Convex Approaches to Text Summarization

by

Brian Christopher Gawalt

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Laurent El Ghaoui, Chair
Professor Bin Yu
Professor Kevin Quinn

Fall 2012

Convex Approaches to Text Summarization

Abstract

Convex Approaches to Text Summarization

by

Brian Christopher Gawalt

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Laurent El Ghaoui, Chair

This dissertation presents techniques for the summarization and exploration of text documents. Many approaches taken towards analysis of news media can be analogized to well-defined, well-studied problems from statistical machine learning. The problem of feature selection, for classification and dimensionality reduction tasks, is formulated to help assist with these media analysis tasks. Taking advantage of ℓ_1 regularization, convex programs can be used to efficiently solve these feature selection problems efficiently. There is a demonstrated potential to conduct media analysis at a scale commensurate with the growing volume of data available to news consumers.

There is first a presentation of an example text mining over a vector space model. Given two news articles on a related theme, a series of additional articles are pulled from a large pool of candidates to help link these two input items. The novel algorithm used is based on finding the documents whose vector representations are nearest the convex combinations of the inputs. Comparisons to competing algorithms show performance matching a state-of-the-art method, at a lower computational complexity.

Design of a relational database for typical text mining tasks is discussed. The architecture trades off the organizational and data quality advantages of normalization versus the performance boosts from replicating entity attributes across tables. The vector space model of text is implemented explicitly as a three-column table.

The predictive framework, connecting news analysis tasks to feature selection and classification problems, is then explicitly explored. The validity of this analogy is tested with a particular task: given a query term and a corpus of news articles, provide a short list of word tokens which distinguish how this word appears within the corpus. Example summary lists were produced by five algorithms, and presented to volunteer readers. Evidence suggests that an implementation of ℓ_1-regularized logistic regression model, trained over the documents with labels indicating the presence or absence of the query word, selected word-features best summarizing the query.

To contend with tasks that do not lend themselves this a predictive framework, a sparse variant of latent semantic indexing is investigated. Sparse principal components were cal-

culated to define corpora built of two distinct styles of news. The results are used to both summarize with word lists, as before, and extract documents representative of their larger collection.

The work concludes with discussion of further study of the impact of the underlying vector space models of text on the resulting summaries, of real-world deployments of this kind of news analysis, and for distributed calculation of these results to support larger and larger corpora.

Contents

List of Figures

List of Tables

Acknowledgments

Laurent El Ghaoui's support and guidance were absolutely essential for this project, and I offer all my thanks to him for the opportunity, the advice, the expertise: the works.

The predictive framework for text analysis came from strong collaboration with Luke Miratrix and Kyle Jia, and the direction provided by Bin Yu led, I feel, to work we can all be proud of.

My thanks as well go to Youwei Zhang and Dafna Shahaf. It was with Youwei's assistance and acumen that the results of sparse principal component analysis over text are presented here; work with Dafna inspired and drove the nearest neighbor approach to document interpolation.

And Saheli Datta possessed an invaluable capability for identifying and developing promising connections to, and background material from, the wide world of media analysis.

Thank you, all.

Chapter 1

Introduction

The daily news, as it's been understood for most of its existence, has been of a substantial volume – substantial, but not unimaginable. A typical major media market as recently as the turn of the century could expect serious news coverage from two to three daily newspapers, mid-single digits of local television broadcasts, a few cable channels, and a dedicated AM or FM radio station. There was a lot of material, but not so much that a truly dedicated citizen couldn't consume it all in at a cost of, say, many hours of the day. Perhaps on the order of hundreds of news items would be a ball-park estimate of the overall volume.

The internet has changed this landscape considerably. The lower distribution costs of digital content has made it easier and more economically feasible for news providers to conveniently make their content available to wider and wider audiences. Modern wire-clipping and news monitoring services as as Mongoose Global Intelligence claim to discover "over 700,000 news articles per day, derived from over 40,000 news sources" online. [29]

And this greater supply of content is finding a growing base of consumption. The Pew Research Center's for the People and the Press conducted a 2008 survey to discover that for the first time, "more people [40%] say they rely mostly on the internet for news than cite newspapers [35%]." [55] The notion of what should be meant by "reading the news" has shifted, and it has shifted to a domain of staggeringly larger news volumes than has historically been the case. It's certainly no longer conceivable that any one citizen could process it all to reach a sort of view-in-full of the day's reporting.

There has been a commensurate jump in technologies engineered to deliver this content. Search engines have been specially designed to facilitate direct search for recent stories, such as the Google News service. Many sites offer "similar to this article" links, and the NYTimes.com can tailor this content similarity metric to particular users ("recommended for you"). In order to power these algorithms, there have been a collection of breakthroughs which all point to a specific development: to help us we're building computer programs which learn how to read the news for us. [6]

This dissertation is an exploration which takes these technologies as its starting point: presume something is learned by our text-mining procedures. If these systems can understand news well enough to deliver it to us, what possibilities are there for them to share

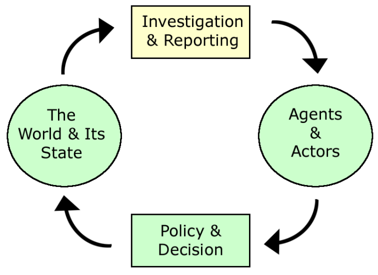


Figure 1.1: A simple schematic depicting the news and its place in the decision cycle.

what they've learned? Not just perform a task – classification, recommendation, clustering – but explain what in the text drives their decisions about the tasks. At document corpora scales of 700,000 articles per day, how should we modify these techniques to present useful and informative summaries?

1.1 Why the News Matters

The most important aspect of the news media, for the concerns of this work, is its ambition, capability, and success in informing a large share of society. Though their are many aspects of news – its entertainment value, its function as a income-generating business, the celebrity status of its more prominent contributors – this fundamental transmission of information, is taken as paramount, especially in the democratic context of much of modern life. H. J. Gans holds that "democracy [...] may belong directly or indirectly to its citizens, but [it] can be only truly meaningful if these citizens are informed." [21]

A schematic view of the media's role appears in Figure 1.1. A decision-making agent and the world it inhabits interact in a cycle. The state of the world is made known (possibly imperfectly) to the decision maker – which in the case representative democracy is the collective citizenry. The decisions and policies pursued in response to this information lead to a new world state, which again is investigated and (partially?) revealed to the decision maker, who can pursue new policies in response, ad infinitum.

The properties of this information channel are in this way are important to real-world outcomes. If the world can only be partially observed, due to budgets of time and attention, it matters what gets emphasized by the news. This has been studied in many contexts, a few of examples of which are below:

Geography and International Relations

In 2009, a survey was conducted to study the portrayal of the nation of Haiti in United States newspapers. [49] A "year's worth" of articles – 711 in total – were analyzed with respect to a predetermined set of keywords. The number of appearances of each of these keywords (terms including *violence, crisis, blood/bloody/bloodshed, slum*, etc.) were counted, and these terms were used as a guide to understand the diversity of articles as emerging from a few key frames (the "illicit drugs" frame, the "refugee problem" frame, ...).

The led the researcher to conclude "[t]he media representation of Haiti is perhaps one of the most devastating problems it faces today.[...] The place image the U.S. media put forth about this island nation fails to adequately acknowledge thefull extent of its present problems [...] inhibiting an understanding that will offer real solutions to its problems." It is argued that too-frequent use of stereotype and cliche in reporting on Haiti represents a bias in our feedback channel.

Psychology and Mental Illness

A panel of volunteers was convened to examine contrast between coverage of mental illness issues in articles from 1989 compared to those in 1999. These readers were instructed how to judge each of 300 articles along specific compositional aspects, including the specific mental disorders discussed, the "tone" (alarming? compassionate?) of the article, and overall thematics. [59]

These hand-coded scores on each dimension were used to argue that news coverage appeared to evolve in a direction of "fewer themes of dangerousness, and fewer articles with negative tone." They posit that "[i]t may well be that mental health advocacy and efforts to better inform reporters and editors about mental illness is paying off with more positive and fewer unambiguously negative stories."

However, while they believe that a *trend* towards what they consider a more honest and complete portrayal of mental illness is taking place, the actual existing *state* of coverage remained biased overall. Overrepresentation of scenarios wherein mental illness led to violent or criminal behavior "may undermine public recognition of the varieties of disorders, symptoms, and outcomes encompassed by the term mental illness."

The study's authors include a discussion of the limitations of their work. They anticipate that there may be weaknesses in their evidence arising from design in a few key areas. Their method of curating the source articles were based on keyword-based search over the LexisNexus news archives; their choice of keywords to isolate their 300 articles may have missed important examples. Similarly, they caution the news sources they chose to include may leave out important channels (for example, the study included no tabloids or small-town papers).

Both provisos appear to be an outgrowth of a fundamental budget: a limit on the attention of those coding the articles.

Immigration Policy

Researchers in the field of political science were interested in factors which induce slant in media coverage of an issue. They focused on California (English-language) newspaper coverage of the debate, policy, and developments on Mexican immigration as a specific case study. [9]

The researchers were able to amass 1,277 articles covering immigration between March 1, 2004 and March 1, 2005 found by searching the online archives provided by NewsBank, Inc.

These articles, some straight news pieces and others from editorial or opinion pages, were coded by four trained assistants as positively slanted towards increased immigration, presented the issue neutrally, or was slanted negatively. (They found substantial to almost-perfect agreement of the different coders' assessments of a held-out validation set.) These content-analysis scores were treated as dependent variables, influenced by the ownership structure and geographic location of the newspaper publisher. Papers published by corporate owners (as opposed to privately held papers), and papers published nearer the border, had higher probabilities of negatively portraying the issue in print.

1.2 How This Work Can Help

This dissertation is about turning an existing body of work in statistical machine learning towards the specific problems and demands faced by researchers such as those in the preceding examples. It is an investigation into how new tools for media analysis can be built to try and improve the evidential soundness of such studies.

Benefits of Software

Text analysis software holds out the promise of bolstering these examinations in a number of fronts.

Number of Articles Examined

A common thread in the above studies was a fixed budget of time and attention available to manually code the articles. This necessarily leads to some narrowing of the focus of the studies. The examination of immigration in California concludes with a recommendation to extend the study to other border states.

This is considerably more feasible if software could be used to code the articles – no need to compensate the four trained article readers for their time. The existing body of labels and articles could be leveraged as a training set for software to learn what article language is likely to signify a positive, negative, or neutral slant. (The high degree of agreement between the coders in the original experiment is a promising sign that the task can be automated.)

Breadth of Analysis

The software has the capacity to examine each document along many more facets than the time-constrained manual analysis can allow. In our examples of the studies of media portrayal of Haiti and mental illness, the manual coders had to be primed in advance of what kinds of language to look out for. This can be limiting – terms and themes may have been unanticipated by the researchers.

This work shows that software is certainly capable of accounting for several orders of magnitude more aspects of text than the cited projects above tracked. The study of Haiti was counting fewer than a dozen word families; this work demonstrates that software can keep track of many tens of thousands of distinct word tokens. Keeping track of everything is a one way to guard against bias in the prior assumptions when designing research methodology.

Reproducibility
The current analyses are based on manual coding, which requires a certain amount of faith. It's easy for researchers to disclose or disseminate their datasets, but the results were somewhat noisily derived from a staff of human readers. There's no guarantee a different set of readers would produce the same coding results as the original team. This fear is allayed by performing checks of inter-coder reliability as part the first experiment (ensuring that there's strong correlation between the results independently produced by coders on the same material), but the original conditions of the experiment can only be approximated.

If instead the evidence driving the conclusions of a research project are the result of software, it's now much easier for independent runs of the experiment to validate the original findings. The software can presumably be shared as easily as the data. This openness should lead to greater credibility in the field, allowing for greater impact.

Challenges of Software

Though there is much appeal in the above benefits, there are several obstacles that arise in designing software to serve media analysis researchers.

Representing the Data Decisions need to be made about how the computer is to interpret the documents it's provided. There are several competing models and techniques to be chosen from; for instance, the "bag-of-words" representation is popular, but has some widely known shortcomings with respect to matching human interpretation of a document (e.g., loss of information about word order). Alternatives, such as inferring a sequence of latent topics evolving as a hidden Markov model [26], may be more appropriate, but require greater computational power.

Validating Results Some algorithms lend themselves to clear, easily-calculated performance metric. A new sorting algorithm, for example, can be judged on the speed with which it sorts some benchmark set of data, compared to existing, competing algorithms. Clear cut performance metric: A shorter time is better. To gauge the performance of automated text analysis, however, is more complicated. If the software is meant to tag documents as positively or negatively slanted, at some point a human reader will have to judge whether the software's output is producing tags similar to a manual coder's.

Computational Complexity Text mining can generally be understood as largely flowing from a few staple tasks: document classification ("is this email unwanted spam or legitimate communication?") [40], document clustering ("is there a family of other news articles similar to what a website user is currently reading?") [54], machine translation ("what is the Portuguese translation of this English-language document?") [14], document summarization ("what few sentences/excerpts from this document best capture the overall message of the piece?") [25], real-valued regression predictions ("what can I predict about tomorrow's stock prices given the news available today?"), etc. These tasks are accomplished with algorithms of various degrees of sophistication.

For example, in the case of document classification, building a naive Bayes classifier over the words used can be accomplished considerably faster than training a support vector machine (SVM) over the same volume of data. However, the performance[1] of the SVM may far exceed that of the simpler naive Bayes model.

The designer of the text mining system would need to decide whether the additional time-costs of running the more sophisticated model would be worth the improved performance. In this particular case, the runtime cost ratio of complex-model-over-simple-model is super-linear: doubling the number of documents to be examined in building the models more than doubles the additional time required to opt for the SVM over the naive Bayes model.

Interpretable Models In order for the results of text mining approaches to be of best use in media studies, they must be easily understood. A random forest classifier [10] may yield fantastically low error rates in performing a classification task, such as distinguishing mental illness articles from different decades. However, such a high-performing model is likely to be opaque in its mechanisms. There'd be no clear explanation for what specifically about the documents suggests a difference between the decades, only a large and complicated model with many contingencies.

