automatically generate a subjectivity lexicon and subjectivity-annotated corpora for a new language (they focus on Romanian) from similar resources available for English. They achieve a 67.85 F-measure for classifying sentiment orientation of sentences using the subjectivity resources built for Romanian.

Yao et al. [98] propose a method of determining sentiment orientation of Chinese words using a bilingual lexicon and achieve precision and recall of 92%.

Benamara et al. [20] argue that adverbs in combination with adjectives are more helpful for sentiment score assignment to individual sentiment units than adjectives alone. Their best algorithm achieves a 0.47 Pearson correlation with human-assigned scores compared to 0.34 without using adverbs.

The Oasys 2.0 opinion analysis system [31] allows the user to identify the intensity of opinion on any topic on a continuous scale, and view how that intensity is changed over countries, news sources, and time. It is based on aggregation of individual positive and negative references identified using approaches described in [20, 30] which have been evaluated on the individual sentiment unit level. Our work, in contrast, focuses on the evaluation of the final entity sentiment score rather than individual entity reference polarity.

5.2.2 The Lydia Sentiment Analysis System

Our international sentiment analysis work is based on the first version of the Lydia text analysis system [47,56,62,63,66], because it was done before the new scalable Lydia architecture described in Chapters 2 and 3 was developed. The Lydia system recognizes named entities in text and extracts their temporal and spatial distribution.

Text sources are spidered daily by customized website scrapers that convert articles to a standard format and store them in an archive. Then, on a daily basis, the articles are run through a pipeline that performs part-of-speech tagging, named entity identification and categorization, geographic normalization, intradocument coreference resolution, extraction of entity descriptions and relations between entities, and per-occurrence sentiment score calculation. The entities are then inserted into a database, and cross-document coreference resolution, entity juxtaposition score and per-entity sentiment score calculation take place.

Sentiment score calculation in Lydia is described in [47]. As a preliminary step, the sentiment lexicon is constructed. Starting from sets of seed positive and negative adjectives, their polarity is propagated through WordNet [69] synonym and antonym links, and every adjective is assigned a polarity score. Then, the top fraction of adjectives from both extremes of this curve are placed into positive and negative parts of the sentiment lexicon respectively.

The next step is entity sentiment calculation in a specific corpus. Using the existing sentiment lexicon, positive and negative word occurrences are marked up in the corpus. For every entity and every day i, the number of positive and negative sentiment words co-occurring with that entity in the same sentence ($pos\_sentiment\_refs_i$ and $neg\_sentiment\_refs_i$) are calculated. For every entity, its polarity score on a given day i is then calculated as

$$entity\_polarity_i = \frac{pos\_sentiment\_refs_i}{total\_sentiment\_refs_i} \tag{5.1}$$

and its subjectivity score as

$$entity\_subjectivity_i = \frac{total\_sentiment\_refs_i}{total\_occurrences_i}. \tag{5.2}$$

The polarity score reflects whether the sentiment associated with the entity is positive or negative, and the subjectivity score—how much sentiment of any polarity the entity receives. These are the two measures of entity sentiment that we use in our analysis.

5.3 Methodology

We spider online newspapers in nine languages: Arabic, Chinese, English, French, German, Italian, Japanese, Korean, and Spanish. In our experiments we used 7 to 39 newspaper sources for each language, with the fewest sources for Chinese and Italian, and 21,000 articles per language on average. We translate foreign text to English using IBM WebSphere Translation Server (WTS) [52, 53]. For Spanish and Arabic, we also used a newer experimental translation system hosted as a web service by IBM Research.

We noticed that for many words that WTS is unable to translate to English it leaves them in the output text in the original language. We conjectured that a higher quality translation system would leave a lower fraction of text untranslated, and compared source text with translation system output using a maximum overlap dynamic programming algorithm at the word level. Higher values of this overlap indicate larger numbers of words that did not get translated. This is particularly important to us because we need entity names to be translated correctly to English to be able to match them across language boundaries. Table 5.1 shows these overlap values for different languages, along with the ratio of translation system output to input size, averaged across all articles. Japanese, Korean, Arabic and Chinese understandably show lowest overlap values, since the scripts used in these languages do not allow for a direct inclusion of untranslated text into the English output. But for the European languages the situation is different: up to 40% of the input text is left untranslated. Note the difference in the overlap value between two Spanish translation systems: the translation server hosted by IBM Research and WebSphere Translation Server. In Section 5.6 we further explore the differences between these two translation systems in application to sentiment analysis.

The same entity may be referenced differently in different languages. To partially account for that, we remove language-specific stopwords such as “la” and “le” for French and “la”/“el” for Spanish, to produce the entity’s *canonical name*. We use these canonical names to match entities across languages in our experiments.

Language	$\mu_{untrans}$	$\sigma_{untrans}$	$\mu_{out/in}$	$\sigma_{out/in}$
Japanese	0.001	0.008	1.149	0.133
Korean	0.002	0.024	0.959	0.135
Arabic (research)	0.005	0.043	0.774	0.302
Chinese	0.008	0.070	1.459	0.197
Spanish (research)	0.091	0.082	0.989	0.083
German	0.099	0.119	0.964	0.137
Italian	0.167	0.153	0.992	0.051
French	0.316	0.228	0.950	0.108
Spanish (WTS)	0.399	0.252	0.966	0.148

Table 5.1: Mean and standard deviation of the overlap between original and translated text ($\mu_{untrans}$, $\sigma_{untrans}$) and of the ratio of translation system output to input size ($\mu_{out/in}$, $\sigma_{out/in}$).

5.4 News Stream Analysis

We computed daily entity sentiment scores over ten days from May 1 to May 10, 2007 for entities extracted from a subset of news text translated from Arabic, Chinese, French, German, Italian, Japanese, Korean and Spanish to English, as well as from a number of major U.S. newspapers. This specific time period was chosen because it has the most consistent spidered news volume over a contiguous period of time in our dataset. Only 19 entities proved common to all nine databases, out of which 14 were countries (France, America, China, Japan, Italy, Canada, Iran, Turkey, India, Australia, Sudan, Pakistan, Vietnam, Singapore), and four were cities (Washington DC, London, Moscow, Tokyo).

Table 5.2 shows the cardinality of intersection of entity sets extracted from each pair of languages. From this table we can observe that the Korean entities are mostly related to the Chinese and Japanese. Of the three Asian languages, the Japanese entities are the most connected to the European languages, which also form a strong cluster by themselves according to this distance measure.

	Arabic	Chinese	English	French	German	Italian	Japanese	Korean	Spanish
Arabic	**7601**	679	1403	1080	1053	552	193	195	1114
Chinese	679	**31783**	1124	941	1064	439	199	808	947
English	1403	1124	**24452**	2282	1989	735	221	281	2086
French	1080	941	2282	**10911**	1749	748	194	252	1818
German	1053	1064	1989	1749	**17882**	704	201	303	1638
Italian	552	439	735	748	704	**2662**	138	132	816
Japanese	193	199	221	194	201	138	**800**	98	196
Korean	195	808	281	252	303	132	98	**2870**	244
Spanish	1114	947	2086	1818	1638	816	196	244	**12843**

	Arabic	Chinese	English	French	German	Italian	Japanese	Korean	Spanish
Arabic	**100%**	9%	18%	14%	14%	21%	24%	7%	15%
Chinese	9%	**100%**	5%	9%	6%	16%	25%	28%	7%
English	18%	5%	**100%**	21%	11%	28%	28%	10%	16%
French	14%	9%	21%	**100%**	16%	28%	24%	9%	17%
German	14%	6%	11%	16%	**100%**	26%	25%	11%	13%
Italian	21%	16%	28%	28%	26%	**100%**	17%	5%	31%
Japanese	24%	25%	28%	24%	25%	17%	**100%**	12%	24%
Korean	7%	28%	10%	9%	11%	5%	12%	**100%**	9%
Spanish	15%	7%	16%	17%	13%	31%	24%	9%	**100%**

Table 5.2: Numbers of entities in intersections of each pair of languages (top) and percentage numbers that indicate the ratio of the intersection size to the smallest number of entities available for either of the two languages being intersected (bottom).

Frequency correlations

	Arabic	Chinese	English	French	German	Italian	Japanese	Korean	Spanish
Ar	**1.00** (2199)	**0.37** (141)	**0.36** (500)	**0.28** (397)	**0.33** (390)	**0.25** (190)	0.19 (78)	**0.73** (51)	**0.17** (210)
Ch		**1.00** (1051)	**0.24** (176)	0.08 (141)	**0.32** (147)	0.10 (94)	**0.74** (59)	0.18 (52)	0.04 (95)
En			**1.00** (12613)	**0.30** (1006)	**0.33** (763)	**0.36** (252)	**0.41** (83)	**0.27** (62)	**0.31** (301)
Fr				**1.00** (3769)	**0.38** (650)	**0.45** (249)	0.06 (74)	0.10 (57)	**0.21** (264)
Ge					**1.00** (4291)	**0.33** (242)	0.11 (74)	0.17 (58)	**0.14** (223)
It						**1.00** (768)	0.09 (56)	0.11 (34)	**0.27** (135)
Ja							**1.00** (241)	**0.40** (35)	0.25 (58)
Ko								**1.00** (416)	0.20 (38)
Sp									**1.00** (980)

Polarity correlations

	Arabic	Chinese	English	French	German	Italian	Japanese	Korean	Spanish
Ar	**1.00** (2199)	**0.56** (141)	**0.49** (500)	**0.45** (397)	**0.48** (390)	**0.57** (190)	**0.36** (78)	0.26 (51)	**0.61** (210)
Ch		**1.00** (1051)	**0.24** (176)	**0.51** (141)	**0.42** (147)	**0.41** (94)	0.08 (59)	**0.44** (52)	**0.49** (95)
En			**1.00** (12613)	**0.53** (1006)	**0.53** (763)	**0.58** (252)	**0.46** (83)	**0.35** (62)	**0.49** (301)
Fr				**1.00** (3769)	**0.53** (650)	**0.45** (249)	**0.51** (74)	**0.63** (57)	**0.40** (264)
Ge					**1.00** (4291)	**0.37** (242)	**0.26** (74)	**0.33** (58)	**0.26** (223)
It						**1.00** (768)	**0.58** (56)	**0.48** (34)	**0.35** (135)
Ja							**1.00** (241)	**0.35** (35)	**0.46** (58)
Ko								**1.00** (416)	**0.40** (38)
Sp									**1.00** (980)

Subjectivity correlations

	Arabic	Chinese	English	French	German	Italian	Japanese	Korean	Spanish
Ar	**1.00** (2199)	-0.05 (141)	0.03 (500)	**0.16** (397)	**0.12** (390)	**0.23** (190)	0.09 (78)	0.00 (51)	**0.39** (210)
Ch		**1.00** (1051)	**0.17** (176)	**0.22** (141)	**0.27** (147)	0.10 (94)	-0.03 (59)	0.20 (52)	-0.04 (95)
En			**1.00** (12613)	**0.13** (1006)	**0.23** (763)	**0.23** (252)	0.13 (83)	**0.27** (62)	0.07 (301)
Fr				**1.00** (3769)	**0.22** (650)	**0.18** (249)	0.21 (74)	-0.13 (57)	**0.16** (264)
Ge					**1.00** (4291)	**0.21** (242)	**0.28** (74)	**0.35** (58)	-0.00 (223)
It						**1.00** (768)	0.21 (56)	-0.02 (34)	**0.37** (135)
Ja							**1.00** (241)	**0.63** (35)	0.25 (58)
Ko								**1.00** (416)	0.08 (38)
Sp									**1.00** (980)

Table 5.3: Pearson correlation of frequency, polarity and subjectivity scores for entities extracted from the news corpus. All entities in the intersection are included in comparison. Counts are aggregated over all days for every entity. Bold correlations are significant with $p < 0.05$.

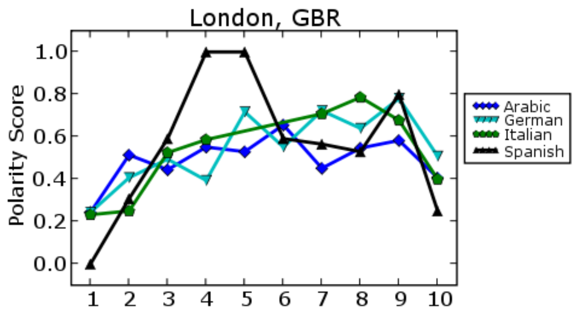


Figure 5.3: Polarity score of London in Arabic, German, Italian and Spanish over the May 1-10, 2007 period.

5.4.1 News Entity Frequency Correlations

The top part of Table 5.3 shows entity frequency correlation for every pair of languages. Every sample in this correlation is an aggregated frequency for a given entity in a given language over all ten days of the time period considered. The correlations significant with $p < 0.05$ according to a two-sided Student's t-test are highlighted with bold. We found no statistically significant entity frequency correlations when the frequency of each entity for each day was treated as a single sample. Note that daily correlation analysis is complicated by inconsistent notions of what a "day" is across different time zones and spidering patterns. Table 5.3 shows that English reaches a significant correlation with all other languages in the experiment, emphasizing its central role in our multi-language analysis approach. Figure 5.7 depicts these frequency correlation relations in a graphical form, making the clustering of European languages and Arabic versus Chinese, Japanese and Korean evident.

5.4.2 News Entity Polarity Correlations

Table 5.3 (middle) shows that entity polarity scores aggregated over the entire time period of experiments are significantly correlated for most pairs of languages—much more so than frequencies (top) or subjectivities (bottom) are. This allows us to conjecture the presence of a common underlying factor influencing entity sentiment in all languages—such as the "real" positivity or negativity of an entity.

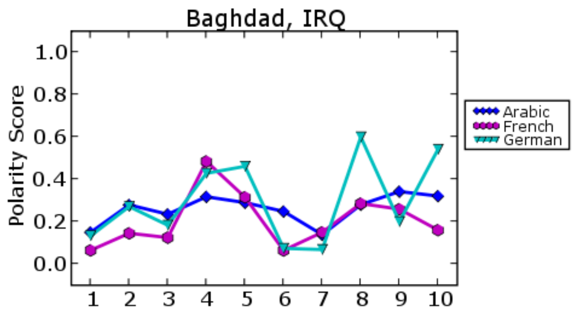


Figure 5.4: Polarity score of Baghdad in Arabic, French and German over the May 1-10, 2007 period.

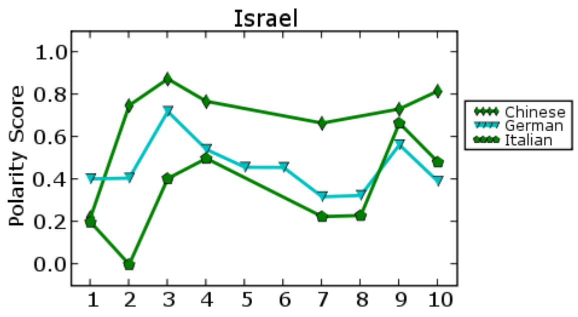


Figure 5.5: Polarity score of Israel in Chinese, German and Italian over the May 1-10, 2007 period.

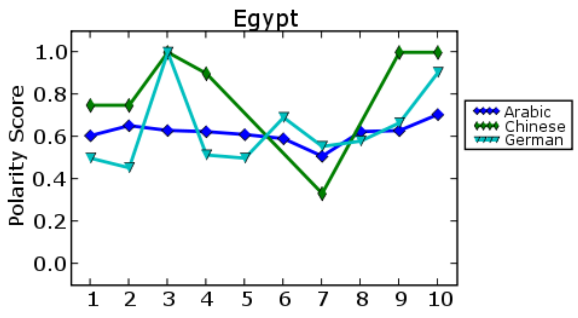


Figure 5.6: Polarity score of Egypt in Arabic, Chinese and German over the May 1-10, 2007 period.

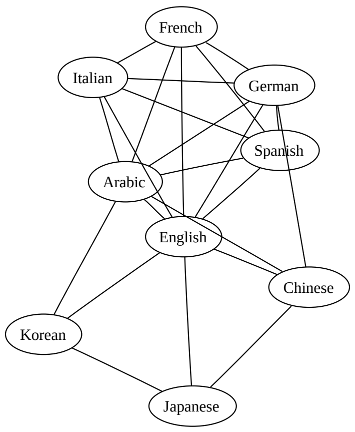


Figure 5.7: Graph of significantly correlated entity frequencies in different languages in the news corpus.

To look for the underlying reasons of the interdependencies between entity polarity scores in different languages, we analyzed the correlations between polarity scores of the same entity in different languages over our ten-day experiment time period. Figure 5.3 shows the sentiment score of London in four languages. An explanation of the consistent drop on May 10 could be the arrest of four people in the United Kingdom in connection with the July 7, 2005 London bombings [2]. In Figure 5.4 the polarity score drop starting May 6 is explained by the car bomb exploding in Baghdad on that day [2]. The spike on May 3 in the polarity score of Egypt in Figure 5.6 coincides with the launch of the International Compact for Iraq at Sharm El-Sheikh, Egypt [55]. The drop in the polarity score of Israel on May 3-6 can be attributed to the protests against Prime Minister Ehud Olmert and his government over their handling of the 2006 Lebanon War [2]. These examples indicate that in cases of significant correlation between sentiment scores in different languages there are often real-world explanations of changes in these scores.

5.5 Parallel Corpus Analysis

We also analyzed entity sentiment scores in the European Commission Joint Research Centre's Acquis multilingual parallel corpus [80]. This corpus contains the total body of European Union (EU) law applicable in the EU Member States. The JRC-Acquis corpus does not contain timestamp information for documents, making temporal analysis impossible. However, we can still analyze correlations of entity frequencies and sentiment scores between different languages. We performed our experiments with five languages out of the 22 in which the JRC-Acquis corpus is available: English, French, German, Italian and Spanish. The documents in languages other than English were first translated to English using IBM WebSphere Translation Server, and the resulting translated documents were processed through our Lydia pipeline, giving a subjectivity and polarity score for each entity as a result.

Table 5.4 shows entity frequency, polarity score and subjectivity score correlations in the JRC-Acquis corpus for pairs of languages, analogous to Table 5.3 for the news corpus. We observe greater frequency and subjectivity correlation between languages in the JRC-Acquis corpus than in the news corpus. This is consistent with expectations because unlike the news corpus, the same text is used in all languages in the JRC corpus. Even though one should not expect strong sentiment expression in law documents, polarity scores also show substantial consistency.

5.6 Cross-Translation System Analysis

Since two different translation systems were available to us for the Spanish language, it was natural to compare sentiment scores of entities in the output of these

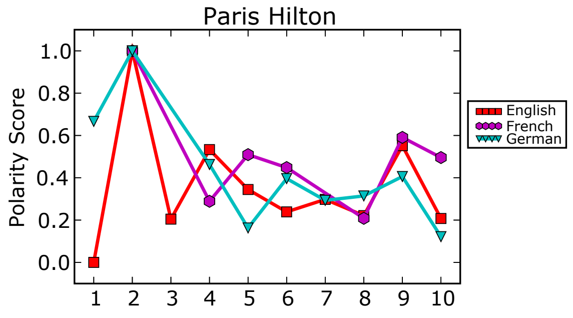


Figure 5.8: Polarity score of Paris Hilton, May 1-10, 2007.

Frequency correlations

	English	French	German	Italian	Spanish
English	**1.00** (619)	**0.21** (144)	**0.20** (186)	**0.64** (196)	**0.59** (181)
French		**1.00** (342)	0.06 (135)	**0.67** (153)	**0.78** (152)
German			**1.00** (1460)	0.13 (172)	**0.17** (166)
Italian				**1.00** (484)	**0.83** (192)
Spanish					**1.00** (527)

Polarity correlations

	English	French	German	Italian	Spanish
English	**1.00** (619)	**0.21** (144)	0.09 (186)	**0.45** (196)	**0.25** (181)
French		**1.00** (342)	0.09 (135)	**0.42** (153)	**0.30** (152)
German			**1.00** (1460)	**0.20** (172)	0.11 (166)
Italian				**1.00** (484)	**0.43** (192)
Spanish					**1.00** (527)

Subjectivity correlations

	English	French	German	Italian	Spanish
English	**1.00** (619)	**0.24** (144)	**0.62** (186)	**0.43** (196)	**0.28** (181)
French		**1.00** (342)	**0.40** (135)	**0.64** (153)	**0.52** (152)
German			**1.00** (1460)	**0.66** (172)	**0.75** (166)
Italian				**1.00** (484)	**0.60** (192)
Spanish					**1.00** (527)

Table 5.4: Pearson correlations of frequency, polarity and subjectivity for entities extracted from the JRC-Acquis corpus. All entities in the intersection are included in comparison. Bold correlations are significant with $p < 0.05$.

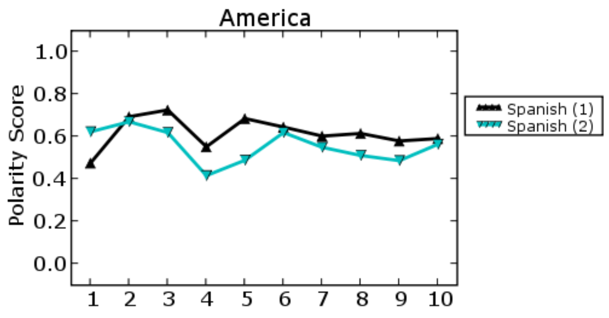


Figure 5.9: Polarity scores of America in the output of (1) IBM WebSphere Translation Server (Spanish); (2) a newer translation system hosted by IBM Research.

two translation systems. We found that when we aggregated sentiment scores over the entire ten-day period for every entity, the resulting correlations of entity polarity, subjectivity and frequency were 0.52, 0.46 and 0.47 respectively, all with $p < 0.001$ significance. When entity scores on individual days were treated separately, however, these correlations went down to 0.19 for polarity, 0.45 for subjectivity and 0.42 for frequency. This indicates that there is a high variance in the amount of positive and negative references but little difference in the overall volume of subjective references between the outputs of the two translation systems.

Looking at polarity as a function of time in the output of the two translation systems, we see that the two scores can be fairly consistent (Figure 5.9). Still, the ten-day aggregated scores were more concordant.

5.7 Cross-Cultural Observations

To explore the suitability of our scores for cross-cultural comparisons, we calculated polarity scores of all countries appearing in at least 7 out of our 9 language-specific databases, in every language. To quantify how comparable entity scores are between languages, we calculated the variance of each entity's polarity score across languages. With polarity scores calculated as in (5.1), the variance was at most 0.068 and the sum of variances across all languages was 0.525.

One source of the differences in polarity scores between languages follows from different probabilities of positive and negative sentiment word appearance in the

	Arabic	Chinese	English	French	German	Italian	Japanese	Korean	Spanish	Mean	StdDev
Cameroon	0.295	0.528	0.219	**0.155** (7)	0.161	0.158	N/A	N/A	*0.566*	0.297	0.178
Lebanon	***0.404*** (1)	0.327	0.311	0.208	0.251	N/A	0.375	N/A	0.258	0.305	0.070
Pakistan	0.393	0.254	**0.456** (2)	0.321	0.313	0.326	*0.583*	0.311	0.286	0.360	0.103
Philippines	*0.462*	0.388	**0.378** (6)	0.428	0.440	N/A	0.191	0.443	N/A	0.390	0.093
Iraq	**0.346** (7)	*0.535*	0.372	0.275	0.354	N/A	0.414	0.498	0.396	0.399	0.084
Cuba	0.422	*0.692*	0.299	0.414	0.447	0.545	0.125	N/A	**0.402** (6)	0.418	0.166
USA	0.404	*0.561*	**0.545** (2)	0.456	0.436	0.520	N/A	N/A	0.305	0.461	0.090
Sudan	**0.500** (4)	0.509	0.444	0.438	0.437	*0.659*	0.358	0.574	N/A	0.490	0.093
Venezuela	0.241	*1.000*	0.468	0.350	0.569	0.155	0.732	N/A	**0.477** (4)	0.499	0.272
Mexico	0.561	*0.859*	0.385	0.423	0.387	N/A	N/A	0.469	**0.533** (3)	0.517	0.166
Canada	0.531	0.522	**0.498** (6)	0.508	*0.705*	0.420	0.450	0.478	0.573	0.521	0.083
China	0.556	**0.420** (9)	0.433	0.473	0.470	0.622	0.612	0.540	*0.663*	0.532	0.088
Germany	0.483	0.421	0.480	0.598	**0.639** (2)	*0.680*	0.561	N/A	0.434	0.537	0.097
Egypt	**0.519** (5)	0.823	0.540	0.361	0.576	0.419	0.355	*0.846*	0.463	0.545	0.181
Australia	0.493	0.528	**0.541** (3)	0.560	0.508	*0.738*	0.506	0.533	0.519	0.547	0.074
America	0.405	0.651	**0.568** (4)	0.502	0.566	0.605	*0.666*	0.480	0.550	0.555	0.083
India	0.571	0.626	**0.487** (8)	0.547	0.396	0.499	*0.719*	0.623	0.555	0.558	0.093
Chile	0.576	0.405	0.586	0.559	0.563	*0.750*	N/A	N/A	**0.502** (6)	0.563	0.104
Argentina	0.461	0.430	0.472	0.654	0.624	*0.738*	N/A	N/A	**0.562** (4)	0.563	0.115
Spain	0.583	0.629	0.466	0.468	0.533	*0.720*	N/A	N/A	**0.613** (3)	0.573	0.092
Japan	*0.689*	0.531	0.542	0.602	0.397	0.668	**0.589** (5)	0.534	0.648	0.578	0.090
Italy	0.554	0.605	0.465	0.557	0.536	**0.615** (2)	*1.000*	0.454	0.417	0.578	0.172
Austria	0.515	N/A	0.489	0.507	**0.568** (4)	0.575	0.672	N/A	*0.851*	0.597	0.128
Saudi Arabia	**0.611** (3)	N/A	0.458	0.564	0.446	0.556	*0.891*	N/A	0.669	0.599	0.151
France	0.561	*0.688*	0.611	**0.566** (8)	0.570	0.673	0.611	0.569	0.602	0.606	0.047
Brazil	0.557	0.848	0.494	0.516	N/A	0.518	*0.911*	N/A	**0.529** (4)	0.625	0.176
Switzerland	0.628	0.455	0.527	0.697	**0.607** (5)	0.559	*1.000*	N/A	0.676	0.644	0.164
Jordan	**0.678** (4)	*0.931*	0.569	0.414	0.739	0.592	0.432	N/A	0.843	0.650	0.185
Belgium	0.652	0.754	0.643	**0.621** (6)	0.583	0.659	N/A	N/A	*0.754*	0.666	0.065

Table 5.5: Normalized country polarity scores in all languages. Countries are sorted by their mean score across all languages. Polarity scores are normalized to bring mean polarity to 0 and variance to 1 across all country entities in each language. The language spoken in the country is highlighted with bold. For every country the rank of its polarity in its own language in the row (1=highest, 9=lowest) is given in parentheses. Maximum polarity for each country is italicized.