A Path Forward

The pros and cons above lead to a set of criteria for bringing text mining techniques to media studies domains.

The approaches tackled in this work will all draw on *convex optimization* techniques [7]. The procedure for fitting a text model to a document corpus can in many cases be posed as a

[1]In a classification setting such as this, "performance" almost always refers to the misclassification rate of the model. For example, when the model is presented with a new, valid email, how often is it incorrectly redirected to the spam folder? Or, if presented a novel spam email, how likely is it to erroneously direct it to the user's inbox? Some models may sometimes strictly dominate their competitors, with better performance along both dimensions, but more common is to find two models with comparative advantage in only one style of error rate. Good design methodology, for any task, will start with a metric of performance tailored to the task (classification? clustering?) as well as to the domain (how many spam emails should be sent to the inbox for every valid email lost to the spam folder for our particular user base?).

convex problem. This implies that the time cost to fit the model is bounded by a polynomial function of the "size" of the input (the number of documents, the degree of detail to which each document is examined/enumerated, etc.).

In order to adhere to constraints of model interpretability, the techniques used will tend towards *sparse* and *linear* models. Sparsity can ensure that the overall model can be succinctly described; only a few textual features of the tens-of-thousands of options available to the learning algorithm will remain in the final model fit. Linearity means that the contingencies and interactions between features will be limited; the effect any feature has on the model's judgements (e.g., the impact of a particular word's presence on the overall "spamminess" score of an observed email) won't be beholden to the appearances of any other features, eliminating the perplexity of feature-feature interaction.

Many of the top performing algorithms for traditional statistical machine learning tasks do not meet these standards. They're frequently designed with performance itself in mind, to act as standalone decision makers. For our purposes of media studies, we'll need to tweak their objectives in order to allow the learning frameworks to function instead as an evidence miner: not a tool just to perform classification or clustering but to also lend itself to a human-interpretable ruleset.

This means that in order to investigate whether these convex, sparse, linear approaches are in fact worthwhile towards media studies ends, new validation steps will need to be established. Human readers will need to have the media studies task explained to them, familiarized with the document corpus, then trusted to judge whether the outputs of the models fit over this corpus are useful to meeting the media studies task as compared to reasonable competing algorithms.

1.3 Sparse Modeling and Text Summarization

Suppose that units of text from a corpus can be represented in a vector space, with each element of the vector in some way encoding the prominence of a particular keyword in that document. (This vector space model of text will be discussed in depth in the coming chapters.) The geometry of models trained over these vectorized documents can help guide us towards design of software to summarize news documents.

Supervised Learning Models

In Figure 1.2, we have an example of this vector space representation for two keyword-dimensions. The document corpus is built of two classes of document. Each text unit is either a legitimate email, or a spam email the recipient would prefer blocked. A classifier can be trained over these example data. A possible result from, e.g., a support vector machine using a Gaussian kernel [33] is shown as the dotted line. It is a decision boundary learned by the computer, and documents whose vectorization appears below the line would be admitted as valid; those above, blocked as spam.

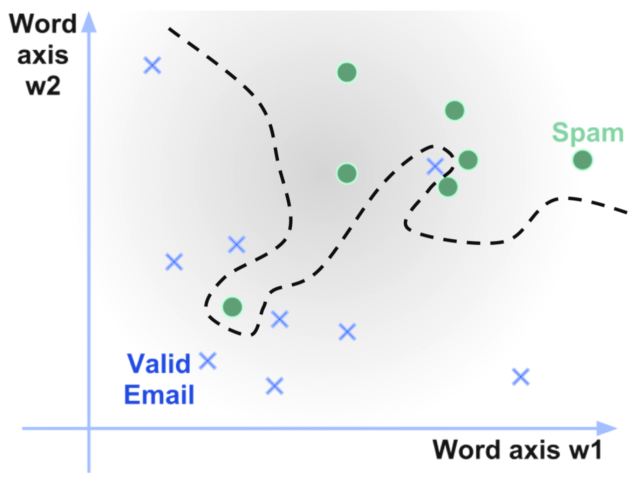


Figure 1.2: The geometry of an overfit classifier, represented by the dotted line, separating spam and valid email in a hypothetical corpus.

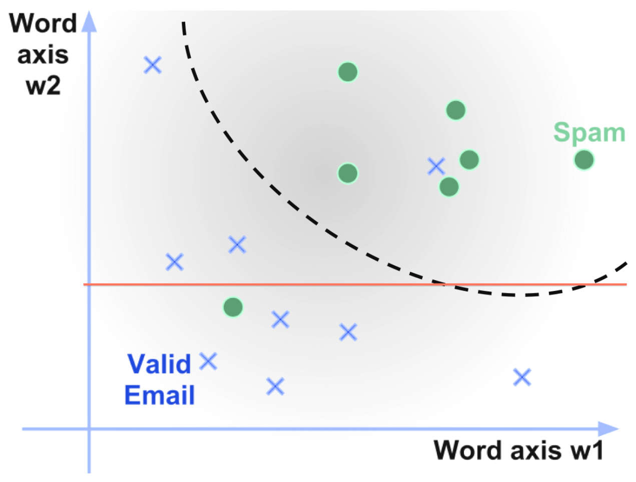


Figure 1.3: The geometry of a nonlinear classifier. At the w_2 value noted by the red line, either a sufficiently low or a sufficiently high value for w_1 implies a valid email.

This particular case was chosen to highlight the problems of overfitting. The classifier veers wildly to try and accommodate two wayward valid and spam data points. It's unlikely that this decision boundary would generalize well: the things the computer has learned are peculiar anomalies likely found just in this example set. Another nonlinear decision rule is presented in Figure 1.3.

Perhaps through the process of regularization, the model fit to the data now appears more reasonable. It seems like the traits of spam-vs.-valid email identified by this simpler model would extend beyond just this corpus. However, though the filter may do a fine job of actually performing the task of sorting email, it's unclear how well its workings could be

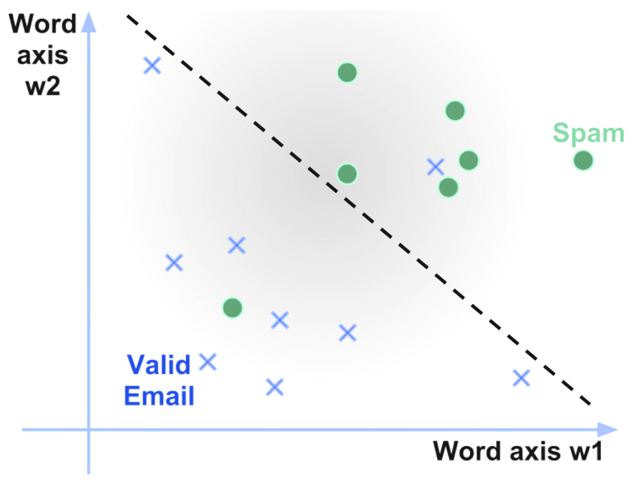


Figure 1.4: The geometry of a linear classifier.

made plain to outside onlookers. It's hard to make the model results interpretable.

Consider the simple question, "what effect does keyword w_1 have on the 'spamminess' of an email?" The answer is complicated and contingent: it depends. A red line has been aded to Figure 1.3 at for a particular value of w_2. Imagine a set of documents with that w_2 value, with varying w_1 values. An increase of w_1 – greater use of the w_1 keyword – would push some documents from the valid region into the spam region. But for others, an increase would move the document vector from the spam region back into the valid region again. These contingencies would potentially grow exponentially as more and more keyword dimensions were considered, making it difficult to understand the relationship between any one keyword's use and the nature of its parent document.

Addressing this, we might consider a strictly linear decision rule, as in Figure 1.4. We would typically expect that a strictly linear rule might underperform on a basis of classification error rate compared to nonlinear competitors. However, it's now much more interpretable. The machine can plainly state a coefficient multiplier for each keyword in the model. In the case of Figure 1.4, the coefficients are both positive: an increase in the number of appearances of either keyword suggests to the computer that the document is more likely to be spam. In this way, we can reflect on what about the keywords is responsible for this effect.

This approach increases interpretability, but there are still contingencies between the keyword dimensions. Correlation and collinearity between these features causes the specific numerical results produced in fitting the model to be dependent on each other [19]. The value of w_1's coefficient only is what it is because w_2's coefficient has the value it has.

There are many ways around this, including repeated subsampling/bootstrapping of the corpus data to produce an averaged model, but even with certainty in the rank order of the keyword coefficients, there's still the problem of model scale. If thousands of these coefficients are produced, that greatly eclipses the capability for a researcher to understand

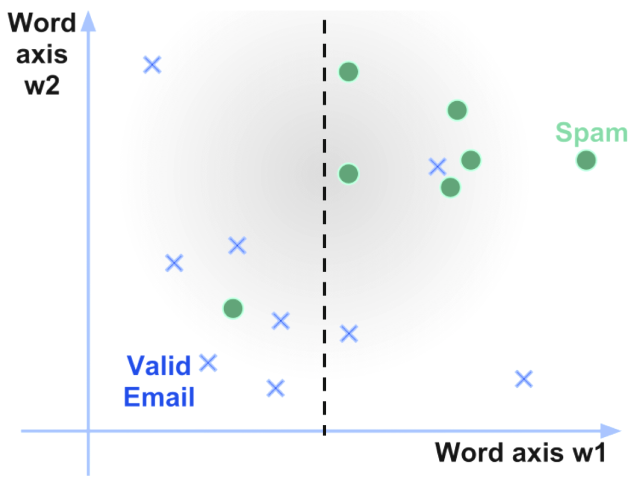


Figure 1.5: The geometry of a sparse, linear classifier.

the relationship between them all.

And so it is important to reintroduce the idea of sparsity to the modelling. Figure 1.5 demonstrates another linear decision rule. In this case, the value of w_2 is irrelevant to the spam-v.-valid email decision. The decision boundary is parallel to the w_2 axis. The coefficient the model associates with w_2 is zero. In a many-dimensional scenario, we can set a rule that a final model can only have at most k nonzero coefficients, such that a list of k keywords is quickly and easily human-interpretable.

We could satisfy this sparsity constraint, for example, by taking the k highest magnitude coefficients from our classifier in Figure 1.4. However, it is one thing to ask "if I predicted spamminess using a thousand keyword values, which k keywords would have the highest model coefficients?", than to ask, "if I could only use k keywords to predict spamminess, which k would I pick?" Convex algorithms for answering this second question – algorithms which can converge reasonably fast even at large scale – have been epitomized by the success of the LASSO, which updates the ordinary least squares objective with an ℓ_1 regularization penalty. [57]

To connect this back to the original discussion of media analysis, imagine now that the task is not separating spam from valid email, but 1989 articles about mental illnesss vs. those from 1999; separating articles about Haiti from those about a control-group of other nations; northern California articles about immigration from southern California articles. This procedure of discovering important keywords could bolster claims made.

This dissertation describes this approach the *predictive framework* for automated analysis. Keywords are extracted, and from them an argument about rhetoric or tone can be made from these keywords. No more need to establish the list of keywords by rhetorical value a priori.

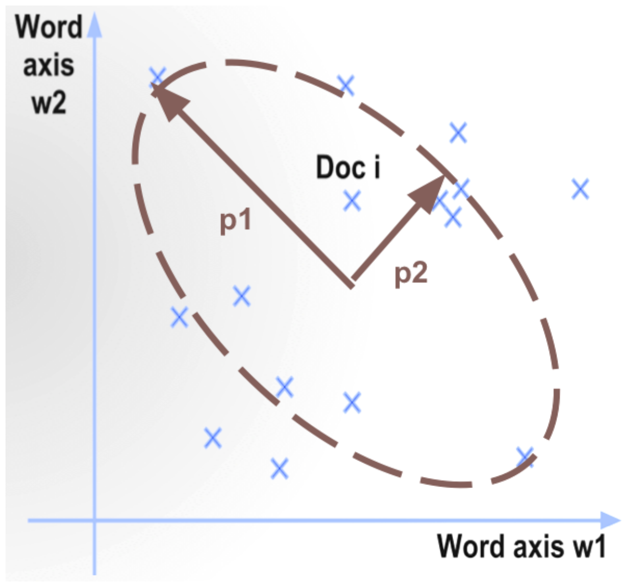


Figure 1.6: The geometry of principal component analysis over text, known also as latent semantic indexing.

Unsupervised Learning Models

Not all news analysis present an easy analog to this prediction task. There may be no comparative analysis required. Instead, it may be enough to discover and summarize the varieties of documents. Algorithms for this setting are long studied. Some of their most successful work have come from applying and extending principal component models to text, generally known as *topic modeling*. [32, 16, 28, 4]

Figure 1.6 provides a basic depiction of the model. Directions within the vector space are identified that capture a great deal of the variation in keyword values between the corpus documents. These directions can be represented as vectors within the keyword vector space, again assigning one coefficient to each keyword analyzed. As before, it is useful to produce a short set of coefficients, and inducing sparsity to the models with an ℓ_1 regularization can help efficiently calculate this set. [15] If there could only be k words used to describe the variation between documents in the corpus, which k would be most useful? The answer can lead a researcher to conclusions about the underlying dynamics of the news and its subject.

1.4 Organization of this Dissertation

The aim of the dissertation is to investigate the application of specific styles of statistical machine learning techniques to media studies tasks as exemplified above.

In Chapter 2, the development of a novel text mining algorithm is documented in order to demonstrate how machines can accomplish useful tasks by mining unstructured text.

The task to be automated asks that when provided two example input documents, the algorithm searches a database of candidate documents to find articles which serve to fill in the gaps between the inputs – a *document interpolation* engine. The new approach to the task uses a *nearest neighbor* approach, reducing the task essential to a single sparse matrix-vector multiplication. It is compared to a state of the art (nonconvex) integer programming approach as well as a technique known to be inferior.