	Arabic	Chinese	English	French	German	Italian	Japanese	Korean	Spanish
pos_per_ref	0.987	0.039	0.894	0.669	0.440	0.438	0.629	1.333	0.509
neg_per_ref	0.622	0.025	0.830	0.438	0.350	0.458	0.598	0.717	0.448
pos_coef	0.906	22.719	1.000	1.337	2.033	2.042	1.422	0.671	1.758
neg_coef	1.334	33.443	1.000	1.893	2.369	1.813	1.389	1.157	1.853
neg_coef/pos_coef	1.473	1.472	1.000	1.416	1.165	0.888	0.977	1.726	1.054

Table 5.6: Normalization coefficients for all languages.

same sentence with an entity. To account for this bias, we calculated the average numbers of positive and negative sentiment words per entity occurrence:

$$pos\_per\_ref = \frac{\sum_{i=1}^{N_{entities}} pos\_sentiment\_refs_i}{\sum_{i=1}^{N_{entities}} total\_occurrences_i}$$

$$neg\_per\_ref = \frac{\sum_{i=1}^{N_{entities}} neg\_sentiment\_refs_i}{\sum_{i=1}^{N_{entities}} total\_occurrences_i}$$

Table 5.6 gives values of these statistics for all languages. The neg_coef/pos_coef line shows that Korean is the most biased language towards positive sentiment, and Italian is the most biased towards negative, although not much more than English is. We discount each positive or negative sentiment word occurrence in ith language versus English:

$$pos\_coef_i = \frac{pos\_per\_ref_{English}}{pos\_per\_ref_i}$$

$$neg\_coef_i = \frac{neg\_per\_ref_{English}}{neg\_per\_ref_i}$$

We then calculate the normalized polarity as

$$\frac{pos\_sentiment\_refs_i}{pos\_sentiment\_refs_i + \frac{neg\_coef}{pos\_coef} \times neg\_sentiment\_refs_i}$$

This technique reduces the sum of cross-language polarity score variances for countries by 6% to 0.494. The normalized polarity scores are given in Table 5.5.

From the standard deviation column of Table 5.5 we can see that the lowest polarity variance corresponds to developed countries (France, Belgium, Australia, Canada, USA) or countries with recent conflicts (Lebanon, Iraq), and the highest variance corresponds to developing countries such as Jordan, Egypt, Venezuela and Brazil.

We also hypothesized that for every country its own language would rank it among the highest. To test this, we included the rank (1=highest, 9=lowest) of country's polarity in its own language among all languages in Table 5.5. There is little evidence in favor of this hypothesis, perhaps because ten days is too short a time period to capture a long-time country sentiment in the news.

5.8 Conclusions

Using our Lydia text analysis system, we analyzed entity sentiment in newspapers in nine languages, and in five languages of a parallel corpus. Our experiments showed that our method of calculating entity sentiment scores is consistent with respect to varying languages and news sources. We also compared scores across two different translation systems for Spanish and concluded that the success of our methods is largely translation system independent. Finally, we proposed a sentiment score normalization technique for cross-language polarity comparison, allowing for meaningful cross-cultural comparisons.

Chapter 6

Concordance-Based Entity-Oriented Search[1]

6.1 Introduction

We consider the challenge of building a search engine that retrieves appropriate entities (e.g. people, places, things) in response to user queries over a web-scale text corpus containing these entities. This contrasts with a typical search engine which returns documents or webpages in response to user queries. Concrete applications of entity search are detailed below, but we begin with some motivational examples of our search engine in action. Table 6.1 shows the top five results returned by our system for six queries. Three of these queries are entities themselves ("NEW YORK YANKEES", "GOOGLE" and "TENNIS") and the top result returned for them is the query itself. The other results returned for entity queries are connected with the queried entity in various ways: famous tournaments and players for tennis; its CEO name, headquarters location and rival search engine companies for Google; and the names of the manager, players and a rival team for the New York Yankees. The other three queries ("POLITICAL CORRUPTION", "POLARIZING FIGURE", and "BRITISH PRIME MINISTER") are not entities but concepts indirectly referring to entities. The results for the "POLITICAL CORRUPTION" query are

[1] This chapter is drawn from our paper "Concordance-Based Entity-Oriented Search" [18], an expanded version of the conference paper [17].

	TENNIS	NEW YORK YANKEES	GOOGLE
1	Tennis	New York Yankees	Google
2	Roger Federer	Joe Torre	Yahoo
3	Andre Agassi	Alex Rodriguez	Eric Schmidt
4	U.S. Open	Derek Jeter	Mountain View
5	Andy Roddick	Boston Red Sox	Microsoft

	POLITICAL CORRUPTION	POLARIZING FIGURE	BRITISH PRIME MINISTER
1	Jack Abramoff	Hillary Rodham Clinton	Tony Blair
2	George Ryan	Katherine Harris	Winston Churchill
3	Tom DeLay	David Geffen	Margaret Thatcher
4	Pete Domenici	Donald H. Rumsfeld	British
5	King Gyanendra	Dick Cheney	Gordon Brown

Table 6.1: Results for certain example entity queries.

three politicians investigated for corruption, one involved in a violation of ethic rules, and the controversial King of Nepal. The "POLARIZING FIGURE" returns the names of four politicians inspiring strong but differing opinions. The "BRITISH PRIME MINISTER" query returns three former and the current Prime Minister of the United Kingdom. We encourage the reader to experiment with our search engine at `http://www.textmap.com`.

Compelling applications of entity-targeted search over unstructured text include:

- *Navigational Search.* Augmenting document results from a conventional Web search engine with entity results is particularly relevant for *navigational* queries [27]. The related entity results may help satisfy the user's information need, as per question answering systems. Otherwise they can provide meaningful navigational alternatives to user's document-oriented query.

  Indeed, we present experimental results demonstrating that our methods can predict roughly 5-10% of a user's subsequent entity queries. This is a large enough fraction of user queries to justify displaying navigational shortcuts to speed search.

- *Encyclopedia Search.* When the text corpus is (or is comparable) to a collection of encyclopedia entries, the performance of article retrieval can be improved by taking into account all mentions of an entity across articles, rather than just in the entry corresponding to that entity.

  We are confident that our techniques could be used to improve the performance of the Wikipedia search engine. For example, a search on Wikipedia for "Microsoft chairman" returns as the top result a stub article for Helmut Panke, who is a member of the Board of Directors of Microsoft and a former chairman of BMW AG. It lists Bill Gates as the 26th most relevant article for this query. Our system, in contrast, correctly returns Bill Gates at the top position. Analysis of entity references within curated text sources (e.g. encyclopedias and news sources) can also be applied to general web search.

- *Product Search.* Applied to a corpus of product reviews, our approach should be able to give relevant product suggestions. Aggregating all mentions of specific products in reviews, blogs and webpages results in a higher recall than existing product search engines, many of which currently just search product names, or limited collections of product descriptions. In the terminology of [27], this can improve handling of *transactional* web queries.

The contributions of our work are as follows:

- *Analysis of entity occurrences in Web query logs.* The best way to understand how search queries should be pre-processed and answered is to analyze past query data [40]. By analyzing the 36 million queries of the AOL dataset [13], we found that 20-40% of web search queries consist solely of single entities, while 70-87.5% queries contain entities as part of the query. These findings demonstrate that a very high percentage of all web searches recognizably target entities or have entities associated with them.

- *A first-in-literature implementation of an entity search engine.* We approach the entity retrieval problem by utilizing all occurrences of each specific entity throughout a text corpus. For every entity, we automatically compose a *concordance*—a document capturing the context of all the occurrences of the entity in the corpus. Then we index and search these documents using an open source information retrieval package (Lucene) with a scoring scheme customized to reflect the specificity of automatically generated documents. Although our prototype search engine has been developed over a modestly-sized 18 GB corpus of news, we see no fundamental difficulties in scaling this to web-scale search.

- *Empirical evaluation of our search engine.* Identifying a gold standard to evaluate the performance of our "first-in-literature" entity search engine is a non-trivial task. We do so by comparing the top entities returned when the query is itself a single entity to the entities having the top statistical *juxtaposition score* [62] with the queried entity. Juxtaposition score is a measure of how much more frequently two entities co-occur than they would by chance. Our search engine provides a much greater flexibility in entity retrieval by allowing free-text queries than is possible using juxtaposition scores.

- *Time-dependent entity/document search.* The most relevant entities associated with a query evolve during time. *Jennifer Aniston* is currently less relevant to *Brad Pitt* than *Angelina Jolie*, even though her total number of co-locations at the time of our experiment exceeded those of her rival. To account for new articles added every day to the corpus, and to weigh recent hits higher, we support generation of multiple separate concordances for the same entity over disjoint time periods (e.g. months) and aggregate the hits for all such periods into a single result. Through analysis of the AOL query dataset, we determine the optimal discounting of entity references over time to identify the most relevant entities at time of search.

The rest of this document is organized as follows. Section 6.2 reviews previous work on entity-aware search. Section 6.3 provides motivation for the use of entities in search engines by analyzing occurrences of news and Wikipedia based entities in a web query log. Section 6.4 describes the design of our news entity search system. Section 6.5 provides an empirical evaluation of the retrieval performance of our system. Section 6.6 concludes the report and outlines the directions of our future work.

6.2 Related Work

Our entity-search engine is built on top of our Lydia news analysis system [47, 56, 62, 63, 66]. The Lydia system automatically builds an entity database from online U.S. newspapers. Customized website scrapers download articles on a daily basis, convert them to a standard format and store them in an archive. Then the articles are run through a pipeline that performs part-of-speech tagging, named entity identification and categorization, geographic normalization, intradocument coreference resolution, and cross-document entity coreference resolution via clustering. The techniques used for entity identification are part-of-speech tagging, hand-crafted rules based on part-of-speech tag, capitalization and punctuation, and gazetteers.

The above-mentioned papers discuss design and implementation of the Lydia entity recognition system. In this work we use the Lydia system output comprising text with marked-up entities and the entity database to provide search targeting the extracted entities.

The current research relevant to entity-aware search can be subdivided into three main directions:

- Augmenting traditional document retrieval systems with the knowledge of entities [28, 35].
- Semantic Web research aiming to create a Web of interrelated entities and thus greatly simplify entity search and improve document search [21, 87];
- Extraction of relations between entities from unstructured text or web pages and searching these relations [14, 33, 34, 73].

6.2.1 Semantic-Aware Document Retrieval

Chu-Carroll, et al. [35] describe how the XML fragment query language can be applied to semantic search. The availability of a technology to augment unstructured text with additional markup including named entities, their categories and relations between them is assumed. The following operations are allowed to be used in queries: conceptualization (specification of a category that should appear in a block), restriction (specification of tag(s) inside which a term must appear), and entity relation to look for. Question answering is handled by converting questions into XML fragment queries. This paper outlines a high-precision approach to semantic-aware information retrieval for applications where loss in recall is less important, such as intelligence investigations.

Carpenter [28] describes an attempt to use named entity recognition to improve search results. They compare a baseline Lucene *tf.idf*-based approach and the one that uses LingPipe named entity recognition to match phrases in the query. Their named entity recognizer was based on a Bayesian generative model tagging words as beginning, continuation or not being in a named entity, with a small history window. They report term extraction to be on the state-of-the-art level. The system was evaluated in the Ad Hoc task of the Genomics track of TREC 2004 on a ten-year subset of MEDLINE citations, but the precision and recall of the phrase-based system were always lower than the baseline. Possible explanations are the shortness of the MEDLINE citations that prevents phrase fragments from appearing as part of a different phrase, and phrase queries boosting scores of wrong documents.

6.2.2 Semantic Web

In [21] Berners-Lee, Hendler and Lassila first introduce the concept of the Semantic Web. Three major components of the Semantic Web are XML (eXtensible Markup Language), RDF (Resource Definition Framework), and ontologies. RDF encodes dependencies between resources in the form of “subject, object, verb”-like triples, where each of elements is a Universal Resource Identifier (URI).

The most typical kind of ontology is said to have a taxonomy and a set of inference rules. A taxonomy supplies a hierarchy of classes, subclasses and properties that subclasses can inherit. Inference rules describe how to derive new associations from existing associations between entities. Semantic Web *agents* are programs that collect Semantic Web content from diverse sources, process the information and exchange the results with other programs. Even agents that were not expressly designed to work together should be able to exchange data augmented with semantics. The idea of agents exchanging “proofs” of their results for verification of their inference and security, and the idea of directories of services provided by agents are also described. Applications to embedded systems and home automation are mentioned.

Shadbolt, Berners-Lee and Hall follow up on the topic of [21] in [87]. This paper on the Semantic Web discussing the expectations of [21] and the current situation around the Semantic Web. Technologies that already exist are RDF and OWL (Web Ontology Language) and they have reached the status of W3C recommendations but need wider adoption. Work has begun on an inference rule interchange format (RIF, `http://www.w3.org/2005/rules/`). The authors state that these technologies need uptake by a small, but dedicated group to be consequently widely adopted, as happened to the Web with the community of physicists. Regarding ontologies, the authors do not view them as static structures, but rather as dynamic and developing representations of knowledge. They compare ontologies to "folksonomies"—collaborative tagging schemes found today on social networking websites such as Flickr. The technical challenges arising with the Semantic Web discussed include effectively querying huge numbers of decentralized repositories and building a Semantic Web browser visualizing the huge connected RDF graph. The authors conclude by pointing out that new local-scale architectures such as Web protocols or peer-to-peer networks can lead to global-scale societal and technical changes.

6.2.3 Relation Extraction

Banko, Cafarella, Soderland, Broadhead, and Etzioni in [14] present a concept of "Open Information Extraction", meaning the extraction of relational tuples from unstructured text without prior specification of relation types. They introduce a fully implemented scalable domain-independent OIE system TextRunner that extracts relation tuples from Web text. TextRunner does not use "heavy" linguistic technologies such as parsers or named entity recognizers, due to performance demand of processing high volumes of Web text. Instead, a parser is used to train a Naive Bayes classifier which is then used on a full volume of Web text to accept or reject relations. After extracting relations, TextRunner assesses the probability of each assertion. Finally, an inverted index is built on the extracted relations, allowing for searching both arguments and predicate fields. The authors report a 33%

error reduction compared to their previous closed information extraction system KnowItAll.

Paşca, Lin, Bigham, Lifchits and Jain [73] describe a new iterative approach for large-scale extraction of facts from the Web. Their system is capable of extracting one million facts from 100 million Web documents, starting from ten seed facts, and using no additional knowledge or lexicons. At every iteration, patterns are extracted from the current set of seed facts using word similarities calculated previously based on vector-space context similarities. These patterns are scored, ranked and used to extract candidate facts that are also scored and selectively added to the seed set for the next iteration. The system is evaluated using "Person-BornIn-Year" relations. Precision of around 90% and recall of 70-90% are reported.

Cheng and Chang in [33, 34] present *MetaQuerier*, a system for finding and querying data sources on the "deep web". They consider a range of entity domains such as phone numbers, book cover images, PDF or PPT files, names, dates, email addresses. Motivating examples they provide include finding the phone number of Amazon.com customer service and the list of professors working in the database area in all Computer Science departments. They attempt to provide a way to answer queries such as above directly instead of having the user formulate the query in terms of web page containing the information of interest. The data inputs consist of crawled web pages and "deep web" pages obtained by querying specific database-driven websites, e.g. real estate and book databases. Entity extraction happens using domain-specific entity models describing how to identify instances of an entity class in webpages. The query engine implemented using the Lemur toolkit [71] performs pattern matching and tuple scoring for online query processing. The system is evaluated on the domain of emails of professors across universities with resulting precision of 85% and recall of 90%.

6.2.4 Entity Search using Existing Search Engines

It should be noted that some conventional web search results might be considered entity results. For example, Wikipedia pages may be thought of as the de facto entity pages, and frequently come up as the top search result of a Google search for the corresponding entity. Another example could be Google's question answering system that precisely answers a certain class of factual queries in the first line of search results. In other applications, such as product search, it is logical to compose a document for every entity (product), perhaps consisting of product specification and all customer reviews, and use a document search engine to index and search these documents. We also use a similar approach, but we propose a universal scheme for building documents corresponding to every entity.

6.3 Entities in Web Queries

To gain insight into the performance of search engines, it is essential to analyze past query logs [40]. For example, to motivate entity-oriented search we should examine the frequency with which web queries target recognizable entities.

Search queries contain highly proprietary information, and therefore search engine companies do not often make it available to researchers. Fortunately a comprehensive web query dataset became available to us in August 2006 when AOL unintentionally released 36 million search queries by 500,000 users collected from March to May 2006. Although this release represented a serious violation of user privacy [13], the dataset has been widely used in the research literature [4, 15, 76]. This dataset was very useful to us for collecting cumulative statistics on web queries. When analyzing it, we did not use the user ID field or manually examine individual low-frequency queries, thus maintaining and respecting user privacy.

6.3.1 Approach to Analyzing Web Queries

We chose to identify entities by matching search queries to existing lists of known entities. Another possible approach might try to recognize named entities in query text using a statistical named entity recognizer such as LingPipe [1], but this would be much less accurate due to the lack of capitalization and contextual information in web query data. For comparative purposes, we also used approximately three million entities identified from entry titles in Wikipedia.

We were interested both in *perfect matches*, where the entire query is an entity from our database, and *partial matches*, where an entity is contained in the query as

Query	Frequency
my space	28516
map quest	27842
ask jeeves	18988
my space.com	14148
craigs list	5717
www google.com	5050
bed bath and beyond	4228
disney channel	4154
www yahoo.com	3637
e bay	3540

Table 6.2: Most frequent partial entity-query matches that are indeed complete matches.

a contiguous range of tokens (a "sub-query"). Many of the partial matches became perfect matches if relaxed criteria for matching entities with queries were used, because of typos and word separation alternatives (see Table 6.2). Therefore, we considered the following levels of strictness of matching queries with entities:

- *Exact comparison.* The exact case-insensitive appearance of entity in the query is required.
- *Alias resolving comparison.* For every entity in the database and every query we construct a list of aliases. By alias we mean the original string with different tokenization, punctuation, prefixes and/or suffixes:
  - URL normalization. The "http://", "www." prefixes and ".com", ".net" etc. suffixes are removed. Spaces are replaced with dots. E.g. the alias list for "www yahoo com" is "www.yahoo.com", "wwwyahoocom", "yahoo.com", "www yahoo com", "yahoo".
  - "&" is replaced with "and";
  - "Inc.", "Co.", "Corp." suffixes are removed;
  - plural nouns are reduced to singular, etc.

If at least one alias of an entity matches with at least one alias of a query, the query is considered an entity query. The same criteria is used for a "sub-query" (a contiguous range of tokens in the query) to locate partial matches.

- *Alias resolving with phonetic hashing.* To further explore possible entity appearances in search queries and to deal with misspellings, we add a Double Metaphone [78] hash of every entity, query and sub-query to its respective alias list.

All Queries

Matches	No aliases	Aliases	Metaphone
perfect	17.91%	26.50%	38.82%
partial	55.14%	53.59%	48.41%
total	73.05%	80.09%	87.23%

Unique Queries

perfect	2.07%	5.33%	18.57%
partial	68.85%	69.72%	65.23%
total	70.92%	75.05%	83.80%

Table 6.3: Match frequencies for all 36,389,577 queries and 10,154,743 unique queries (after duplicate removal) compared against Lydia entity list. Each entry represents the percentage of queries (including duplicates in the "all queries" part) that are perfect or partial matches with an entity name.

6.3.2 Frequencies of News Entities in Queries

Table 6.3 presents the percentage of perfect and partial matches of AOL queries to Lydia entities under three levels of matching strictness discussed above. As the "no aliases" column in the "all queries" part of Table 6.3 indicates, almost 18% of queries exactly match one of the entities in our database. Interestingly, these queries constitute only 2% of all unique queries. On average, every query that is a perfect entity match, is repeated $\frac{17.91\% \times 36{,}389{,}577}{2.07\% \times 10{,}154{,}743} \approx 31$ times, whereas an arbitrary query is repeated only $\frac{36{,}389{,}577}{10{,}154{,}743} \approx 3.6$ times. Entities extracted from the news are inherently popular and likely to be searched for.

Moving to the "aliases" column in the "all queries" part of Table 6.3, we see a $\frac{26.5\%-17.9\%}{17.9\%} \approx 48\%$ gain in the number of perfect matches, indicating that there are many mistyped variations of how entity names are formatted in search queries. This shows that a set of hand-crafted rules could significantly improve entity recognition rate by a search engine. The addition of metaphone hashing of queries increases the number of perfect entity matches by $\frac{38.8\%-26.5\%}{26.5\%} \approx 46.5\%$. Clearly, spelling correction is an important problem in both entity-oriented and document-oriented search engine design.

When we look at how the number of partial matches changes between columns in Table 6.3, we see two trends. First, partial matches decrease from left to right as they become perfect after alias expansion. Second, we see an increase from 68.85% to 69.72% in "unique queries" because more non-matches became partial matches than partial matches became perfect matches.

To summarize, from Table 6.3 we see that 73%-87% queries contain part that is recognizable as an entity, and 18%-39% queries are completely recognizable as entity names.

All Queries

Matches	No aliases	Aliases	Metaphone
perfect	19.86%	29.90%	41.75%
partial	49.85%	50.21%	45.75%
total	69.71%	80.11%	87.50%

Unique Queries

perfect	2.69%	6.05%	19.47%
partial	65.95%	68.42%	64.25%
total	68.64%	74.47%	83.72%

Table 6.4: Match frequencies for queries compared against Wikipedia entity list.

Perfect Matches

No aliases		Aliases		Metaphone	
unknown	3741787	unknown	3538105	unknown	4347665
person	842244	website	1851898	title	4192761
website	641492	title	1672178	website	1949374
organization	268081	person	944648	person	1798689
name	237937	company	301037	company	274560
company	114401	name	193818	place	188644
disease	68631	organization	169132	name	169435
place	66809	place	102282	organization	151581
last name	59450	TV series	98717	TV series	116031
university	46352	disease	71247	movie	87783

Partial Matches

No aliases		Aliases		Metaphone	
unknown	13139962	unknown	8849382	unknown	6579174
person	2644690	title	4941389	title	5615035
organization	743740	person	1832229	person	2795352
name	652718	website	667839	place	291081
last name	515811	name	366955	website	278234
place	312335	organization	345457	city	254026
other	300025	last name	307232	company	173091
location	228221	place	274457	name	161915
first name	209389	company	213399	movie	139118
county	108325	city	210430	organization	137198

Table 6.5: Matches of news entities with queries by category.

6.3.2.1 Frequency of Entities in Queries by Category

The Lydia system's entity database defines category information (place, person, city, country etc.) for each entity. We use the taxonomy of categories described in [86]. This category information has been obtained using a naïve Bayes classifier trained on a 2-to-3 word context of entity occurrences in news, so it is not always accurate. However, counting frequencies of search query categories provides some useful insight.

Table 6.5 shows category distribution corresponding to cells of Table 6.3. Only top ten categories are shown for each experiment. Unknown represents the 30-60% of entities the naïve Bayes classifier was unable to assign categories to. Moving from the "no aliases" to "aliases" column in the top part of Table 6.5 much more websites and titles get recognized, proving the usefulness of our URL matching method. When in the same part of Table 6.5 we go from the "aliases" to "metaphone" column, the number of title matches increases much more significantly than the number of website matches does, indicating that misspellings are more likely to occur in titles than in website URLs.