Chapter 3 builds from the approach of the algorithm in Chapter 2 – learning text via a vector space model – and discusses how a relational database for text mining can be designed and implemented. Considerations of table normalization are balanced against the specific queries common in working with news data.

Chapter 4 then returns to the field of media studies. A particular task is described: summarize with a short list of phrases which a particular subject's appearances in a news corpus. The criteria of convexity, sparsity, and linearity are applied. The success of the convex algorithms as summarizers are validated by volunteer human readers. The effects of different vector space representations of the text are also tested and found to be important contributors to overall summarizer performance.

Chapter 5 presents a few experiments in sparse principal component analysis for text summarization, run over *New York Times* and Twitter data. The resulting summaries are presented as word lists, as well as articles the sparse model implies are representative of the corpus. Two competing methods are tested: using a standard term-document matrix as input, or using a centered variant.

Chapter 5 concludes the work, suggesting what future research would be of interest. Application domains are discussed given the advantages and disadvantages found in Chapter 4.

Chapter 2

A Text Mining Case Study: Document Interpolation

The explosion of data has a great potential to improve our quality of life in many ways. Unfortunately, without reasonable techniques of sorting through this data to find what is valuable, that potential goes unrealized. As more and more material is added to our stockpiles by the hour, we are quickly eclipsing the scale which we are capable of handling ourselves.

This chapter directly addresses this problem of data overload for a particular text mining case. It begins by assuming a user has a large database of potentially interesting documents at her disposal. In addition, she picks two designated articles, which share a common thread of some kind. This thread is of most interest to her – it's the pair of documents themselves which are her query. What documents can she find in this database which shed the most light on the connection between her pair of documents?

The framework of document interpolation has a variety of applications. For example,

1. A lawyer has a huge amount of evidence to pore through. He has two memos, written weeks apart, and needs to know which other discovery documents could elaborate on the process which led from the situation described in the first memo to the situation described in the second.

2. An academic researcher knows already of a highly-regarded paper, ten years old, but has just read a paper on the latest developments to grow out of this original line of research. What publications can best detail the evolution of the field?

3. A casual and occasional consumer of the news hears about a sudden, new development in the world, and remembers bookmarking a story about the same region months earlier. Which few other articles can best describe the progression from then to now?

All these cases have the same underlying goal: create a firmer and clearer understanding of the relationship between the two input documents. Herein, this process is called *document interpolation.* It finds intermediary documents to help fill in the gaps of understanding left between the two original articles. See Section 2.1 for more details.

The document interpolation problem is well studied. Work appearing in [52] has already tested the effectiveness of a number of different approaches to document interpolation. Several of these approaches are reviewed in Section 2.2.

This paper contributes a new algorithm for document interpolation. The approaches drawn from [52] are based on finding traversals over the directed graph of document-document similarities, leading from the older input document to the more recent. Consider a representation of documents in a *vector space model*, where the position of any particular document in this space is defined by its lexical content. One could then select the documents which are closest under this vector space to the set of convex combinations of the user's two input documents – closest to the literal mixtures of the two inputs. The computational efficiency of such an algorithm would be considerably faster than previous approaches. This is explained further in Section 2.3.

Outputs from these approaches have been generated for a corpus of news articles drawn over the last two years from the International section of the *New York Times*. The dataset is described more fully in Section 2.4. These results were then put in front of a panel of readers who graded their quality, as explained in Section 2.4. Section 2.4 analyzes the results of this survey.

Section 2.6 then argues that this nearest neighbor approach performs document interpolation quite well, especially when for an approach with its relative computational simplicity. Avenues for further research in document interpolation appear in Section 2.5. The nearest neighbor approach lends itself particularly well to many of these recommended avenues.

The main contributions are:

1. Proposing an elegant, efficient new approach to the task of document interpolation.
2. Conducting a user study over a real-life dataset.
3. Demonstrating results comparable to those of state-of-the-art approaches, at a fraction of the computational cost.

2.1 Document Interpolation

This section will formalize a definition and notation for the document interpolation task.

First, the user chooses two documents. Let *Document A* be the one published first (the older one) and *Document B* the other. Since document interpolation is designed to connect two related documents, let the pair of documents together be known as the *subject*. The user also specifies K, the number of intermediate documents he would like to receive as output.

The *document interpolation algorithm* or *approach* accepts these inputs, then consults the *database* (or *corpus*) of candidate documents. It selects K of these candidate documents to return to the user, ordered by their publication date. Call this ordered collection a *chain* of documents for this subject, leading from Document A to Document B.

A block diagram of this process is sketched in Figure 2.1.

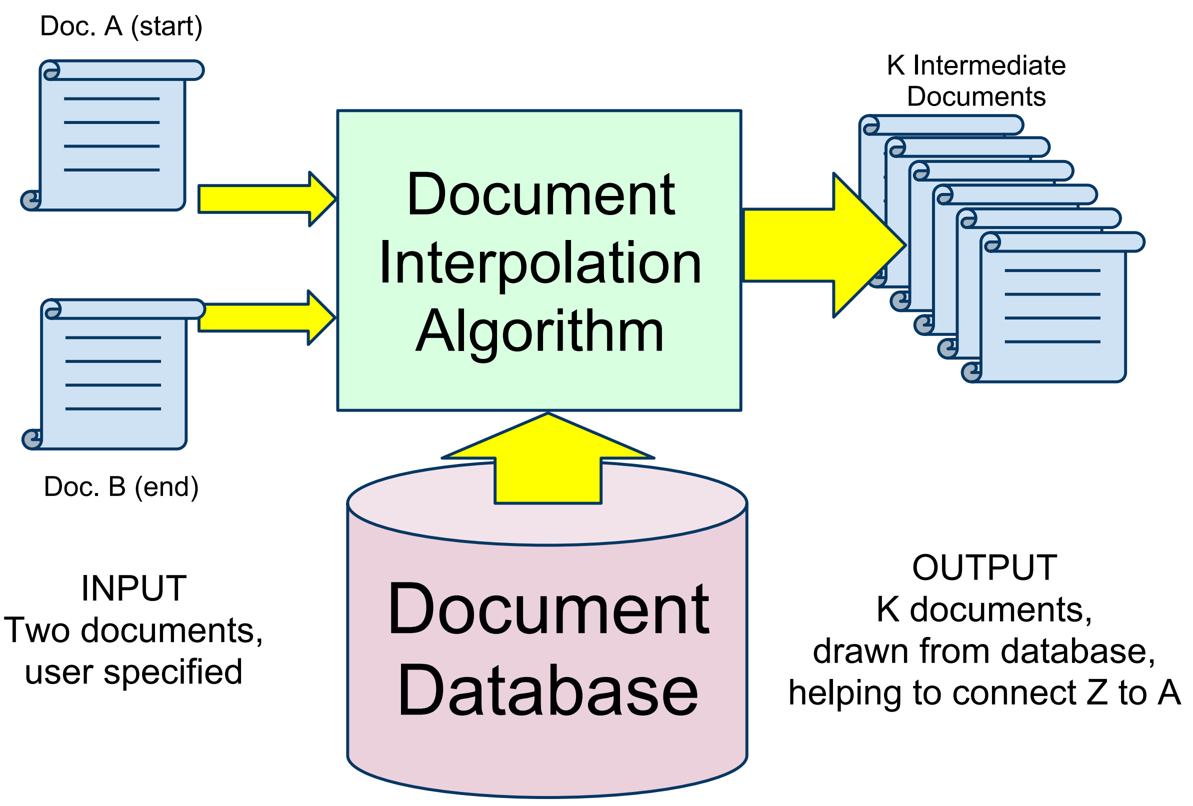


Figure 2.1: The document interpolation algorithm finds which of the database's documents serve as the best intermediaries linking the two input documents.

2.2 Related Approaches

The task of document interpolation is relatively new. Nevertheless, there has been much related work. This section outlines several of the main related approaches.

Shortest Path

The approach begins by constructing a graph over the corpus articles. Each article corresponds to a vertex; weighted, directed edges are added between every pair of articles. The weight is computed by cosine similarity, and the direction corresponds to chronological order. In order to speed up the computation, one can prune away the edges which represent the lower similarity values.

Once the graph has been built, efficient path finding algorithms can simply compute the shortest path from Document A's vertex to Document B's. Every vertex passed through on this traversal is added to the output chain. If the traversal is too short, the cut off for what

Table 2.1: A chain of four articles connecting two stories about the trapped Chilean miners, as generated by Shortest Path.

Headline (Date)	
[Doc. A]	"World Briefing — The Americas: Chile: President Seeks Foreign Help to Rescue 33 Trapped Miners" (Aug. 10, 2010)
2)	"As Europe Kicks Coal Habit, Hungarian Town Feels Pangs" (Sep. 16, 2010)
3)	"As Europe Kicks Coal, Hungarian Town Suffers" (Sep. 16, 2010)
4)	"Amid Tension, China Blocks Crucial Exports to Japan" (Sep. 23, 2010)
5)	"China Blocks Vital Exports to Japan" (Sep. 23, 2010)
[Doc. B]	"Chile Miners Honored by President in Capital" (Oct. 26, 2010)

edges should be pruned can be made more stringent. This can continue until few enough edges remain that the shortest path passes though K vertices.

Since each pair of consecutive documents is similar, it is anticipated a strong chain results. However, Shortest Path is a local method. Every consecutive pair of documents along the chain are related, but the chain does not necessarily display a global, coherent theme. It is quite possible that the content which makes the first and second documents in the chain similar is not the same content which makes the second and third similar – and so the first and third documents may seem to have little relationship to each other. Shortest-path chains may exhibit a sort of "stream-of-consciousness" behavior, cycling through many concepts and areas along the way from Document A to Document B.

Table 2.1 shows a chain connecting a news article, "President Seeks Foreign Help to Rescue 33 Trapped Miners" to "Chile Miners Honored by President in Capital", as generated by the Shortest Path method. The effect presents itself readily, as mining is linked to coal, coal linked to China and it's exports, and then back to the Chilean miners.

This example also highlights another pitfall of document interpolation. The chain produced by Shortest Path (Table 2.1) features two pairs of documents which largely duplicate each other. All approaches to document interpolation are going to be susceptible to this effect to some degree: a dirty database is going to lead to poor quality interpolations by the principle of "garbage in, garbage out." However, greedy algorithms like Shortest Path may be uniquely prone to this problem. Duplicate documents within a corpus are by nature highly similar and may always present as the most attractive next link to be added to the chain.

Other Local Approaches

Several other methods for connecting two fixed endpoints have been proposed in the literature. [34] formulate a new problem called *storytelling* as a generalization of redescription mining. The strength of a transition measured by its Jaccard coefficient (the ratio of the size of common elements to elements on either side of the redescription). Since this is a local measure, the output chains are of a similar nature to shortest-path.

The topic detection and tracking community has also conducted a lot of relevant research. For example, [44, 39] studied how to discover sub-clusters in a news event and structure them by their dependency, generating a graph structure. Again, since edges are computed locally, global coherence is less likely.

"Connect the Dots"

[52] proposed a different way to look at this problem. The authors formalized the characteristics of a good chain (*coherence*). The main problem of Shortest-Path seems to be its locality; each two consecutive documents are related, but the topic that they share may not be shared by any other document in the chain.

In order to avoid the stream-of-consciousness effect possible under Shortest Path, the algorithm selects a small set of words, common throughout the chain, which are to be taken into account when computing document similarity. Limiting the number of words enforces a global theme, as transitions are forced to re-use words. Therefore, it helps dampen the capacity to shift from topic to topic.

A linear program is used to minimize the number of words needed to assess similarity is minimized while maximizing the authors's formalized good chain characteristics over the output chain. The authors provide theoretical guarantees that the algorithm efficiently connects two fixed endpoints. Altogether, the procedure was named *"Connect the Dots"*.

This approach has been shown in [52] to outperform the other methods mentioned above. However, the computational complexity it requires is also great.

2.3 The Nearest Neighbor Approach

It is often useful, when performing a retrieval task, to consider documents as points in a vector space. If to the document interpolation task is framed in terms of a vector space model, it makes sense to consider the points immediately between Documents A and B. Nearest neighbor algorithms have proven workable and effective in many tasks, such as filtering [1] and ranking [22].

Let v_A be the vectorization of Document A and v_B the vectorization of Document B. For $\lambda \in [0, 1]$, we define

$$v_\lambda = \lambda v_A + (1 - \lambda) v_B \tag{2.1}$$

The set $\{v_\lambda \mid \forall\, \lambda \in [0,1]\}$, what this paper calls the *mixture set*, contains points which are all a lexical mixture of the contents of the two user supplied documents. When λ is close to 1, the point v_λ corresponds to a hypothetical document almost identical in content to Document A. Similarly, if λ is near 0, v_λ corresponds to a document with strong resemblance to Document B. Points in between represent a literal compromise – if each dimension of the vector space reflects a certain word token use rate, the points in mixture set reflect word use that is a weighted average the word use rates in each of the input documents.

For each document i in the database, we can calculate the Euclidian distance between document i's vectorization (v_i) and the mixture set:

$$\begin{aligned} d_i &:= \min_{\lambda\in[0,1]} ||v_i - v_\lambda||_2 && (2.2)\\ &:= \min_{\lambda\in[0,1]} ||v_i - \lambda v_A - (1-\lambda)v_B||_2 && (2.3)\end{aligned}$$

The value of λ corresponding to the point in the mixture set closest to v_i can be found in closed form as the answer to a one-dimensional constrained least squares problem:

$$\lambda_i^* := \arg\min_{\lambda\in[0,1]} ||(v_i - v_B) - (v_A - v_B)\lambda||_2 \quad (2.4)$$

$$= \begin{cases} 1, & \frac{(v_A-v_B)^T(v_i-v_B)}{||v_A-v_B||_2^2} > 1 \\ \frac{(v_A-v_B)^T(v_i-v_B)}{||v_A-v_B||_2^2}, & \frac{(v_A-v_B)^T(v_i-v_B)}{||v_A-v_B||_2^2} \in [0,1] \\ 0, & \frac{(v_A-v_B)^T(v_i-v_B)}{||v_A-v_B||_2^2} < 0 \end{cases} \quad (2.5)$$

We can then substitute back in to find the distance d_i:

$$d_i = ||v_i - v_{\lambda_i^*}||_2 \quad (2.6)$$

Figure 2.2 illustrates this process for a two-dimensional case.