The *website* category has 600,000–2,000,000 completely matching queries and below 300,000 partial matches. The change from 667,839 to 278,234 partial matches for websites going from "aliases" to "metaphone" can be explained by many of these partial matches becoming perfect matches when using metaphone. This shows that when the user is looking for a particular website using parts of its domain name, he or she is unlikely to include additional information in the query, which greatly simplifies the handling of website queries.

6.3.3 Frequency of Wikipedia Entities in Queries

Table 6.4 shows the same experiments as Table 6.3, but done using entity lists obtained from Wikipedia, under the assumption that each Wikipedia page represents a named entity. At the time of this writing, there are about three million pages on Wikipedia. There is surprisingly little difference between the results obtained using the Lydia and the Wikipedia lists. The biggest difference between Table 6.3 and Table 6.4 (5.29%) is between the number of incomplete matches with no alias

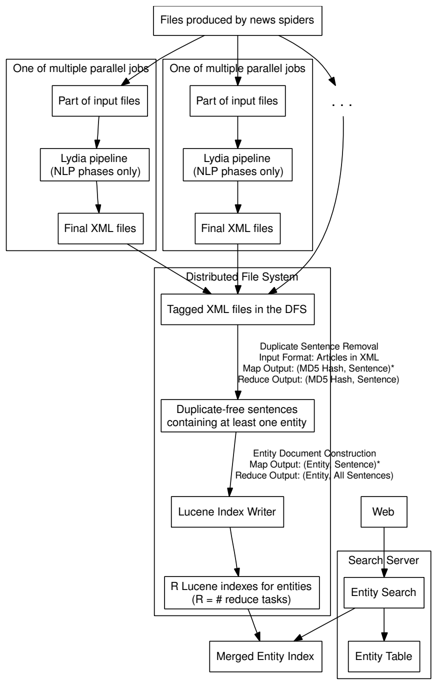

Files produced by news spiders
One of multiple parallel jobs
Part of input files
Lydia pipeline (NLP phases only)
Final XML files
One of multiple parallel jobs
Part of input files
Lydia pipeline (NLP phases only)
Final XML files
. . .
Distributed File System
Tagged XML files in the DFS
Duplicate Sentence Removal
Input Format: Articles in XML
Map Output: (MD5 Hash, Sentence)*
Reduce Output: (MD5 Hash, Sentence)
Duplicate-free sentences containing at least one entity
Entity Document Construction
Map Output: (Entity, Sentence)*
Reduce Output: (Entity, All Sentences)
Lucene Index Writer
R Lucene indexes for entities (R = # reduce tasks)
Web
Search Server
Entity Search
Entity Table
Merged Entity Index

Figure 6.1: High-level design of our Lydia entity-oriented search system.

expansion, counting duplicate queries. No other differences exceed 3.4%. This indicates that entity analysis in search queries can in fact be done using freely available sources such as Wikipedia, even when a complex text mining system such as Lydia is not available. The prospects of a more reliable version of Wikipedia [12] make this approach even more attractive. This also confirms the viability of Lydia's named entity recognition scheme.

6.4 Concordance-Based Entity Search

For the design of our entity search engine, we cannot assume availability of a preexisting text document corresponding to every entity, because such an assumption would restrict our technique to manually prepared entity collections such as that of products in e-commerce websites or encyclopedia articles. Instead, we perform retrieval of entities based on all occurrences of each particular entity throughout a text corpus. For every entity we generate a “concordance”—a text document containing all unique sentences from the text corpus in which that entity occurs. Then we search these documents with an open-source search engine (Lucene). This approach allows us to leverage the existing development in document-oriented information retrieval.

Figure 6.1 shows the high-level architecture of our system. During the indexing phase, an entity index is constructed. This index is used by the search server to handle online queries. The indexing procedure is described in detail in the next section.

6.4.1 Indexing

During the indexing phase, an entity index is constructed, which is later used by the search server to handle online queries. The indexing procedure starts with processing news articles with the Lydia pipeline. The next step separates input documents into sentences so that concordances can be built. During that step we also eliminate duplicate sentences to deal with duplicate and near-duplicate articles in the input. After that, the system constructs concordances for each entity and indexes these concordances with Lucene. Each operation is done in a scalable distributed way.

6.4.1.1 Processing News Articles with the Lydia Pipeline

We process news articles through the Lydia pipeline performing named entity recognition and coreference set identification, and obtain output in XML format with entities marked up with `<pn>` tags and a category assigned to each entity occurrence. An example of pipeline output is given in Figure 6.2. For bulk processing of large amount of news, we run multiple instances of the Lydia pipeline using the Condor job scheduling system [90].

6.4.1.2 Dealing with Near-Duplicate Articles

The web contains a substantial fraction of duplicate and near-duplicate documents [50]. We deal with the duplicate and near-duplicate article problem by eliminating

```
<p> <pn category = "COMPANY"> Eli/NNP Lilly/NNP
</pn> encouraged/VBD primary/JJ care/NN
physicians/NNS to/TO use/VB <pn category = "DRUG">
Zyprexa/NNP </pn> ,/, a/DT powerful/JJ drug/NN
for/IN <pn category = "DISEASE"> schizophrenia/NN
</pn> and/CC <pn category = "DISEASE"> bipolar/JJ
disorder/NN </pn> ,/, in/IN patients/NNS who/WP
did/VBD not/RB have/VB either/DT condition/NN ,/,
according/VBG to/TO internal/JJ <pn category =
"COMPANY"> Eli/NT Lilly/NNP </pn> marketing/NN
materials/NNS ./.  </p>
```

Figure 6.2: One paragraph example of Lydia pipeline output.

duplicate sentences. If two articles are near-duplicates, there is a high chance they will contain many exact duplicate sentences. We consider two sentences equivalent for the purpose of duplicate removal if after replacing every sequence of whitespace characters with one space and lowercasing all letters they have the same MD5 hash codes.

To obtain the set of unique sentences (as well as for other tasks) we use the Hadoop open-source implementation [11] of Google's MapReduce distributed computation model [39]. The map function takes an XML news article as input and produces (MD5 hash, sentence) tuples as output for every sentence in the input article that contains at least one entity marked up with a `<pn>` tag. The reduce function takes (MD5 hash, list of sentences) as input and outputs (MD5 hash, first sentence). The output of this MapReduce job is a collection of files containing all unique sentences of the input corpus.

6.4.1.3 Collecting Context of Every Entity

To produce a searchable document for each entity, we collect all the sentences containing that entity and concatenate them together. This is again done by means of a

```
Earlier this week George W. Bush defended the missile defense plan for Eastern
Europe .

70 names were on the list , including national figures like George W. Bush
.

Another State Department official confirmed her status as the most senior
Arab-American in the George W. Bush administration .

They say they want a candidate who does not have the strident approach to
the war that Mr.  George W. Bush does .

I would n't approve of him again unless George W. Bush stopped the war in
Iraq .

And , for that opportunity , the Democrats can thank George W. Bush .

34 percent of Mr.  George W. Bush 's 2004 voters are now critical of George
W. Bush handling of the war .

Now , more than 2 years later , the majority of those voters say they are
satisfied with the George W. Bush presidency .
```

Figure 6.3: An excerpt from a concordance document generated for the entity "George W. Bush".

MapReduce job that takes a collection of unique sentences as input. The map function produces a set of (entity$_i$, sentence) tuples for every sentence, where entity$_i$ goes over all distinct entities occurring in the sentence. The reduce function takes (entity, list of sentences) as input and adds a document with two fields (entity, concatenation of sentences) to a Lucene index as output.

To estimate the increase in the relative index size compared to conventional document indexing, we calculated $\sum_s n_e(s)len(s)/\sum_s len(s)$, where s is a sentence and n_e is the number of entities in it. This statistic accounts for the fact that sentence s gets included in $n_e(s)$ concordance documents, and equals to 2.6 in our corpus.

6.4.2 Searching

Suppose document d_i is a concatenation of all unique sentences from the input corpus that contain the entity e_i. As the result of steps described in Section 6.4.1,

Entity	Score	Month
Muhammad Yunus	1.000	200610
Muhammad Yunus	0.649	200612
Grameen Bank	0.548	200610
Bangladesh	0.509	200610
Muhammad Yunus	0.428	200611
Nobel Peace Prize	0.363	200612
Grameen Bank	0.336	200612
Nobel Peace Prize	0.336	200610
Nobel	0.324	200610
Dhaka	0.313	200610
Bangladeshi	0.313	200610

Entity	Score
Muhammad Yunus	1.000
Grameen Bank	0.548
Bangladesh	0.509
Nobel Peace Prize	0.363
Nobel	0.324
Dhaka	0.313
Bangladeshi	0.313

Table 6.6: On the left—Lucene results for a "MUHAMMAD YUNUS" query indexed in a time-dependent manner. On the right—the same results aggregated so that $score(entity) = \max_i score(entity, month_i)$.

we get a Lucene [8] index of documents with fields (e_i, d_i).

Lucene's scoring scheme must be modified to meaningfully search these automatically generated documents. The scoring formula used in Lucene, assuming that we are searching a single field, giving equal weights to all query terms, and omitting factors irrelevant to document ranking, is the following:

$$score(q, d) = coord(q, d) lengthNorm(d) \times \sum_{t \in q} tf(t, d) idf(t)^2$$

In the default implementation of Lucene scoring [6] $lengthNorm(d) = \frac{1}{\sqrt{numTerms(d)}}$, where $numTerms(d)$ is the number of terms in the document d. With this type of document length normalization, the top results almost always turn out to be very short documents, where the few matching terms gain an enormously high weight. To compensate we set $lengthNorm(d) \equiv 1$ regardless of the document d.

6.4.3 Time-Dependent Indexing and Search

News search appears different from other document search because the news is a continuous flow of text, and the reader's interest is often focused on the recently added documents. This has implications on the design of news article and entity retrieval systems [38]. In particular, the impact of recent text on search results should be higher than that of older text. Similar phenomena hold (to a lesser extent) in general web document search.

Web search engines have limits on indexed document size (200 KB - 1.1 MB, [24]). Similarly, in Lucene the maximum number of terms that can be indexed in a single field is limited by the amount of main memory. But for a large enough input corpus our automatically generated documents can grow up to an indefinitely large size.

To address the time dependency problem and the document size restriction problem, we create multiple Lucene documents for each entity, with each document containing the context of all occurrences of the entity in a particular time period (in our case, month). Every concordance document in this case has three fields: (entity, concordance, month), and can be uniquely identified by the (entity, month) pair. We can view the Lucene results for our time-dependent index as a list of (entity, month) pairs with associated scores. An example of such results is shown in Table 6.6.

There are multiple ways of assigning scores to an entity e based on multiple scores assigned to $(e, month_i)$ pairs:

- Exponential decay: This expresses exponentially decreasing user interest in past news.

  $$score(e) = \sum_{i=0}^{k} \exp(-\alpha(k-i)) score(e, month_i) \tag{6.1}$$

  where k is the index of the current month, assuming that 0 corresponds to the earliest month for which news are included in the index.

- Maximum value:

  $$score(e) = \max_{i=0,\dots,k} score(e, month_i) \tag{6.2}$$

The advantage of this method is speed of computation: if n top-scoring entities are requested, the scan of an (entity, month) hit list returned by Lucene can be stopped once n unique entities have been seen. The disadvantage of this method is that entities that were once very popular score higher than entities that have had steady popularity without high peaks. We currently use this method for our demo at `http://www.textmap.com` due to its speed. The right part of Table 6.6 shows the entity scores calculated using this method.

6.4.4 Modeling User Interest in an Entity

We hypothesize that the number of web queries containing an entity expresses user interest in that entity as a function of time. We use the daily scale instead of monthly for these experiments because only three months of web query data are available to us. We try to predict daily entity frequency in queries using its frequency in the news using the following models:

- Exponential decay with window w (including $w = \infty$):

$$h_{\beta,w}(e,i) = \sum_{i=\max(0,d-w+1)}^{d} \exp(-\beta(d-i))n(e,i) \tag{6.3}$$

  where e is an entity, d is the index of the current day and $n(e,i)$ is the frequency of the entity e in the news on day i.

- Historical mean:

$$h_{average}(e,i) = \frac{1}{d+1}\sum_{i=0}^{d} n(e,i) \tag{6.4}$$

- A convex combination of both:

$$h_{\lambda,\beta,w}(e,i) = (1-\lambda)h_{\beta,w}(e,i) + \lambda h_{average} \tag{6.5}$$

Then, we optimize model parameters to maximize the Pearson correlation between $h(e,i)$ and the actual frequency $q(e,i)$ of entity e in web queries on day i. In our experiments correlation is averaged across 152 entities chosen as the intersection of 1000 most frequent entities in the news and in web queries. A correlation of 0.275 is reached by (6.4). A higher correlation of 0.349 is reached by (6.5) when $\beta = 0.01$ and $\lambda = 0$, showing that (6.3) is a better model than (6.4). We use the exponential decay factor $\alpha = \beta_{optimal} \times 30.4$ (average days per month) in our scoring function (6.1).