The nearest neighbor approach to document interpolation finds and returns the K documents closest to the mixture set, ordered by publication date.

The calculation of these d_i can be made quite quickly, especially when considering the typical sparsity patterns of document vectors. It requires only multiplication of a sparse matrix (the concatenated v_i's) by a sparse vector ($v_A - v_B$), followed by a few inner product calculations. This is in contrast to the complexity of graph-based approaches – Connect the Dots, for instance, requires solving a linear program with constraints of size comparable to Nearest Neighbor's multiplying matrix.

Documents with lower distances can be thought of as purer intermediates. For instance, when document i uses terms not mentioned in either Docs. A or B, it can only serve to increase its distance from the mixture set. This penalizes the introduction of new concepts along the chain, similar to the way influence is limited to as few words as possible in the Connect the Dots approach.

We repeated the nearest-neighbor approach to document interpolatoin using several different vectorizations, including bag-of-words, normalizing the bag-of-words representation by

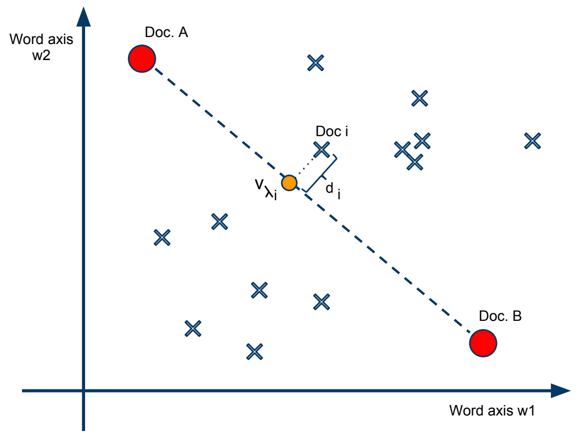


Figure 2.2: A 2-dimensional sketch of the nearest-neighbor projection process. X's represent database documents, the large circles represent user input documents, and the dashed line between the input documents represents the mixture set. Documents closest to the mixture set are chosen as the interpolating intermediates.

article length, and tf-idf. Looking at some of the chains which resulted, we decided that the highest quality interpolations were achieved using a vectorization which used bag-of-words normalized by corpus word count. Element j of v_i, where j ran from 1 to the number of distinct words found in the corpus, was set to be:

$$v_{ij} = \frac{\text{no. of times word } j \text{ appears in document } i}{\text{no. of times word } j \text{ appears in corpus}} \tag{2.7}$$

The Porter stemming technique [48] was applied as a pre-processing step to try and increase robustness to noise in word use. By replacing words in their text with their stem forms, e.g., ensuring "construct", "constructs", "constructed" and "constructing" are all replaced with "construct", the ideas and concepts motivating word use are that much less likely to spread their expression across multiple word dimensions.

2.4 Evaluation

This paper seeks to test the Nearest Neighbor approach as a document interpolation algorithm. We selected a set of subjects (Doc. A/Doc. B pairs), a corpus, and a chain-length value of $K = 4$ in order to compare the quality of the chains produced by Shortest Path, Connect the Dots, and Nearest Neighbor.

Dataset

To implement these methods, we assembled a corpus of 18,641 articles drawn from the International section of the *New York Times*, ranging in publication date from Dec 2, 2008, to October 31, 2010. They were scraped daily from links in their web publication's RSS feed[1].

Most are written in a typical "inverted pyramid" news-article style, relaying details of a particular event. About 2,750 articles are shorter "World Briefing" pieces. These are around one hundred words in length, compared to a typical article length of two-hundred to four-hundred words. Additionally, 175 are "News Analysis" pieces that try to tie together the events originally reported in the other stories, being roughly 800 to 1,000 words in length.

This corpus was selected because of the material's relative accessibility. Many of the applications listed in the introduction are easier to evaluate with expert knowledge. News articles, however, are constructed for readability by as broad a population as possible. Chain quality was to be evaluated by non-expert users, and non-experts would find a news corpus most agreeable.

Within this corpus, five pairs of documents were selected for interpolation. We refer to these separate input pairs with shorthand reference to their subjects, *gitmo*, *chile*, *mumbai*, *israel*, and *greece*. We list the details of these pairs in Table 2.2.

As an example of the kinds of interpolations produced, Table 2.3 showcases the output of each approach on the *mumbai* pair of inputs. Nearest Neighbor articles tend to focus on the larger atmosphere of terrorism of which the particular attack referenced in the input articles are a component. Connect the Dots selected a chain that focuses much more closely on the specific attack. Shortest Path produced a chain more concerned with India and Pakistan's diplomatic relations. These results are distinct from each other, but note that Shortest Path and Nearest Neighbor each selected "India Vows No Retaliation," from December 11, 2008.

These test cases were chosen to try and cover many of the possible kinds of input document pairs. We varied the duration between each article's publication date: the *gitmo* pair cover a span of 19 months, while *chile* took place in a much more temporally localized 3 month period. The *israel* pair uses a shorter "World Briefing" story as an input, providing less content from which to learn. *greece* begins with an article focused specifically on the Greek debt crisis, then tries to connect that to the more general situation of debt as considered across all of Europe in its Document B. A good document interpolation algorithm should be able to perform well on each of these.

Survey

In order to assess the quality of these approaches to document interpolation, we assembled a group of 36 people to read and evaluate the story chains. Bringing them together in one room, each sitting at his or her own PC terminal, the respondents were given a brief introduction to the document interpolation task and the dataset. The story chains were

[1]The URL is `feed://feeds.nytimes.com/nyt/rss/World`

Table 2.2: The five pairs of documents interpolated in this study.

Subject	Document Headlines (Date)
gitmo	*Doc. A:* "Obama Seeks Halt to Guantanamo Trials" (Jan. 21, 2009) *Doc. B:* "Guantanamo Detainee Pleads Guilty in Terror Case" (Jul. 8, 2010)
chile	*Doc. A:* "World Briefing — The Americas: Chile: President Seeks Foreign Help to Rescue 33 Trapped Miners" (Aug. 10, 2010) *Doc. B:* "Chile Miners Honored by President in Capital" (Oct. 26, 2010)
mumbai	*Doc. A:* "India Says All Mumbai Attackers Came by Ship" (Dec. 12, 2008) *Doc. A:* "Pakistani Man Convicted in 2008 Mumbai Attacks" (May 3, 2010)
israel	*Doc. A:* "World Briefing — Middle East: Israel: Palestinian Militants Fire Rockets From Gaza" (Dec. 18, 2008) *Doc. B:* "New Approach Sought as Pressure Grows to End Gaza Embargo" (June 10, 2010)
greece	*Doc. A:* "Greece Struggles to Stay Afloat as Debts Pile On" (Dec. 11, 2009) *Doc. B:* "Europe Debates How to Avoid Another Debt Crisis" (Sep. 21, 2010)

presented on a website one a time, immediately after which they were asked to score the chain's quality on a scale of 1 (poor) to 7 (excellent) via in-browser radio buttons.

Respondents were not primed by the authors as to what constituted quality. They were asked only to imagine that a document interpolation tool had produced the chains they were about to read, and then how well they imagined the result could be of use. Respondents were invited to write specific comments about each chain to shed light on their thought process. A sample of the comments left on chains over the *mumbai* subject are listed in Table 2.6.

Respondents were given an hour to read through and respond to the survey in this way. Each respondent's survey used a random ordering of the 15 chains, though not all were adjudicated whether due to the time constraint or technical difficulty. In total, 324 scores were registered, distributed over the subject-approach chains as shown in Table 2.4.[2]

[2]With the corpus fixed, as well as K, the easiest way to index specific chains of articles going forward is through the subject and approach used to generated them. With 3 approaches and 5 stories to test, that leaves 15 chains total under consideration.

Table 2.3: The four stories chosen by each method to interpolate the *mumbai* pair

1) "India Says All Mumbai Attackers Came by Ship" (Dec. 12, 2008) [Doc. A]

Nearest Neighbor	
2)	"Indian Police Disclose More Suicide Attackers" (Dec. 9, 2008)
3)	"Pakistan Tries to Curb Militant Group" (Dec. 10, 2008)
4)	"India Vows No Retaliation" (Dec. 11, 2008)
5)	"Terror Ties Run Deep in Pakistan, Mumbai Case Shows" (July 27, 2009)
Connect the Dots	
2)	"A Solemn Israel Buries Dead From Mumbai Attack" (Dec. 2, 2008)
3)	"Mumbai Attacks Were Partly Planned In Pakistan, Official Says" (Feb. 12, 2009)
4)	"In Court, Mumbai Gunman Says He Is From Pakistan" (Mar. 23, 2009)
5)	"Pakistan to Charge 7 in Mumbai Attacks" (Sep. 19, 2009)
Shortest Path	
2)	"India Vows No Retaliation" (Dec. 11, 2008)
3)	"News Analysis: India Presses Pakistan on Terrorism but Finds Its Own Options Limited" (Dec. 12, 2008)
4)	"Balancing Act for India as Talks With Pakistan Resume" (Feb. 24, 2010)
5)	"India and Pakistan Resume Talks" (May 3, 2010)

6) "Pakistani Man Convicted in 2008 Mumbai Attacks" (May 3, 2010) [Doc. B]

Table 2.4: Number of survey responses registered for each chain.

	chile	*greece*	*mumbai*	*israel*	*gitmo*
S.P.	18	20	20	19	15
C.t.D.	22	18	27	23	26
N.N.	16	20	23	22	20

Table 2.5: Average score performance each approach achieved on each story, along with the standard deviations calculated across these averages.

	chile	*greece*	*mumbai*	*israel*	*gitmo*	St. Dev.
S.P.	4.78	4.85	4.95	**5.57**	**5.4**	**0.35**
C.t.D.	**5.86**	4.83	**5.93**	5.52	4.77	0.55
N.N.	5.56	**4.95**	5.85	5.55	4.7	0.47

*Best performance in each column highlighted in **bold**.*

Results

Figure 2.3 depicts the histogram of quality scores given to interpolations generated by the Shortest Path approach, aggregated over all subjects. Similarly, Figures 2.4 and 2.5 show the quality scores for Connect the Dots and Nearest Neighbor approaches, respectively. We see that Shortest Path chains were judged to be of low quality (1, 2, or 3) considerably more frequently than Connect the Dots or Nearest Neighbors.

These results conform to the pattern established in [52]. This work also found that Shortest Path and other greedy approaches to the problem underperformed compared to Connect the Dots.

What is of most interest is that quality score distributions between Nearest Neighbor greatly resemble the distribution for Connect the Dots. In fact, Nearest Neighbor's mean score slightly tops the mean score for Connect the Dots.

Nearest Neighbor was able to match the quality of Connect the Dots, at a greatly reduced computational expense.

Table 2.5 enumerates the average quality score each approach produced on each subject, and documents the standard deviation of these sets of five averages. The standard deviation of Nearest Neighbor's mean chain performance was lower than that of Connect the Dot. This suggests Nearest Neighbor is at least as robust and able to handle a diversity of subjects as the state-of-the-art.

Nearest Neighbor performed comparably on all subjects. This is a promising facet of Nearest Neighbor's results, as it suggests it is robust to many styles of input.

The full heat map of scores is presented in Figure 2.6. Typically, the majority of responses for a chain tended to fall in either one or two bins – survey respondents tended to more or less agree on chain quality. This is especially true for Nearest Neighbor over *israel* and *gitmo*. In general, the heat map repeats the earlier pattern: Nearest Neighbor tends to inspire the same vote of quality as Connect the Dots. Note how the similarity between the two vote patterns seems to hold on every subject. Nearest Neighbor is considerably closer in quality to Connect the Dots than it is to Shortest Path.

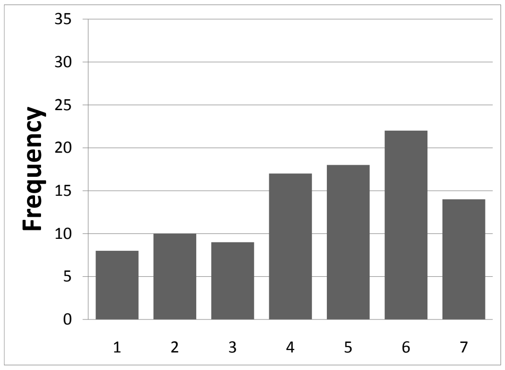


Figure 2.3: Histogram of quality scores for all interpolations developed using the Shortest Path approach. The mean score was 4.52.

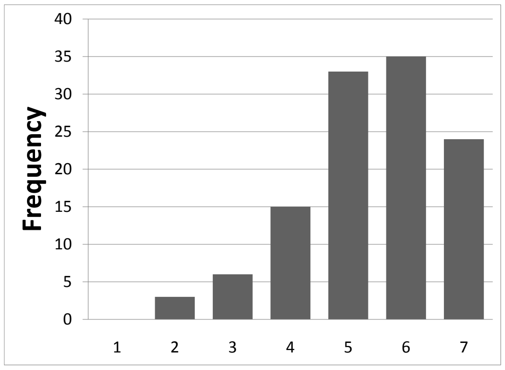


Figure 2.4: Histogram of quality scores for all interpolations developed using the Connect the Dots approach. The mean score was 5.42.

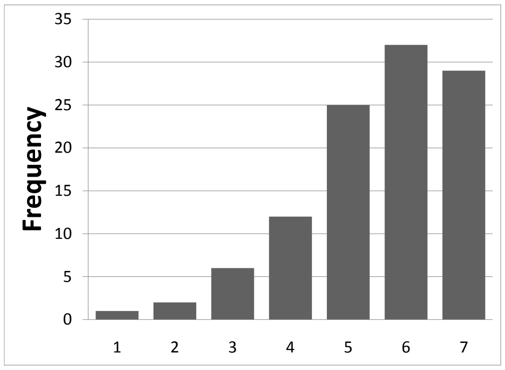


Figure 2.5: Histogram of quality scores for all interpolations developed using the Nearest Neighbor approach. The mean score was 5.52.

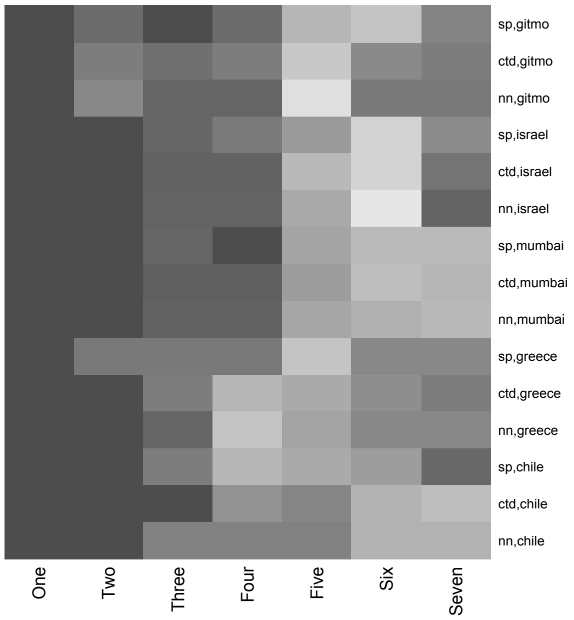


Figure 2.6: Heat map of quality scores distributions by chain. Lighter shades in a box indicate more survey responses to that row's chain registered that column's quality score.

Table 2.6: Some of the scores and comments left for the chains created over the *mumbai* subject.

Approach	Score	Comment
N.N.	5	"not well distributed along the time line, give me the feeling that some information may be lost during 2009-2010"
	6	"These six articles display a progression of the Mumbai attacks over time. Each article provides new information about the same storyline, which is enjoyable to read."
	7	"All are related and follow a clear progression."
C.t.D.	5	"Second story *['A Solemn Israel...']* does not really apply"
	6	"Article 2 does not belong- although it gives this chain a humanistic touch, it is largely irrelevant for those who want information about the chain of events of what happened in Mumbai."
	7	"well distributed along the time line, and presents a specific story."
S.P.	3	"the middle part is not quite leading to the end"
	5	"Articles 4 and 5 *['Balancing Act...' and 'India and Pakistan Resume...']* are not exactly related to the rest of the chain."
	5	"While chronologically sound, I feel like the chain could have done a better job focusing on a specific topic within this news item.[...]"

2.5 Future Work

In the future, we plan to further explore the document interpolation framework, especially in the following areas:

Part-of-speech tagging The vectorization used in this paper involved unsophistication parsing: one dimension per distinct character string. Results may be improved by using a vector space which considers the word and it's sense in the sentence. The scalability of the Nearest Neighbor approach makes growing the dimensionality, by splitting word tokens like "walk" or "talk" into both their noun and verb senses, a reasonable pre-processing step.

Additionally, it would be interesting to discover if limiting document content considerations to only a few parts of speech produce better results, all else equal. These intuitions about chain coherence, for instance, might suggest hypothesizing that only using the noun phrases from each document could improve chain quality.

Topic modeling Topic models, best exemplified by the latent Dirichlet allocation [4], have shown that text documents can be dealt with more effectively by inferring a mix of topics each document is expressing with its word use rates. Interpolation quality may be improved by using topic-model vectorization in place of this paper's bag-of-words approach.

Sensitivity to useless documents If a document interpolation algorithm produces a high-quality chain given a certain database, then we should expect it to return that same high-quality chain if it were given that same database, plus another collection of documents known to be irrelevant to the queried subject. In the case of this paper's subject, *israel*, we would want to see the same chains produced even if the database also contained two years of sports section articles. If we can confidently describe a set of documents as noise, the choice of chain documents should be insensitive to their inclusion or exclusion[3]. What are the relative sensitivities of Nearest Neighbor and the graph-based approaches?

2.6 Conclusions

As the quantity of data, especially text data, continues to grow, the document interpolation task will prove to be a useful mechanism for extracting value from these databases. This paper has introduced a new approach to document interpolation, the Nearest Neighbor approach.

By identifying candidate documents close to the set of mixtures of the two query documents, the Nearest Neighbor approach is an effective document interpolation algorithm. The quality of chains assembled by the Nearest Neighbor in this paper's experiments were found to match the quality of chains assembled by state-of-the-art graph-based techniques.

This capacity for quality comes in addition to the comparative ease of computation for Nearest Neighbor. Nearest Neighbor requires only a few sparse matrix multiplies, compared to repeated solutions of finding a shortest path, or solving a large-scale linear program. This simplicity can enable implementation of the document interpolation task for situations where speed or scale constraints make other approaches infeasible.

[3]Without this insensitivity, the task becomes complicated with a new question: how broad or narrow a collection of documents is called for before we can be confident in the quality of the document interpolation?

Chapter 3

Database Design for Text Mining

In this chapter, I will discuss the design of a database which facilitates learning and summarization over news data. A chief design constraint is the ability to conduct the same summarization task many times over different subsets of the data. For instance, finding key summarization terms for articles mentioning Canada could be repeated on articles falling in rolling one-year windows, and the changes in summarization term sets over time could be used as evidence arguing for a shift in the nation's role and perception on the world stage. To do that effectively, the database should allow for quick identification of

In the text mining tasks this dissertation discusses, there is always a benefit from analyzing a greater number of documents. The more training examples one can provide to the algorithm, the better the results will be. And as discussed in the introduction, a major motivation of this work is that the volume of text data worth analyzing is increasing in time.

To see why increased capacity to data is a net plus, imagine a case where we have an opportunity to grow the dataset from an existing body of K to documents to some $K + N$. Having access to more data can only help – because if the extra N seem to be having a deleterious effect, there's always the option of just ignoring these additional N after the fact. Or if it's the case that the algorithm can quickly deal with K documents, but due to superlinear runtime scaling chokes on $K + N$, having access to the larger number of documents still allows us to find K from this across this new superset. It's likely that lousy examples from the original sample can be one-for-one replaced with better or more informative candidates from the superset.

It's in our interest, then, to efficiently store and retrieve a great number of news articles.

3.1 Database Use Cases

Databases involve four basic actions. Three of these – insert, update, and delete – are considered state actions, changing what data is stored in the tables; the fourth, access, involves retrieving data from the tables. Design of a database should consider which actions will predominate its usage.

Insertions New data needs to be added to the database's tables. For news data, we can take trend data discussed earlier [29] to set an upper bound of one million new article insertions per data. Of course, the case studies examined focused only on specific subjects within the news media; it's possible that only a few articles per day would be harvested and inserted. The dataset used in the text mining work of Chapter 2 were taken from the *New York Times* International section, which produced about 25 to 30 articles on a typical day.

Updates It may be that data already hosted in the database needs altering. I'd expect this action to be rare. The documents being studied are presumably the news articles which made it to final print. Their pertinent attributes are fixed. It may be that certain aspects of *the software* change, such as an update to the word tokenizer, but realizing these changes would be a matter of essentially re-inserting the entire dataset under the new parsing rules anyways.

Deletions These actions, the removal of an article from the dataset, are also likely to be of lesser frequency, for reasons similar to the update action. It could be that some class of articles initially presumed interesting to the researcher might turn out irrelevant (e.g., the researcher has scraped the whole *San Francisco Chronicle* newspaper, but later decides there was no value to including sports section articles in the mining task). But presuming sufficient metadata has been collected, there's always instead the option of simply filtering these known bad classes out when preparing the corpus for mining.

Selections These actions – retrieving stored article data from the tables – could be quite common. Imagine a researcher looking for a difference in voice or tone on a certain topic between two sections of the paper, say news and opinion. The experiment to find this difference is conducted over each monthly subset of news data, to see if the effect changes over time. For a corpus containing years of articles, that's a considerable number of selection queries requested.

3.2 Corpus Representation

The heart of these convex approaches to text mining will depend, as we saw in Chapter 2, on the vector space representation of the news articles: breaking the articles into appropriate document units, and representing the number of times each word appears in each document unit.

Document Units

Let's stipulate in this chapter that the databases core text entity, or *document unit* is a paragraph and its constituent words. Paragraphs are convenient measures of topic. Its rare for an author to change the subject without also breaking into a new paragraph. If our summarizers are learning relationships between words based on cooccurrence patterns across the document units, it could be helpful to ensure as well as possible there's a tight semantic link in the words simultaneous appearances.

Each paragraph comes from an article, published on a particular date from a particular author under a particular headline in a particular section of a particular news source, scraped off the internet at a particular (and unique) URL. All of these attributes – the *metadata* which annotates the text dataset – represent the kind of information a media studies researcher might require when using software to gather evidence.

Using paragraphs as document units in this chapter's design study is also useful because of this shared aspect of the metadata. Paragraphs from the same article all need to share the same metadata. We can recurse this design if we wanted to incorporate a more complicated chained structure (e.g., publishing houses print books which contain chapters which are built from paragraphs, and we need metadata about each level; the age and office location of the publisher, the year of publication for the book, the chapter title, etc.), the same relational database design principles laid out here connecting articles to paragraphs could be applied at each link (chapter to book, book to publisher).

Word Tokens

Define a *word token* as a distinct sequence of alphabetical characters. Let's use this term as distinct from *word*. The sentence `''The dog ate the dog food.''` contains six *words*, but only four *word tokens*. The number of distinct word tokens found in a corpus tends to grow linearly with the number of document units in a corpus.

Figure 3.1 demonstrates two corpora, grown over time, and the number of distinct word tokens found. Document units were added one by one to a corpus, and the number of distinct tokens found in that corpus was recorded. The two document sources depicted are articles drawn from the *New York Times* International News section, and postings to the social media network Twitter.

These word tokens can have their own associated metadata. For example, some words can be designated uninteresting a priori: *stop words*, such as the, in, no, of, etc. Databases designed for mining tasks should set aside room to flag these tokens as nonpertinent to the mining tasks.

3.3 Relational Database Design

For the purposes of this project, the corpus and its matrix representation will be housed in a relational database. [13] A series of tables is defined, with each row or record of that table bearing a series of columns or attributes. The attributes will be synthesized to produce a table structure that allows for efficient mining.

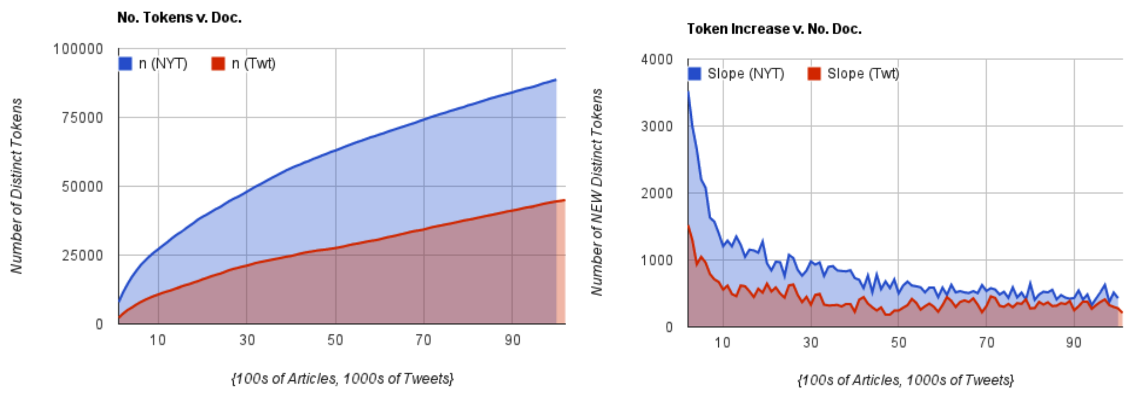


Figure 3.1: At left: Total number of word tokens against growth in number of document units. After an initial "burn-in" period, the rate of new token discovery per marginal is approximately linear. At right: the stabilization of new token discovery at a constant rate for each corpus. (N.B. The horizontal axes are scaled differently for each corpus.)

An entity-relationship diagram depicting table design for a news mining platform appears in Figure 3.2. It describes the relations between five tables:[1]

Article A single article's metadata is recorded: the article's source/publication name, URL, date of publication, news section, author, headline text, and full-body text. Each article comes with it's own unique, numerical identifier, labelled `ArtID`. This `ArtID` undergirds a one-to-many relationship between records in the `ARTICLE` table and those in both the `ARTPAR` and `HEADWORD` tables. (Paragraphs and headlines both have only one article from which they're drawn; however, articles can have many paragraphs, and headlines can contain many word tokens.)

ArtPar This table encodes information about a single paragraph. Each paragraph gets a unique numerical identifier, `ParID`, the paragraph's own text, as well as the article ID of its parent article (a many-to-one link back to the `ARTICLE` table). `ParID`

Word This table describes all the word tokens used parsed from across the corpus – any token found in any article. Each record in the table has the word token's own text and a field for its stop code, indicating whether the word is a stop word. Again, a unique, numerical identifier in `WordID`, which has a many-to-many relationship with

[1]Earlier, I described that this template could be easily modified to describe a corpus with even larger degrees of hierarchy in metadata. For example, suppose we included the highest level document unit to be the entire section of a newspaper on a given day. Imagine inserting another table at the top of the diagram, named `SECTION`, with fields for section name, publication, and date; perhaps the page editor's name. We'd now need a `SECTIONART` table (containing all the metadata currently used in the `ARTICLE TABLE`, except `Date`, if we're normalizing). There would be a one-to-many relationship linked by the section identifier and the rows in the `SECTIONART` table.