The dependency of the correlation given by $h_{0,w}(e, i)$ on the window size w is shown in Figure 6.4. The optimal $w = 28$ suggests the usual span of user interest in an entity after it is mentioned in the news. However, this model would not be suitable for search result scoring, because it discards old references to an entity which could undoubtedly be of interest to the user. Therefore, we use $w = \infty$ in the implementation of the exponential decay model described in Section 6.4.3.

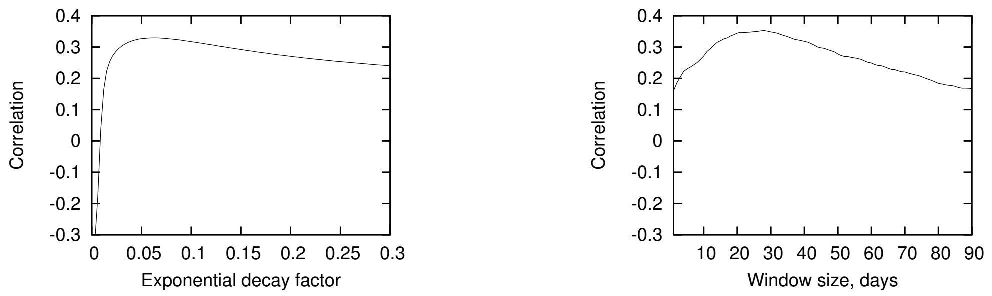


Figure 6.4: The correlation between predicted and actual entity frequency in queries (left) depending on the exponential decay factor β for model (right) when predicted by summing entity frequency in the news on last w days.

6.5 Evaluation

We have designed and implemented a search engine returning entities related to the user's query instead of regular web documents. Evaluating an entity search engine is a non-trivial problem. TREC (Text REtrieval Conference) provides a de facto standard to document retrieval and question answering systems evaluation. The closest benchmark to entity search that TREC provides is the Expert search task in the Enterprise track, which may be viewed as a very specific instance of the problem we consider. However, the general problem of returning entities relevant to the user's query that we address here is different from all of the mentioned TREC tasks.

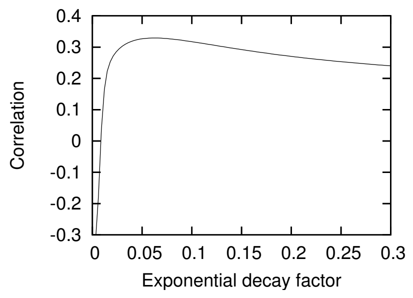


Figure 6.5: The correlation between predicted and actual entity frequency in queries depending on the exponential decay factor β for model (6.3), $w = \infty$. The maximum correlation of 0.33 is achieved for $\beta = 0.062$.

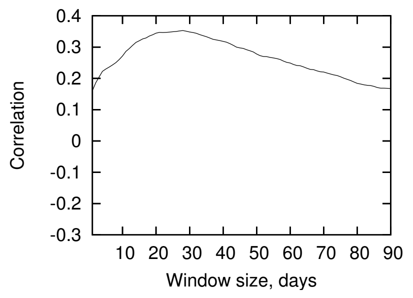


Figure 6.6: The correlation between predicted and actual entity frequency in queries when predicted by summing entity frequency in the news on last w days. The maximum correlation of 0.353 is achieved for $w = 28$.

Our Lydia text analysis system already contains a way to measure how strongly two entities are related to each other (*juxtaposition score*), expressed as an upper bound of the probability of them occurring in the same sentence under the assumption of their independence [62]. This measure is similar to collocation significance measures in natural language processing [64], but instead of word occurrences next to each other we consider entity occurrences within the same sentence. Juxtaposition scores are calculated based on the entire news corpus. It would be natural for our search engine, when given a query that is a known entity itself, to return results close to the list of entities having top juxtaposition scores with the queried entity. The significant improvement of our search engine over the juxtaposition technique, however, is its ability to answer free text queries.

Another quite simple benchmark that we considered is based on the observation that when an entity is given to our search engine as a query, ideally it should appear as the top result. Therefore, calculating statistics of the rank of the queried entity in the results is a way to check basic correctness of our search engine performance. It is covered in Section 6.5.3.

6.5.1 Comparison with Juxtaposition Lists

A method to identify entities that occur near a particular entity in an overrepresented way (i.e. more frequently than it would happen randomly) is described in [62]. Suppose, n_a and n_b are numbers of sentences in the entire news corpus that contain entities a and b respectively, F is the number of sentences containing them both, and N is the total number of sentences in the corpus. Then, according to [62], the probability of the observed number of occurrences under the assumption that these two entities are independent is not more than

$$P_{bound}(n_a, n_b, F, N) = \left(\frac{e^{\frac{FN}{n_a n_b} - 1}}{\left(\frac{FN}{n_a n_b}\right)^{\frac{FN}{n_a n_b}}} \right)^{\frac{n_a n_b}{N}} \tag{6.6}$$

We call $-\log$ of (6.6) the *juxtaposition score* of the entities a and b, meaning that the higher the juxtaposition score, the more dependency exists between the two entities. If for a given entity a we retrieve a list of $k-1$ entities $b_1, \dots, b_{k-1}$ from our corpus that have highest juxtaposition scores with entity a, we will get a list of $k-1$ "most associated" with a entities in the sense of juxtaposition scores. It is natural to add the entity a itself onto the top of this list as being ultimately the most associated with itself. The resulting $(a, b_1, \dots, b_{k-1})$ list is what we compare against the results of our search engine given the query a.

For comparison of top k lists we use the K_{min} distance measure described in [44]. According to [44], the K_{min} distance measure can be calculated as follows for two top k lists τ_1 and τ_2:

$$K_{min}(\tau_1, \tau_2) = K^{(0)}(\tau_1, \tau_2) = \sum_{\{i,j\} \subseteq D_{\tau_1} \cup D_{\tau_2}} \bar{K}^{(0)}_{i,j}(\tau_1, \tau_2)$$

where D_{τ_1} and D_{τ_2} are the sets of elements of τ_1 and τ_2 respectively, and $\bar{K}^{(0)}_{i,j}(\tau_1, \tau_2)$ is defined as 0 if the elements i and j appear in the same order both in τ_1 and τ_2 and 1 if they appear in different order. An element that does not appear in one of the lists is considered appearing at the bottom of that list. If the elements i and j appear in one top k list but do not appear in the other top k list,

$$\bar{K}^{(0)}_{i,j}(\tau_1, \tau_2) = 0 \tag{6.7}$$

(the reason for the notation $K^{(0)}$).

To make these scores calculated for different list sizes comparable, we divide them by $\binom{k}{2}$ where k is the size of the top lists. This corresponds to the value of the K_{min} distance measure between two top lists that contain the same k elements in the reverse order. This distance measure, however, is even higher when two lists are completely disjoint. Specifically, if $D_{\tau_1} = \{a_1, a_2, \ldots, a_k\}$ and $D_{\tau_2} = \{b_1, b_2, \ldots, b_k\}$, with $D_{\tau_1} \cap D_{\tau_2} = \emptyset$, then $\bar{K}^{(0)}_{a_i,a_j}(\tau_1, \tau_2) = \bar{K}^{(0)}_{b_i,b_j}(\tau_1, \tau_2) = 0$ for all $1 \le i < j \le k$ according to (6.7), and $\bar{K}^{(0)}_{a_i,b_j}(\tau_1, \tau_2) = 1$, because each of the elements a_i and b_j, appearing in only one of the lists τ_1 and τ_2, is considered appearing at the bottom of the other list, and therefore the pair of elements (a_i, b_j) appears in the two top k lists in different order for all $1 \le i \le k$, $1 \le j \le k$. Consequently,

$$K_{min}(\tau_1, \tau_2) = k^2. \tag{6.8}$$

6.5.1.1 Results by Entity Category

To determine how entity category effects search results, we have experimented with the top 10,000 entities in each category. The distance measures between juxtaposition lists and search results are given in Table 6.7. The search results turn out to be the closest to the juxtaposition-based results for the "person" category, probably because Lydia provides more precise identification and categorization of "person" entities.

Figure 6.7 shows the distribution of top list distances between juxtaposition results and phrase query results for the person category. The small size of the spike at $(k-1)^2/\binom{k}{2} = 1.8$ indicates that the portion of queries with disjoint (except for the top result) lists based on juxtapositions and entity index was rather insignificant.

Category	Mean	Std Dev
person	0.57	0.32
all	0.83	0.54
institute	0.86	0.44
address	0.88	0.47
political	0.89	0.54
company	0.92	0.45
university	0.95	0.43
place	0.96	0.51
unknown	1.06	0.57
last name	1.47	0.46

Table 6.7: Distances between search results and juxtaposition lists by category.

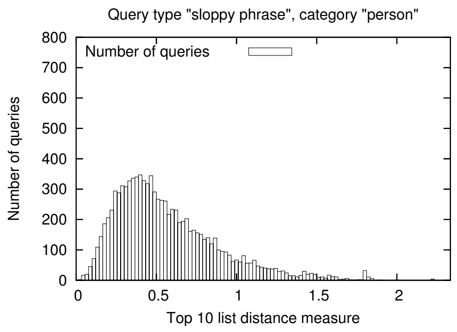


Figure 6.7: Distribution of top list distances for phrase queries for the person category.

6.5.1.2 Results by Query Type

There are many possible ways to interpret an unstructured search query entered by the user. We evaluate the following approaches in order to minimize the distance measure between search results and juxtaposed entities for an entity query:

- *"Bag of words" query*. In this case, the terms in the query are allowed to occur anywhere in the target document independently of each other, and a bonus score proportional to the number of appearing terms is given to each document.

	Phrase	Bag of words	Combination
Distance mean	0.831	1.154	1.148
Distance std dev	0.544	0.564	0.564

Table 6.8: Distance measure between search results and juxtapositions for different query types for 9919 entities used as queries.

Slop value	Mean	Variance
0	0.825	0.542
1	0.828	0.543
2	0.830	0.544
# of terms	0.831	0.544
3	0.832	0.544

Table 6.9: Distance measure between juxtapositions and search results for phrase queries depending on the slop value.

- *"Phrase" query with different slop values.* The terms are required to appear in the document next to each other, but the order might differ from that specified by the user. The maximum edit distance where units correspond to movements of words should not exceed the slop value.

- *Combination of both "bag of words" and phrase queries.* We give the phrase query a significantly higher weight("boost" in Lucene terminology), so that when too few results are found for the phrase query, the results of the bag of words query are mixed in.

Table 6.8 shows the result of comparison of top-10 lists returned for about 10,000 most popular entities with top juxtaposition lists for the same entities. Of the three query types we used, the phrase query matches juxtapositions the best. In phrase queries in the Table 6.8, and in the experiments in Section 6.5.1.1, the slop value equal to the number of terms in the query is used.

Table 6.9 shows the dependency of the distance measure for phrase queries on the slop value. The smallest distance measure is achieved when the slop value is

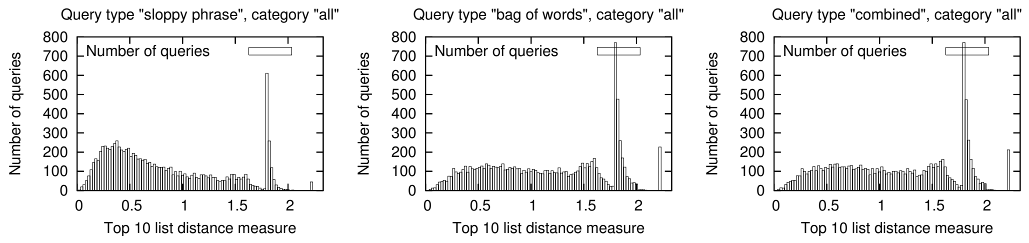


Figure 6.8: Distribution of top list distances for “phrase”, “bag of words” and “combined” queries.

Second(→) First(↓)	Perfect, the same	Perfect, different	Partial, the same	Partial, different	None
Perfect	2185059	360191	242165	411329	283483
Partial	164635	435530	11631065	1233368	644294
None	N/A	431160	N/A	934830	4402650

Table 6.10: Frequencies of different combinations of types of matches of consecutive (within 10 minutes) queries by the same user with entities.

zero. However, all distance measure differences in the Table 6.9 are within about 1%, so we use the slop value equal to the number of terms in the query for our following experiments, as the most flexible choice.

Figure 6.8 shows the distribution of distance measures for three mentioned types of queries. The spike at 1.8 corresponds to the case when two lists only have the top element in common, and $K_{min}(\tau_1, \tau_2) = (k-1)^2$, because in this case $\bar{K}_{i,j}^{(0)} = 1$ if and only if i and j are non-top elements from different lists. This can be explained by the fact that we artificially add the queried entity at the top of the the juxtaposition result list for that entity, and thus it has a higher chance to match with the top Lucene result.