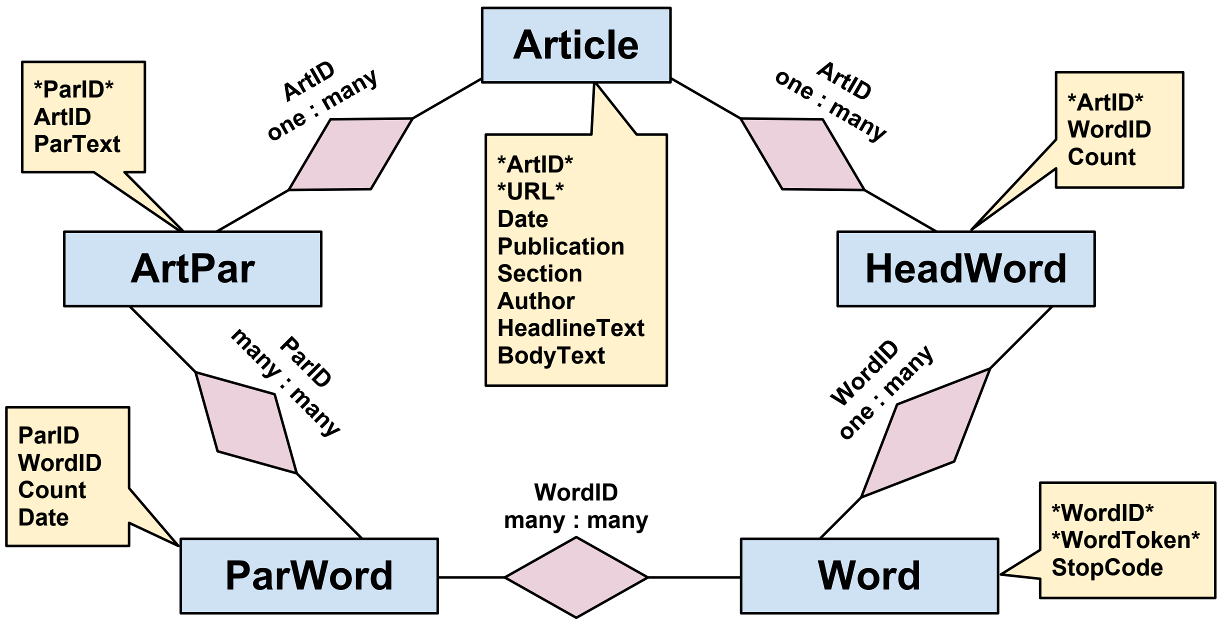


Figure 3.2: An Entity-Relationship model for designing a mineable news database. Rectangles represent relational database tables, annotated with rectangular callouts listing that table's columns. Asterisks around column names indicate that the column should be a unique identifier for the table. Relational links are denoted with diamonds, labelled with the attribute linking information between the tables.

both the `HEADWORD` and the `PARWORD` tables (as each word token can appear in multiple headlines or paragraphs, and each headline and paragraph can contain multiple word tokens).

ParWord This table encodes the appearances of word tokens in each paragraph document unit. The `Count` column counts the number of times the word identified by `WordID` appears in the document unit identified by `ParID`. There is also a `Date` column, more about which below. There are many-to-many relationships between this table and `ARTPAR` (linked by `ParID`) and `WORD` (linked by `WordID`).

HeadWord Much like `PARWORD`, this table encodes the number of times each article's headlines contains each word token.

A guiding principle of relational database design is *normalization*. A normalized database keeps a single source of truth for each attribute; no attribute of an entity is recorded in two separate tables. This can help with consistency: there's no risk that by oversight an update to an attribute is performed in one table, but not the other, creating a conflict between the two sources of truth.

This design deviates from this ideal. In both `ARTICLE` and `PARWORD`, there are attributes for the date of publication. I justify this by making a domain-specific case: news is intrinsically a time-sensitive affair. Quick filters by publication date are likely to be common use cases. Additionally, I would make an appeal that this data is quite likely to be written once. This at least limits the downside risk of inconsistent messiness in the data.

3.4 The Salton Matrix

Of these database tables, it's hard to overemphasize how crucial `PARWORD` is. This representation – a document ID, a token ID, and a count – is essentially defining a sparse matrix – a row ID, a column ID, and a non-zero value for the matrix value at that row and column. This data structure is referred to as the *Salton matrix* of a text corpus in this dissertation (though it is also frequently referred to as the corpora's term-document matrix).

The expense of holding this structure in our database is proportional to the number of non-zero elements in the Salton matrix. Figure 3.3 depicts the growth of this expense as a function of number of document units for two corpora, a selection of *New York Times* articles (note: full articles, not paragraphs), and a set of tweet messages from the social networking site Twitter. There is a tendency for the number of non-zeros to grow strictly linearly with the number of document units. (Figure 3.3 also depicts the rapidly decaying density of this matrix, to get a sense of the gains from storing the information in a sparse format.)

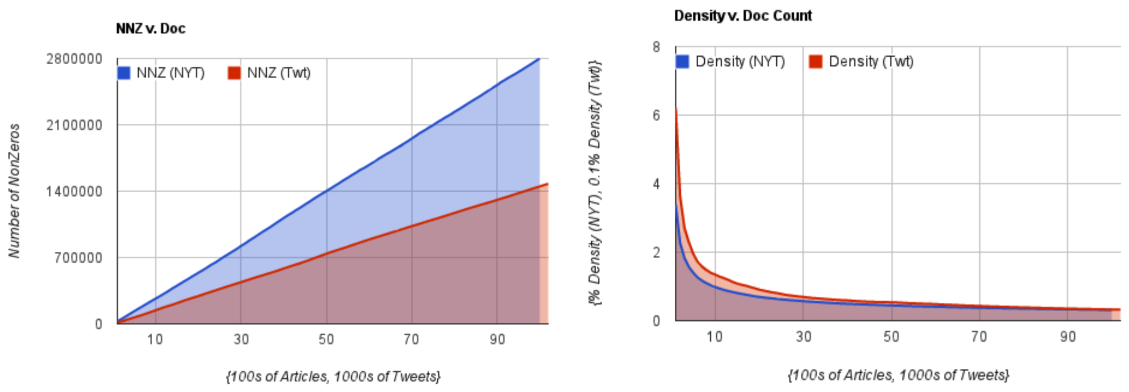


Figure 3.3: At left: Growth in number of non-zero Salton matrix entries for two text datasets. At right: Rapid decay in overall Salton matrix density (number of non-zero entries divided by matrix size). (N.B. horizontal axes differ in scale by a factor of ten between each data set; additionally, vertical scale differs by factor of ten in density figure at right.)

Viewed in this light, the difference between the two corpora in Figure 3.1 is instructive. Across the first 10,000 *NYTimes* articles examined, there are 88,691 distinct word tokens.

In the first 10,000 tweets, there are only 10,719 distinct tokens; after 100,000 there are only 44,496.

The number of columns in the *NYTimes* Salton matrix greatly exceeds the number of rows; the reverse is true of the Twitter Salton matrix. The proportion of columns to rows for these two matrices, as a function of the number of document units analyzed is in Figure 3.4. These matrix proportions are important to our larger aim of learning about voice, tone, emphasis, or bias, as reflected by the word tokens, through those tokens' patterns in the documents. The rank of the *NYTimes* matrix is limited by the number of documents; there's a definite risk now of overfitting if we try to train a model over the columns with strictly fewer rows.

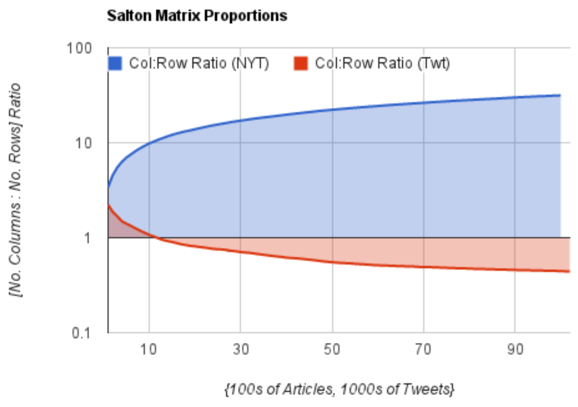


Figure 3.4: In a log scale, the ratio of number of columns to number of rows for Salton matrices built from *NYTimes* and Twitter documents. A system over the word-columns governed by the *NYTMatrix* is always underdetermined; with Twitter, it is potentially overdetermined as number of tweets grows. (N.B. horizontal axes differ in scale by a factor of ten between each data set.)

The particular reasons for the differences between Twitter and its predecessor media have been well studied. [35] As one possible explanation, Twitter is famous for the retweet behavior; multiple tweets all quoting the same exact phrase. Duplicate document units lead to fewer novel tokens generated per document unit. Of course, verbatim quoting means rows of the Salton matrix are duplicates, and each duplicate reduces the overall linear independence of the rows. It's quite possible neither the "skinny" Twitter matrix nor the "fat" *NYTimes* matrix guarantees a non-underdetermined system.

This tension between a desire to say something about token use across the data, running into the fact that (in many, if not most, corpora) there can never be enough observations to keep pace with the rapid growth in token-space dimensionality will be a strong motivator in the choice of models this dissertation pursues in exploring summarization tasks for media studies.

Chapter 4

Feature Selection, Sparse Classification, and Summarization of News Topics

Progress in technology allows the public daily access to an unprecedented volume of news, coming from various sources. However , given this volume, significance and meaning of these reports can be hard to assess and decipher. Can machine learning help? In turn, can this field of news media analysis modify and inform progress in machine learning?

Text classification is a vibrant field [51, 53] that has been extensively used for news data. It can help to categorize documents [51, 23, 37], to provide sentiment analysis [38] and opinion mining on articles [46], and to predict future market trends. It appears that relatively little has been done, until now, to connect the extensive technological progress in the area of text classification to issues that are of concern to the social scientist, such as how the media does or does not influence our perception of the world.

There is a long academic tradition in humanities and social sciences scholarship of extracting quantitative data from manual classification methods and then qualitatively assessing the significance of usage patterns within the resulting categories and category schemes [3, 27]. Many of these media studies usually make use of simple word frequency or co-occurence counts. Recent work such as [24], in which the authors seek to quantify media slant, calls into play more elaborate statistical methods. There does seem to be an opportunity for a strong interplay between text classification methods, as presented in the machine learning literature, and qualitative approaches to discourse analysis. A call for such an effort (within the context of literary discourse) has been made by Moretti in [42].

Studies of word usage in the media studies have often focused on the portrayal of international issues in domestic or international media [45]. The need for such analyses is indeed acute for news related to foreign policy. Jervis in 1976 [30] opened foreign policy analysis to include the role of perceptions and misperceptions in international politics, especially in terms of security. In addition, many media studies have clearly identified the mainstream media as the primary source of information about world affairs. Entman's book, *Projections* *of Power* [18], argues that the media not only frames the agenda, but is part and parcel of the exercise of power, specifically where the predominant frame is transmitted from the executive to other elites, then to the media and finally to the public. Intended or not, Thompson [56] states that "media presentation is a crucial determinant of the public perception of international politics." Therefore any systematic and easy-to-use tool that can help in understanding *how* issues, events and policies are presented in the media, has a fundamental role to play in social sciences. One such tool could provide analysis of word associations in a given corpus, allowing one to better understand how concepts are linked to shape our perceptions.

4.1 Word Imaging

This chapter's focus is the potential use of text classification in understanding the "image" of a certain word in a given news stories corpora. The term *image* simply refers to the words in the dictionary that, in the corpus and within the time window under consideration, are associated in some statistical sense with the word under study (referred to as the *query* word). Such images could provide a great service to researchers in media studies.

Sparse classification and word imaging

There are many ways to define association, from co-occurence within (say) a paragraph to more sophisticated methods. A first challenge in the imaging problem is thus to choose a meaningful method to specify and quantify association between terms. This work's methodology rests on sparse classification, which roughly refers to a set of algorithms that perform efficient feature (term) selection while maintaining satisfactorily minimal classification error rates.

Based on a given query term, the documents units (these could be the articles themselves, or their paragraphs, or headlines) from the corpus of news items are separated into two classes: those that contain the term and those that do not. A sparse classifier can then be trained over this labelled data, which results in an assignment of weights to each term in the dictionary. The term *sparse* here means that most of the classifier weights are zero. Thus, the few words that are most relevant in predicting the presence or absence of the queried word in any document are singled out by training a particular classification method. In this sense, the resulting short list is a distinctive image of the query via this particular classification method. One might expect that the higher the classification performance, the more accurate the image.

In contrast with many previous applications of text classification to news data, this work is not explicitly concerned about sentiment analysis or categorization. The principal focus here is on the identification of a short list of words that has reasonably good classification performance. The meaning and import of the words themselves can be identified after the fact by a domain expert. To be easy for this expert to use, this list should be extremely short with respect to the size of the entire dictionary. Hence, the feature selection method should result in extremely sparse classifiers.

The algorithms tested here all make use of logistic regression as a classification model. There are many other models relevant to this task, ranging from random forests [10] to Support Vector Machines [31]. This choice is motivated by the interpretability of logistic regression and the corresponding probabilistic model, as well as the computational efficiency of the underlying maximum-likelihood problem [23]. The algorithms selected differ only by the feature selection method, that is, the part responsible for the sparsity of the classifier. Four of the algorithms rely on independent feature models that have proven popular in text classification; the last algorithm, sparse logistic regression, uses a penalized version of logistic regression [23], to perform feature selection without any independence assumptions.

An example query: Microsoft

To illustrate this approach, let us show how it can be used to visualize the history of a term's image. Applying this sparse classification algorithm to the study of the term “microsoft” in all the New York Times headlines between January 1981 and December 2006 (so each headline is a document unit). The sparse logistic regression “BBR” algorithm [23] was applied on a sliding window of one year at yearly increments. (For more details of this algorithm's application, see section 4.3).

One year's worth of headlines were examined at a time and computed a sparse logistic regression classifier to classify the headlines containing the term “microsoft” from those not containing that term, repeating this for year after year. Thus, for each year, there now exists a short list of terms, and associated weights, that are predictors of the appearance of the query term in any headline. This results in a matrix of weights, each column corresponding to a year, and each row to a word that ever appeared in a “microsoft” image. If the rows of the matrix are arranged so that words are listed in order of appearance, the matrix heatmap reveals a staircase pattern. Tracing any row reveals the corresponding term blinking in and out over time as a salient descriptor of “microsoft”.

One visualization of the matrix of weights is shown in figure 4.1. In the figure, each little rectangle indicates the presence of the word as an important feature for that year. The darkness of the rectangle indicates its relative weight to the other words selected for that year. The vertical axis corresponds to the terms in the resulting collection of lists, shown by order of appearance. Thus, the plot shows a staircase pattern, where terms with large total sum of absolute weights over time are highlighted in red. Table 4.1 provides a list of the top 30 such words.

The list in Table 4.1 appears to provide an accurate summary of Microsoft, with the top prize going to “software” (the long-term focus of the company) and “xbox” (its most recent best-selling product). Figure 4.1 goes further, in providing a story of the evolution of the company that is consistent with common knowledge, and indeed the Wikipedia entry on Microsoft. The initial terms refer to a high-growth corporation (with terms like “company”, “net”, “profit”, “doubled”). The list of words then visits terms related to products, from

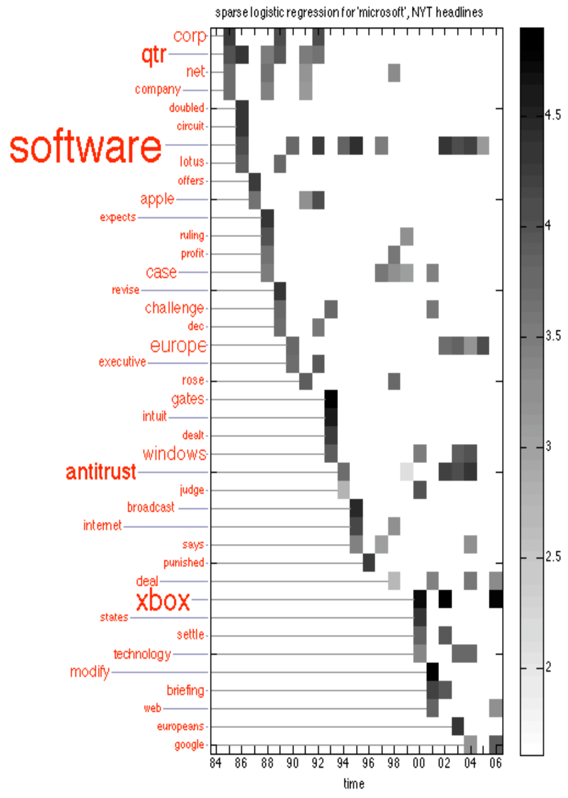


Figure 4.1: Study of the term "microsoft" The New York Times headlines: matrix of logistic regression weights with corresponding high-weight terms highlighted.

"lotus" to "windows" to "xbox". Another important topic involves legal terms ("case", "judge", "settle") , with a reference to the famous anti-trust case in Europe. More recently, the terms reflect the growing importance of the Internet ("web") and media ("broadcast"). Throughout, the names of important related companies are mentioned: "lotus", "apple", "google" ("intuit" is also part of the list, but is not shown).

Reading the plot vertically gives the main topics for a particular year. For example, the year 2002-2003 has "anti-trust", "europe", and "software". The plot also allows one to pinpoint terms that frequently recur in the news over a long stretch of time (e.g., "software").

This particular example is encouraging in that sparse classification algorithms could be useful in providing a quantitative, consistent, common-sense summary of a widely cited topic. Obviously the question arises as to which classification algorithm should be used.

Contributions

In this chapter, the aim is to evaluate several algorithms that are potentially well suited to a near-real-time imaging task. Two styles of algorithm evaluation are employed: predictive

1	software	11	apple	21	executive
2	xbox	12	challenge	22	rose
3	qtr	13	modify	23	internet
4	antitrust	14	briefing	24	broadcast
5	europe	15	technology	25	ruling
6	corp	16	deal	26	says
7	windows	17	settle	27	expects
8	case	18	intuit	28	europeans
9	gates	19	lotus	29	dec
10	net	20	company	30	profit

Table 4.1: Most important words found by sparse logistic regression analysis of the term "microsoft" in The New York Times headlines, ranked by sum of absolute regression coefficients over time.

classification performance and is human evaluation based on a rigorous protocol of comparison. The main finding is that, even though the predictive performances of these algorithms are similar, human-based evaluation seems to favor sparse logistic regression. This implies that predictive performance alone is not enough to choose algorithms for the word imaging task, and further research is needed to better understand "what humans want" in terms of word images and to see whether sparse logistic regression can serve as an automated method for the word imaging task more generally.

This chapter is organized as follows. Section 4.2 presents details on pre-processing the text data. Section 4.3 provides an overview of the algorithms used. Section 4.4 describes the statistical metrics of the different algorithms' predictive performance. Section 4.5 is devoted to a human evaluation protocol. In both these sections, results are provided pertaining to the image of various oft-cited countries in the international section of The New York Times between 2008 and 2009.

4.2 Preparing the Data

Corpus

The data used in this chapter are a series of news articles from the International section of the New York Times, as syndicated on their RSS feed. Publication dates run from December 15, 2008, to October 18, 2009. The corpus was stripped of capitalization, reverting all characters to lower case. Punctuation was also scrubbed, with some marks (periods, question marks, commas) replaced with whitespace and other marks (hyphens, apostrophes) simply deleted and ajoining their neighboring characters. For example, the plaintext "Arab-American" becomes "arabamerican". From here, the text was vectorized. In these analyses, only single-word terms (no digrams or trigrams) were considered, and each paragraph was treated as a single document unit.

Bag of words

To extract the statistical structure of a corpus, the news data must first be somehow enumerated. This *bag-of-word* approach represents each paragraph of the corpus by a vector whose dimensionality includes one element for each distinct word. The j-th element for vector i is then set to the number of times word j appears in the i-th document. The news corpus is comprised of $79,494$ distinct words (the term dictionary) used across 109,686 paragraphs, leading to a data matrix $X \in \mathbb{R}^{m\times n}$, with $m = 109,686$ rows (number of data points) and $n = 79,838$ columns (dimension of feature space, that is, dictionary size). As most paragraphs have a word count under sixty, less than 0.05 percent of elements of this matrix are nonzero.

Document labels

The imaging task seeks to distinguish between paragraphs containing a given query word and those that did not. Let $q \in \{1,\dots,n\}$ be the index of the query word. Each paragraph i was labeled as a positive example if the query word appeared in it at least once, and negative otherwise:

$$y_i = \begin{cases} +1 \text{ if } X_{iq} > 0, \\ -1 \text{ else,} \end{cases} \quad i = 1,\dots,m.$$

The number of positive examples for the several experiments (each based on querying a different country) varied between fifty and two-thousand. In the experiments, the respective q-th column of the data matrix, which corresponded to the query word, was removed.

Stop words

In many cases, words may be deemed intrinsically uninteresting. Terms such as “in”, “with”, “and”, “but”, “the”, etc., carry little-to-no descriptive weight. It is known a priori that they have no place in a word image. Dropping them from the matrix before processing the data costs little, helps decrease runtime, and ensures more descriptive images. However, this process is not riskless: while the word“said” is typically used as a neutral linking verb with little connotative value, its proper noun heteronym “Said” (as in Edward Said, the literary theorist), is indeed informative. These experiments used a limited list of 300 words, available in Appendix B. These 300 words were removed from the dataset.

Stemming

Many distinct words share meaning: verbs can describe an identical action but vary by tense, a noun can be another noun’s plural, etc. As with stop words, it can be helpful to reduce the size of the overall dictionary by mapping words with shared roots into a common feature. This process of stemming is common in many applications. The procedures used in these experiments avoided stemming the dataset.

The same risk of lost information as in stop words above applies even more severely here. For example, a stemmer might be expected to consider “iraq” and “iraqis” equivalent, but the connotation of an image that focuses on a nation’s individual citizens as opposed to one focused on the nation as a whole is an important distinction for the purposes of media analysis. Though a stop word list is sufficiently short so as to be manually tuned, the space of possible stemmings is too large to allow individually flagging each as acceptable or unacceptable. Note that some authors recommend stemming in text classification [53], while others warn of a potential loss of predictive performance [51].

After these steps, the dataset is ready for statistical analysis. The algorithms used are all based on a first step where feature selection is performed. Then a standard logistic regression algorithm, described next, is applied to assign weights to the selected features.

4.3 Feature Selection Algorithms

The algorithms tested fall under two approaches. Four algorithms use independent feature models in their selection: co-occurence count, Binomial Normal Separation (BNS), Delta-TFIDF (D-TFIDF), and a threshold statistic to control the false discovery rate (FDR). A fifth feature selection method (L1LR) does not rely on independence assumptions, and instead uses a penalized variant of logistic regression to select features.

After reducing the number of features to a few tens, a standard logistic regression algorithm is used to assign classifier weights to each selected feature.

Logistic regression

Logistic regression is a classical classification method based on a generalized linear model [23]. Take data points $x_i \in \mathbb{R}^n$ and associated labels $y_i \in \{-1, 1\}$, $i = 1, \ldots, n$. The logistic regression model is based on the following expression for the conditional probabilities:

$$P(y_i = 1 | x_i) = \frac{1}{1 + \exp(-x_i^T \beta - \gamma)},$$

where $\beta \in \mathbb{R}^n$ is the vector of regression coefficients in the model, also referred to as “weights”, and $\gamma \in \mathbb{R}$ is an intercept. An estimate of the vector β can be obtained by solving the corresponding maximum (log-)likelihood problem, which can be expressed as

$$(\hat{\beta}, \hat{\gamma}) = \arg\min_{\beta,\gamma} \ \mathrm{L}(\beta, \gamma), \tag{4.1}$$

where

$$\mathrm{L}(\beta, \gamma) := -\sum_{i=1}^{m} \ \log\Big(1 + \exp(-y_i(x_i^T \beta + \gamma))\Big)$$

is the log-likelihood function. Logistic regression has been widely used in data mining and text classification [23].

Feature selection methods

The words have been indexed by set $J = \{1, 2, ..., n\}$ and documents (paragraphs) by the set $I = \{1, 2, ..., m\}$. These documents have been perfectly partitioned into two subsets, $I^+ = \{i \in I | y_i = +1\}$, of cardinality $\#I^+ = m^+$, and $I^- = \{i \in I | y_i = -1\}$, of cardinality $\#I^+ = m^-$. Each method seeks a subset $K \subseteq J$ with cardinality as close as possible to a target k. The algorithms are summarized below.

Co-occurence For each word $j \in J$, compute $c_j^+ = \sum_{i \in I^+} X_{ij}$. Let $\bar{c}$ be the $k+1$th highest value found in vector c^+. By this method, build $K = \{j \in J \ : \ c_j^+ > \bar{c}\}$. This method selects the k non-stop-words which most frequently appear in paragraphs in which the query also appears.

Delta TF-IDF (D-TFIDF) The Delta TF-IDF method (D-TFIDF for short) [38] uses a variant of the well-known Term Frequency, Inverse Document Frequency (TF-IDF) scores for text documents.

Having established c^+ above, calculate for each word $j \in J$ the count of positive and negative paragraphs with that word. Namely, let $d_j^+ := \#\{i \in I^+ : X_{ij} > 0\}$ and $d_j^- := \#\{i \in I^- : X_{ij} > 0\}$ Note that d_j^+/m^+ is the percent of times word j appears at least once in the positive examples. Similarly for d_j^-.

Use these values to produce

$$\delta_j = c_j^+ \log(\frac{m^+}{d_j^+}\frac{d_j^-}{m^-}), \ \ j = 1, \ldots, n.$$

Let $\bar{\delta}$ be the $(k+1)$-th highest value found among the magnitude of these values $|\delta|$. Build $K = \{j \in J \ : \ |\delta_j| > \bar{\delta}\}$. This method selects by a combination of seeking words that appear commonly alongside the query term, with added sophistication to penalize those words that co-occur too often in the positive examples (perhaps an indication of what is effectively, for this query word, a stop word) and rewarding those that appear rarely in the negative example paragraphs.

Bi-normal separation (BNS) The bi-normal separation (BNS for short) method has been proposed in [20].

Take vectors d^+ and d^- as above. For each word $j \in J$, compute $b_j = \Phi^{-1}(\frac{d_j^+}{m^+}) - \Phi^{-1}(\frac{d_j^-}{m^-})$, where Φ^{-1} is the inverse of the cumulative distribution function of the standard normal distribution. Let $\bar{b}$ be the $(k+1)$-th highest value found among the magnitude of these values $|b_j|$. Build $K = \{j \in J \ : \ |b_j| > \bar{b}\}$. This method selects words with divergence between rates of appearance in each paragraph class, bringing into consideration an underlying normal model for appearance rate that allows for greater distinction between tail and modal behavior – extremely rare or common words are gauged by a different standard than words that appear about as often as not.

False discovery rate (FDR) Multiple testing problems involve identifying multiple significant hypotheses simultaneously. Rank p-values are used to get the significant hypothesis. Some error rate, such as false discovery rate or family wise error rate, can be controlled by selecting a threshold of the ranked p-values, based on which the hypothesis are rejected or not [2, 47]. This method has been adapted to the feature selection problem, based on log-likelihood ratios. The decreasing order of the log-likelihood ratios is the same as the increasing order of p-values, if the p-values are calculated based on the log-likelihood ratios which approxiamtely follow χ^2 distributions. That is, the bigger log-likelihood ratio, the smaller the p-value and the more likely the null hypotheses is rejected.