6.5.1.3 Statistical Significance

It is interesting to note that the correlation between the two types of lists of relevant entities that we are comparing to each other is largely explained by the same entity

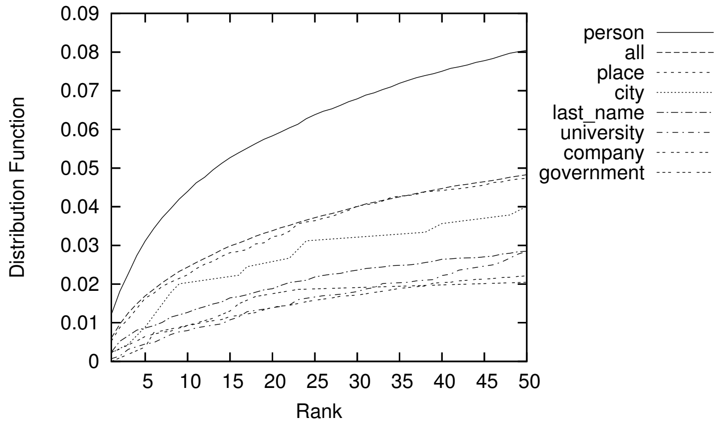


Figure 6.9: Predicting the user's next query, i.e. the probability that an AOL search query ranks among the top k entities associated with the previous query (unique pair frequencies).

frequency and juxtaposition frequency data being used for both algorithms. For instance, if two result lists were independent, the probability of the top k lists drawn from N documents being completely disjoint would be

$$\frac{\binom{N-k}{k}}{\binom{N}{k}} \approx \left(\frac{N-k}{N}\right)^k \approx \exp\left(-\frac{k^2}{N}\right) \approx 0.9999 \tag{6.9}$$

with $k = 10$ and $N = 10^6$. Disjoint lists correspond to the distance measure of $\approx$ 2.222, but in any of our experiments at most 9500 queries out of 10000 have this distance measure. Therefore, the p-value of our observations is not more than $\sum_{k=0}^{9500} \binom{10000}{k} 0.9999^k 0.0001^{10000-k} < 10^{-100}$. This extremely low p-value can be explained by the fact that in our case the two top-k lists are not really independent at all, but are based on the same input data.

6.5.2 Query Prediction using Juxtapositions

Modern search engines suggest refined search queries to users, by (1) identifying likely spelling errors, and (2) analyzing query frequencies in search logs by collaborative filtering. To augment these techniques, we propose using our entity search engine to suggest entities related to the previous search query as navigational links.

Category	Query found	Rank mean	Rank std dev
person	9995	0.02	0.22
all	9919	0.19	0.73
unknown	9664	0.46	1.20
place	9424	0.46	1.28
address	9790	0.68	1.30
institute	9658	0.75	1.49
political	9300	0.83	1.63
company	9557	0.89	1.61
university	9461	1.27	1.81
last name	7052	2.33	2.57

Table 6.11: Queried entity position depending on category.

The critical question is how often we can predict the user's next query from their current search.

In an attempt to answer this question, we examined successive queries by distinct users in the AOL query data set. (The IDs of all users remained anonymous through this process). We considered the 360,191 cases where users transitioned from one perfect entity query to a different perfect entity query. We were interested in the extent to which our search engine would report the latter entity in response to the first.

For efficiency of experimentation, we used the second entity's rank in the first entity's juxtaposition list as a proxy for the predictions of our search engine. Figure 6.9 presents our results for distinct pairs of perfect entity search queries as a function of rank position. It shows that our predictions are most accurate for people queries, predicting the next query over 8% of the time for rank=50. The decision procedure to refine the ranked list has not been optimized, and we anticipate substantially lower ranks will suffice for this level of predictability in practice.

These results are quite encouraging. They demonstrate that our methods can predict roughly 5-10% of subsequent user entity queries; enough to justify displaying our navigational shortcuts to aid their search.

6.5.3 Queried Entity Rank in Results

A basic way to evaluate entity retrieval performance is to measure how many times, when an existing entity is used as a query, it is returned as the top result. Generalizing this, we measure the rank of an entity in the search results for that entity used as a query, and expect it to be close to the top position.

Table 6.11 shows the number of queries for which the queried entity was found in the result list, the mean and the standard deviation of the zero-based rank of the queried entity in the result list for different categories. It uses the "phrase" query interpretation which turns out to be the best according to our criteria (see section 6.5.1.2). For most categories, in 95% of cases the queried entity was present in the result list and on average appeared within the top two positions of the result. The high number (almost 30%) of entities in the "last name" category not found in the result lists for those entities can be explained by the result lists being dominated by person names consisting of a first and a last name.

6.6 Conclusions and Future Work

It is becoming recognized that the next generation of search engines will need to be aware of entities in addition to searching unstructured text. We analyzed web query logs and found that up to 87% of web queries contain part that is recognizable as a news or Wikipedia entity. We then designed and implemented a prototype entity search engine that automatically composes a concordance capturing the context of all occurrences of each entity and leverages an off-shelf document retrieval technology (Lucene) to search these documents. The prototype is available at `http://www.textmap.com`.

Directions for future work include:

- Alternative evaluation techniques, possibly converting TREC list questions to queries to our search engine and finding the percentage of correct answers in the top list.

- Customizing our approach for product review search.

- Detection of target entity categories from the query and filtering results accordingly.
- Integration of entity search results into conventional web search engines.

Chapter 7

Conclusions

This thesis was inspired by the potential of the Lydia news analysis system to provide useful data to social science researchers. We have come a very long way to realizing this potential. In particular, we have met the following challenges in our approach to this problem:

- *Scalability.* To keep up with the ever-increasing daily volume of text streams available online and the variety of corpora, the Lydia system had to be made to run on a computer cluster and unrestricted by a single relational database bottleneck as the previous system had been. We achieved this by re-designing the data aggregation part of the system on top of the Hadoop map-reduce framework. A particularly difficult part of this was reconciling the batch processing model of the map-reduce paradigm with the necessity of regular updates to our entity statistics datasets.

- *Ease and flexibility of access.* Once extracted from the text and stored in the Lydia depository, the entity statistics data is easily accessible through our depository server. It provides a variety of access granularity options allowing to capture both temporal and spatial distributions of all entity statistics supported. We provide a web frontend for interactive exploration and visualization of entity statistics and data download, as well as an API for custom experimental code.

- *Modularity and extensibility.* We introduced an intermediate data management and scheduling layer between Hadoop and our backend data aggregation/processing phases that operates on the abstraction of *artifacts*. An artifact in the Lydia framework is a dataset stored in parts identified by a time period (such as date range), having a certain key/value or index structure, and computed and updated in a certain way. All of entity statistics and indexes in the core Lydia depository were implemented in terms of this artifact abstraction. Experimental backend aggregation datasets can be created by setting up an additional depository and using some artifacts from the core depository as inputs.

We have presented performance evaluation results of our system and discussed optimization of certain parameters such as the number of simultaneously built artifacts so as to minimize the total text corpus processing time. These results show a significant increase in performance and scalability compared to the previous version of the Lydia system. With the current rapid growth of applications built on top of Hadoop, such evaluations are becoming of significant interest to the developer community.

A useful feature of the new Lydia architecture is its statistics aggregation subsystem allowing to collect temporal, spatial, and juxtaposition statistics for groups of entities given the same statistics for individual entity names and the entity to group mapping. The system supports entity groupings by their co-reference set, autodetected ethnicity, and the most-associated nationality inferred from the juxtaposition data. Using the ethnicity classifier of [5] we have performed geographic news analysis of cultural groups, detected interesting time-series trends in cultural/ethnic/linguistic groups frequency and sentiment, and examined inter-group interactions.

As another study, indirectly related to building an entity analysis system for the social sciences, we have considered the problem of finding entities the most relevant to the given full-text query. We have assessed the importance of this problem by analyzing the AOL query logs and found out that a significant fraction of real web queries represent or contain named entities that are present in our real-world corpora. Then we proceeded to building an entity search engine that works by constructing a “concordance” document corresponding to each entity. These concordance documents contain all context of the given entity in the text corpus. We implemented entity search by indexing and searching these concordance documents with the open-source Lucene information retrieval library. The results proved to be quite consistent with our notion of juxtapositions, justifying the use of our entity search engine as a part of the new Lydia text analysis system.

The large-scale sentiment scoring subsystem is one of the most important parts of our system for social science applications. We have validated the Lydia sentiment scoring methodology using text from international newspapers and parallel corpora translated to English with state-of-the-art machine translation technology. As significant correlations were observed between entity sentiment scores coming from various language sources and using different translators, we concluded that our sentiment scoring approach is largely independent of the source language and the translator used.

7.1 Future Work

There are many areas where the new Lydia system can still be extended and improved:

- Using the newly introduced sentiment word histogram artifact that allows to obtain time series of every sentiment word used in conjunction with a given entity (see Figure 2.9), the scores assigned to sentiment words can be refined to make polarity better reflect certain real-world time series, such as politician approval / poll ratings.

- We are still working on refining our entity globality scoring algorithm and obtaining meaningful entity globality time series.

- We anticipate that the depository server will eventually become a bottleneck in our system as we increase the volume of data we process. Its amount of memory limits the total number of parts randomly accessible artifacts can be split into. Using a column-oriented database such as HBase [7] or Hypertable [102] for the depository server should help solve this problem.

- The web frontend still offers room for improvement. Inline help could be added to explain various non-obvious fields on data retrieval pages. Spelling correction could be included to handle mistyped entity names. The source group specification interface could be made more intuitive.

- More collaboration with social scientists. In one of our current projects we are using our dataset and analysis tools to develop a methodology for identifying and explaining cumulative advantage effects in news data. This would allow to study the influence of an entity's past popularity on its future references in the news.

- We could make it possible for the users to upload and process custom document corpora to build their own depositories.

Bibliography

[1] Alias-i Inc. LingPipe. http://www.alias-i.com/lingpipe.

[2] Wikipedia. May 2007. http://en.wikipedia.org/wiki/May_2007.

[3] 101tec. Hypertable. http://hypertable.org/.

[4] E. Adar. User 4XXXXX9: anonymizing query logs. In E. Amitay, C. G. Murray, and J. Teevan, editors, *Query Log Analysis: Social And Technological Challenges. A workshop at the 16th International World Wide Web Conference (WWW 2007)*, May 2007.

[5] A. Ambekar, C. Ward, J. Mohammed, S. Reddy, and S. Skiena. Name-Ethnicity Classification from Open Sources. In *Proc. of the ACM SIG-KDD*, 2009.

[6] Apache Software Foundation. Apache Lucene—Scoring. http://lucene.apache.org/java/docs/scoring.html.

[7] Apache Software Foundation. HBase. http://hadoop.apache.org/hbase/.

[8] Apache Software Foundation. Lucene. http://lucene.apache.org/.

[9] Apache Software Foundation. Nutch. http://lucene.apache.org/nutch.

[10] Apache Software Foundation. Solr. `http://lucene.apache.org/solr`.

[11] Apache Software Foundation. The Hadoop Project. `http://lucene.apache.org/hadoop`.

[12] J. Appel. More "reliable" Wikipedia soon to launch. eSchool News, `http://www.eschoolnews.com/news/showstoryts.cfm?Articleid=6877`, 2007.

[13] M. Arrington. AOL Proudly Releases Massive Amounts of Private Data. `http://www.techcrunch.com/2006/08/06`, 2006.

[14] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 2670–2676, 2007.

[15] M. Barouni-Ebrahimi and A. A. Ghorbani. On query completion in web search engines based on query stream mining. In *WI '07: Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 317–320, Washington, DC, USA, 2007. IEEE Computer Society.

[16] M. Bautin, A. Dwarakinath, and T. Chiueh. Graphics engine resource management. In *Proc. of the 15th Annual Multimedia Computing and Networking Conference (MMCN 2008)*, San Jose, California, USA, January 2008.

[17] M. Bautin and S. Skiena. Concordance-based entity-oriented search. In *Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 586–592, 2007.

[18] M. Bautin and S. Skiena. Concordance-based entity-oriented search. *Web Intelligence and Agent Systems: An International Journal*, 7(4), 2009.

[19] M. Bautin, L. Vijayarenu, and S. Skiena. International Sentiment Analysis for News and Blogs. In *Proc. of the International Conference on Weblogs and Social Media*, Seattle, WA, 2008.

[20] F. Benamara, C. Cesarano, A. Picariello, D. Reforgiato, and V. Subrahmanian. Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In *Proc. of the ICWSM'07*, pages 203–206, March 2007.

[21] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[22] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[23] D. Bond, J. Bond, C. Oh, J. C. Jenkins, and C. L. Taylor. Integrated Data for Events Analysis (IDEA): An Event Typology for Automated Events Data Development. *Journal of Peace Research*, 40:733–745, Nov 2003.

[24] S. Bondar. Search Engine Indexing Limits: Where Do the Bots Stop? `http://www.sitepoint.com/article/indexing-limits-where-bots-stop`, 2006.

[25] J. Box-Steffensmeier, D. Darmofal, and C. Farrell. The endogenous relationship of campaign expenditures, expected vote, and media coverage. In *American Political Science Association annual meeting*, 2005.

[26] H. Brandenburg. Revisiting the "Liberal Media Bias": A Quantitative Study into Candidate Treatment by the Broadcast Media During the 2004 Presidential Election Campaign. In *Proc. of the Annual Meeting of the American Political Science Association*, Philadelphia, Sep 2006.

[27] A. Broder. A taxonomy of web search. *ACM Special Interest Group on Information Retrieval (SIGIR) Forum*, 36(2):3–10, 2002.

[28] B. Carpenter. Phrasal queries with LingPipe and Lucene. In *Proc. of the 13th Meeting of the Text Retrieval Conference (TREC)*, Gaithersburg, Maryland, 2004.

[29] U. S. Census. Maps of American Ancestries. http://en.wikipedia.org/wiki/Maps_of_American_ancestries, 2000.

[30] C. Cesarano, B. Dorr, A. Picariello, D. Reforgiato, A. Sagoff, and V. Subrahmanian. Oasys: An opinion analysis system. In *Proc. of the AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*, 2004.

[31] C. Cesarano, A. Picariello, D. Reforgiato, and V. Subrahmanian. The OASYS 2.0 Opinion Analysis System. In *ICWSM'07*, pages 313–314, March 2007.

[32] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: a distributed storage system for structured data. In *OSDI '06: Proc. of the 7th symposium on Operating Systems Design and Implementation, citeulike-article-id = 3765219*, pages 205–218, Berkeley, CA, USA, 2006. USENIX Association.

[33] T. Cheng and K. C.-C. Chang. Entity search engine: towards agile best-effort information integration over the web. In *Proc. of the Third Biennial Conference on Innovative Data Systems Research*, pages 108–113, Jan 2007.

[34] T. Cheng, X. Yan, and K. C.-C. Chang. Supporting entity search: a large-scale prototype search engine. In *SIGMOD '07: Proc. of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 1144–1146, New York, NY, USA, 2007. ACM.

[35] J. Chu-Carroll, J. Prager, K. Czuba, D. Ferrucci, and P. Duboue. Semantic search via XML fragments: a high-precision approach to IR. In *Proc. of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 445–452, New York, NY, USA, 2006. ACM Press.

[36] Concurrent Inc. Cascading. `http://www.cascading.org`.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition.* MIT Press and McGraw-Hill, 2001.

[38] G. M. D. Corso, A. Gullí, and F. Romani. Ranking a stream of news. In *Proc. of the 14th International Conference on World Wide Web*, pages 97–106, New York, NY, USA, 2005. ACM Press.

[39] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proc. of the OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150.

[40] P. V. Dijck. Better search engine design: beyond algorithms. `http://www.onlamp.com/pub/a/onlamp/2003/08/21/better_search_engine.html`, Aug. 2003.

[41] T. L. Dixon, C. L. Azocar, and M. Casas. The portrayal of race and crime on television network news. *Journal of Broadcasting & Electronic Media*, 47(4), 2003.

[42] Facebook Inc. Cassandra: A Structured Storage System on a P2P Network. `http://code.google.com/p/the-cassandra-project/`.

[43] Facebook Inc., Apache Software Foundation. Hive. `http://wiki.apache.org/hadoop/Hive`.

[44] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[45] D. J. Gerner, R. Abu-Jabr, P. A. Schrodt, and O. Yilmaz. Conflict and Mediation Event Observations (CAMEO): A New Event Data Framework for the Analysis of Foreign Policy Interactions.

[46] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43, 2003.

[47] N. Godbole, M. Srinivasaiah, and S. Skiena. Large-Scale Sentiment Analysis for News and Blogs. In *Proc. of the International Conference on Weblogs and Social Media*, Mar. 2007.

[48] N. Godbole, M. Srinivasaiah, and S. Skiena. Large-Scale Sentiment Analysis for News and Blogs. In *Proc. of the International Conference on Weblogs and Social Media*, pages 219–222, Mar. 2007.

[49] V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In *Proc. of the 8th Conference of the European Chapter of the*

Association for Computational Linguistics, pages 174–181, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

[50] M. Henzinger, B.-W. Chang, B. Milch, and S. Brin. Query-free news search. In *Proc. of the 12th International Conference on World Wide Web*, New York, NY, USA, May 2003. ACM Press.

[51] K. Hiroshi, N. Tetsuya, and W. Hideo. Deeper sentiment analysis using machine translation technology. In *COLING '04: Proc. of the 20th International Conference on Computational Linguistics*, page 494, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[52] IBM Corporation. WebSphere Translation Server for Multiplatforms. `http://www-306.ibm.com/software/pervasive/ws_translation_server`.

[53] IBM Corporation. Guidelines for Writing Content that Will Be Machine-Translated, Sept. 2001.

[54] M. A. Johnson. Predicting News Flow from Mexico. *Journalism & mass communication quarterly*, 74:315, 1997.

[55] T. Kato. International Compact with Iraq. `http://www.imf.org/external/np/speeches/2007/050307.htm`.

[56] J. H. Kil, L. Lloyd, and S. Skiena. Question Answering with Lydia. In *Proc. of the Fourteenth Text Retrieval Conference (TREC)*, 2005.

[57] S.-M. Kim and E. Hovy. Determining the sentiment of opinions. In *Proc. of the 20th International Conference on Computational Linguistics*, page 1367, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[58] M. Lebo. Personal communication, 2008.

[59] LexisNexis. LexisNexis Academic. `http://www.lexisnexis.com/us/lnacademic`.

[60] L. Lloyd. *Lydia: A System for the Large Scale Analysis of Natural Language Text*. PhD thesis, Stony Brook University, 2006.

[61] L. Lloyd, P. Kaulgud, and S. Skiena. Newspapers vs. Blogs: Who Gets the Scoop? In *Proc. of the Computational Approaches to Analyzing Weblogs (AAAI-CAAW 2006)*, Stanford University, 2006.

[62] L. Lloyd, D. Kechagias, and S. Skiena. Lydia: A system for large-scale news analysis. In *SPIRE*, pages 161–166, 2005.

[63] L. Lloyd, A. Mehler, and S. Skiena. Identifying co-referential names across large corpora. In *Proc. of the 17th Annual Symposium on Combinatorial Pattern Matching (CPM 2006)*, volume LNCS 4009, pages 12–23, July 2006.

[64] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, May 1999.

[65] C. McClelland. World Event/Interaction Survey (WEIS) Project, 1966-1978.

[66] A. Mehler, Y. Bao, X. Li, Y. Wang, and S. Skiena. Spatial Analysis of News Sources. In *IEEE Trans. Vis. Comput. Graph.*, volume 12, pages 765–772, 2006.

[67] A. Mehler and S. Skiena. Expanding network communities from representative examples. *ACM Trans. Knowledge Discovery from Data (TKDD)*, 2009.

[68] R. Mihalcea, C. Banea, and J. Wiebe. Learning multilingual subjective language via cross-lingual projections. In *Proc. of the 45th Annual Meeting of the ACL*, pages 976–983, June 2007.

[69] G. A. Miller. WordNet: a lexical database for English. *Commun. ACM*, 38(11):39–41, 1995.

[70] D. Mutz. Media Fairness? It's in the Eye of the Beholder, National Annenberg Election Survey Data Show. `http://www.annenbergpublicpolicycenter.org/Downloads/Releases/NAES%202008/pressfairnessmay12008.pdf`, May 2008.

[71] P. Ogilvie and J. P. Callan. Experiments using the Lemur toolkit. In *Text Retrieval Conference*, 2001.

[72] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: a not-so-foreign language for data processing. In *SIGMOD '08: Proc. of the 2008 ACM SIGMOD International Conference on Management of data*, pages 1099–1110, New York, NY, USA, 2008. ACM.

[73] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: fact extraction in the fast lane. In *ACL '06: Proc. of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 809–816, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[74] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of the ACL*, pages 271–278, 2004.

[75] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[76] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale '06: Proc. of the 1st International Conference on Scalable information systems*, page 1, New York, NY, USA, 2006. ACM.

[77] L. Philips. Hanging on the metaphone. *Computer Language*, 7, 1990.

[78] L. Phillips. The Double Metaphone Search Algorithm. *C/C++ Users Journal*, June 2000.

[79] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the data: Parallel analysis with Sawzall. *Sci. Program.*, 13(4):277–298, October 2005.

[80] S. Ralf, B. Pouliquen, A. Widiger, C. Ignat, T. Erjavec, D. Tufis, and D. Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proc. of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, Genoa, Italy, May 2006.

[81] D. Romer, K. Kenski, K. Winneg, C. Adasiewicz, and K. H. Jamieson. *Capturing Campaign Dynamics 2000 & 2004: The National Annenberg Election Survey.* University of Pennsylvania Press, Philadelphia, 2006.

[82] W. Schäfer. Global history. In *Encyclopedia of Globalization, ed. by Roland Robertson and Jan Aart Scholte et al.*, volume 2, pages 516–521, New York and London.

[83] W. Schäfer. Lean globality studies. *Globality Studies Journal*, 7, May 2007.

[84] W. Schäfer. Personal communication, 2008.

[85] P. A. Schrodt. Automated Coding of International Event Data using Sparse Parsing Techniques. *Presented at the meeting of International Studies Association, Chicago, February*, 2001.

[86] S. Sekine, K. Sudo, and C. Nobata. Extended named entity hierarchy. In *Proc. of the Third International Conference on Language Resources and Evaluation*, 2002.

[87] N. Shadbolt, B. T. Lee, and W. Hall. The semantic web revisited. *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, 21(3):96–101, 2006.

[88] M. Slee, A. Agarwal, and M. Kwiatkowski. Thrift: Scalable cross-language services implementation. `http://developers.facebook.com/thrift/thrift-20070401.pdf`, 2007.

[89] S. B. Sorenson, J. G. Manz, and R. A. Berk. News media coverage and the epidemiology of homicide. *American Journal of Public Health*, 88(10):1510–1514, 1998.

[90] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. *Condor – A Distributed Job Scheduler.* MIT Press, October 2001.

[91] C. A. Taylor and S. B. Sorenson. The nature of newspaper coverage of homicide. *Inj Prev*, 8(2):121–127, 2002.

[92] C. Ward, M. Bautin, and S. Skiena. Identifying Differences in News Coverage Between Cultural/Ethnic Groups. Submitted to *ACM Journal on Computing and Cultural Heritage (JOCCH)*, 2009.

[93] F. Weigel, B. Panda, M. Riedewald, J. Gehrke, and M. Calimlim. Large-scale collaborative analysis and extraction of web data. *Proc. VLDB Endow.*, 1(2):1476–1479, 2008.

[94] T. White. *Hadoop: The Definitive Guide*. O'Reilly, 2009.

[95] J. Wiebe. Learning subjective adjectives from corpora. In *AAAI/IAAI*, pages 735–740, 2000.

[96] J. Wiebe and E. Riloff. Creating subjective and objective sentence classifiers from unannotated texts. In *Proc. of the International Conference on Intelligent Text Processing and Computational Linguistics*, volume 3406 of *Lecture Notes in Computer Science*, pages 475–486, Mexico City, MX, 2005. Springer-Verlag.

[97] T. Wilson, D. Pierce, and J. Wiebe. Identifying opinionated sentences. In *Proc. of the Meeting of Human Language Technologies-North American Chapter of the ACL (HLT-NAACL-2003) Companion Volume (software demonstration)*, 2005.

[98] J. Yao, G. Wu, J. Liu, and Y. Zheng. Using bilingual lexicon to judge sentiment orientation of chinese words. In *Proc. of the Sixth IEEE International Conference on Computer and Information Technology*. IEEE Computer Society, 2006.

[99] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *ICDM '03*, page 427, Washington, DC, USA, 2003. IEEE Computer Society.

[100] J. Yi and W. Niblack. Sentiment mining in webfountain. In *ICDE '05: Proc. of the 21st International Conference on Data Engineering (ICDE'05)*, pages 1073–1083, Washington, DC, USA, 2005. IEEE Computer Society.

[101] W. Zhang and S. Skiena. Trading Strategies To Exploit News Sentiment. Submitted to *the International Conference on Data Mining (ICDM)*, 2009.

[102] Zvents Inc. Hypertable. `http://hypertable.org/`.