Take d^+ and d^- as above. For each word $j \in J$, compute:

$$\begin{aligned} f_j &= d_j^+ \log(\frac{d_j^-}{m^-}) + (m^+ - d_j^+)\log(1 - \frac{d_j^+}{m^+}) + \\ & \quad d_j^- \log(\frac{d_j^-}{m^-}) + (m^- - d_j^-)\log(1 - \frac{d_j^-}{m^-}) - \\ & \quad (d_j^+ + d_j^-)\log(\frac{d_j^+ + d_j^-}{m^+ + m^-}) - \\ & \quad (m^+ - d_j^+ + m^- - d_j^-)\log(1 - \frac{d_j^+ + d_j^-}{m^+ + m^-}). \end{aligned}$$

Let $\bar{f}$ be the $(k+1)$-th highest value found among the magnitude of these values $|f|$. Build $K = \{j \in J \ : \ |f_j| > \bar{f}\}$.

Breaking Ties

In certain cases, there may be a tie for the k-th highest score under any particular process discussed above. These incidences can be reduced in frequency by repeatedly executing the process on a subset of the training data, allowing scores to accumulate for words with each iteration. Specifically, each process was repeated for 10 iterations, holding out a randomly selected 10% of the training data each time. This has the added benefit of promoting stability in the word choices: the effect of outlier paragraphs on word scores can be muted in this way. Should any ties remain in the cumulative scores, the wordlist was padded out to a length of k by randomly drawing from the words tied for k-th place.

Assumptions of feature independence

The appeal of the above methods lies in their scalability. The order of computational complexity is linear in the number of distinct words and documents. However, this benefit results from an underlying assumption of independence between the appearances of words across documents. Below, this project investigates applications of more computationally intensive methods which may take advantage of acknowledging correlation and collinearity between words and tests whether this approach leads to better word images (and if the image improvement is worth the increased computation costs).

Sparse logistic regression (L1LR)

Sparse logistic regression (L1LR for short) [23] allows for simultaneously performing feature selection and model fitting, via the introduction of an l_1-norm penalty to the maximum-likelihood problem:

$$(\hat{\beta}(\lambda), \hat{\gamma}(\lambda)) := \arg\min_{\beta,\gamma} \; \mathrm{L}(\beta, \gamma) + \lambda \sum_j |\beta_j|, \tag{4.2}$$

where $\lambda > 0$ is a penalty parameter. The presence of the l_1-norm encourages many components of the estimated vector β to be zero, an effect that becomes more pronounced as $\lambda \to +\infty$. A line search of λ can produce a run of the algorithm which obtains a specific cardinality k of nonzero elements in vector $\hat{\beta}(\lambda)$. An implementation of efficient BBR software described in [23] was employed in these experiments. According to these experiments, for a data set with $109,686$ observations and $79,494$ predictors, BBR software can produce a solution for the optimization problem (4.2) in 15 seconds for a given λ.

In this approach, L1LR was usedpurely as a feature selection mechanism. As with the four previous approaches, once the features are identified this way, the classifier weights of the selected features were found by training (unpenalized) logistic regression.

4.4 Predictive Performance Evaluation

Although the predictive performance of text classification is not as an end in itself for these models–this work is concerned chiefly with interpretable, short word lists– it is useful to explore whether classification performance could be used as a proxy to evaluate which method gives "better" results for this word imaging task. (An elaboration on what is meant by "better" appears in the later Section 4.5 on Human Evaluation.)

Train-test split

As a standard procedure in machine learning, the dataset has been divided into two partitions. The documents from the larger partition are used to train a classifier according to a particular model, and the predictions this classifier yields on the smaller partition's documents are compared to their known, true labels. A random-split partition was performed, ensuring that there are four training documents for every test document, and that the proportion of positive examples to negative examples are equivalent in each partition. Rows marked for testing are removed from data set $[X, y]$ and stacked in test set $[X^{\text{test}}, y^{\text{test}}]$.

Procedure

A list of 45 query terms were selected from the list of the most oft-cited countries in the corpus. (This list is provided in Appendix B.)

For each training set X, y associated with each of 47 query words found in the news corpus, and for a cardinality target k held constant across all experiments, the five logistic regression models were established after utilizing one of the above feature selection methods: co-occurence, D-TFIDF, BNS, FDR, or L1LR. Each in turn was used to create its own feature set K', from which a particular input matrix could be crafted $X' = \{x_{ij} : i \in I, j \in K'\}$. This matrix, combined with label vector y, was used to produce a logistic regression model (now of dimensionality k, thanks to aggressive feature selection), leading to a vector of coefficients β' and intercept γ'.

Each of the trained logistic models is then used to generate predictions $\hat{y}$ based on X^{test}. Given an input test vector x^{new} and a logistic model with parameter (β, γ), a probability

$$p(x^{\text{new}}, \beta, \gamma) = \frac{1}{1 + \exp(-\beta^T x^{\text{new}} - \gamma)}$$

can be calculated. For a given threshold $\bar{p}$, a new label is predicted as follows:

$$\hat{y}^{\text{new}} = \begin{matrix} +1 \text{ if } p(x^{\text{new}}, \beta, \gamma) > \bar{p}, \\ -1 \text{ otherwise.} \end{matrix}$$

In these experiments, a threshold of $\bar{p} = 0.5$ was used unless otherwise noted.

The performance of the prediction as compared to the known, true values in y^{test} was evaluated according to four well-known scores: precision, recall, F1 (for a given $\bar{p}$) and Area-Under-Curve (AUC, a metric which averages over all $\bar{p} \in [0, 1]$). Precision measures the ratio of the number of correct positive predictions $\#\{i : \hat{y}_i = 1,\ y_i^{\text{test}} = 1\}$ to the number of positive predictions $\#\{i : \hat{y}_i = 1\}$; recall measures the ratio of number of correct positive predictions to the number of positive examples $\#\{i : y_i^{\text{test}} = 1\}$. As there traditionally exists a tradeoff between these two measures, the measure F1, which is the geometric mean of precision P and recall R, $F1 = \frac{2PR}{P+R}$, can be used as a summarization of the two.

The AUC score requires a sweep of the $\bar{p}$ parameter from 0 to 1, establishing true positive rate:

$$\text{TPR} = \frac{\#\{i : \hat{y}_i^{\text{test}} = 1,\ y_i^{\text{test}} = 1\}}{\#\{i : y_i^{\text{test}} = 1\}} \tag{4.3}$$

and false positive rate:

$$\text{FPR} = \frac{\#\{i : \hat{y}_i^{\text{test}} = 1,\ y_i^{\text{test}} = -1\}}{\#\{i : y_i^{\text{test}} = -1\}} \tag{4.4}$$

Plotting TPR against FPR for each $\bar{p} \in [0, 1]$ provides the receiver operator characteristic for the classification model, and the area under this curve (AUC) is a metric of the model's fitness to the data.

Results The boxplots for precision, recall, F1 and AUC can be found in Figure 4.2. From these figures and statistical comparisons in Tables 4.2, 4.3, 4.4, and 4.5, it is seen that, roughly speaking, L1LR, FDR and D-TFIDF are comparable performers to each other, and together are better than co-occurrence and BNS. BNS performs strictly worse than L1LR, FDR and D-TFIDF are comparable and they are better than BNS; The Co-occurrence method is somewhat in between – comparable to the first three methods (L1LR, FDR and D-TFIDF) in terms of precision, but worse in terms of other three measures (Recall,F1, AUC); Co-occurrence is better than BNS in terms of AUC and comparable to BNS in terms of other three measures.

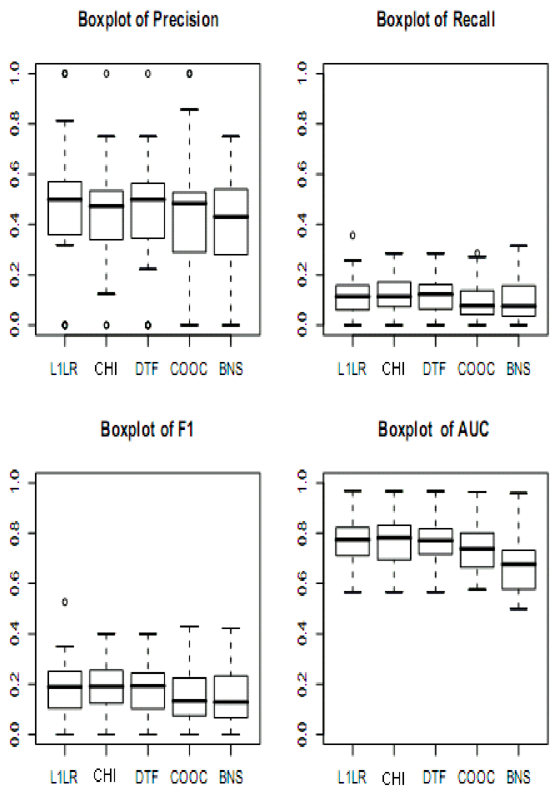


Figure 4.2: Statistical Evaluation

4.5 Human Evaluation

The Experiment

The performance of the five algorithms in terms of classification error has been demonstrated abov. Ideally, for a given query, the main features selected as a by-product of classification

	L1LR	FDR	DTF	COOC	BNS
L1LR		0.29	0.34	0.10	**0.03**
FDR	0.71		0.62	0.20	**0.03**
DTF	0.66	0.38		0.16	**0.03**
COOC	0.90	0.80	0.84		0.14
BNS	0.97	0.97	0.97	0.86	

Table 4.2: Precision. Paired T-test to compare the five different methods on Precision. The table shows the p-values to compare if one method in the column performs better than a method in the row. Significant comparisons ($p < 0.05$) are high-lighted. This table together with Figure 4.2 suggests that BNS is the worst method, and the others are comparable.

	L1LR	FDR	DTF	COOC	BNS
L1LR		0.96	0.87	**0.00**	**0.04**
FDR	0.04		0.11	**0.00**	**0.01**
DTF	0.13	0.89		**0.00**	**0.01**
COOC	1.00	1.00	1.00		0.73
BNS	0.96	0.99	0.99	0.27	

Table 4.3: Recall. Paired T-test to compare the five different methods on Precision. The table shows the p-values to compare if one method in the column performs better than a method in the row. Significant comparisons ($p < 0.05$) are high-lighted. This table together with Figure 4.2 suggests that Co-occurrence and BNS are comparable and both of them are worse than three other methods, which are comparable.

capture an inherent aspect of this query in a given data set. This potential connection is a theory this work can substantiate with a human validation experiment. In this experiment, human readers read random samples of three paragraphs from the data set about a given query and selected which of different generated word lists best captured the "image" of those paragraphs.

As with the classification analysis, all the algorithms were given the same dataset with the same stop-words removed. As before, the queries were drawn from a set of 47 single-word proper nouns $Q \subset J$ containing the names of the most often cited (single word) countries in the corpus. All five algorithms were tested: D-tf-idf, co-occurence, L1LR, BNS, and FDR.

For each subject, the following steps were repeated 60 times:

1. Select two algorithms a and b. Select a query $q \in Q$. Select a decoy-query, $r \in Q$, $r \neq q$. Select three paragraphs about that query, but not the decoy query, uniformly at random (so $X_{kq} > 0$ and $X_{kr} = 0$ for the randomly selected paragraphs k_1, k_2, k_3).

2. Show the subject a screen with the three paragraphs followed by four word lists. The

	L1LR	FDR	DTF	COOC	BNS
L1LR		0.91	0.81	**0.00**	**0.02**
FDR	0.09		0.17	**0.00**	**0.00**
DTF	0.19	0.83		**0.00**	**0.01**
COOC	1.00	1.00	1.00		0.54
BNS	0.98	1.00	0.99	0.46	

Table 4.4: F1. Paired T-test to compare the five different methods on Precision. The table shows the p-values to compare if one method in the column performs better than a method in the row. Significant comparisons ($p < 0.05$) are high-lighted. This table together with Figure 4.2 suggests that Co-occurrence and BNS are comparable and both of them are worse than three other methods, which are comparable.

	L1LR	FDR	DTF	COOC	BNS
L1LR		0.45	0.68	**0.00**	**0.00**
FDR	0.55		0.78	**0.00**	**0.00**
DTF	0.32	0.22		**0.00**	**0.00**
COOC	1.00	1.00	1.00		**0.00**
BNS	1.00	1.00	1.00	1.00	

Table 4.5: AUC. Paired T-test to compare the five different methods on Precision. The table shows the p-values to compare if one method in the column performs better than a method in the row. Significant comparisons ($p < 0.05$) are high-lighted. This table together Figure 4.2 suggests that BNS is the worst, and both Co-occurrence and BNS are worse than three other methods, which are comparable.

word lists are (in a random order) the word lists for algorithm a and algorithm b for both query q and decoy r.

3. Ask the subject to pick their first and second choice for word lists that best capture the "image" of the three paragraphs.

4. On a seperate screen, ask the subject if the three paragraphs are about q, r, both, or neither.

Random permutations of the 10 possible algorithm pairings were cycled through 6 times, in order to balance the number of head-to-head comparisons. The queries were permuted instead of sampled with replacement to increase balance for both the true- and decoy-query selections, with some queries being replicated since there existed had fewer than 60 distinct options.

Two rounds of experiments were administered, with mild differences. The first round consisted of two news experts (an expert in international relations and a former journalist of five years experience, both affiliated with this project). The second round consisted of five undergraduate volunteers. The main distinction between the rounds involved censoring of the word lists.

In the initial pair of experiments, words were from the word lists that appeared in any of the paragraphs. This attempted to ensure that subjects were assessing word lists by their collective content and meaning, rather than simply selecting lists that had high overlap with words from the paragraph text. Upon review, however, censoring rates were higher than anticipated; it appeared the integrity of the word lists was being compromised.

The second trial's word images were not censored. There were also some minor problems with the randomization and with denoting negative words that were corrected. In all experiments, word lists were truncated to the shortest word-list so all lists presented at a time were the same length. All algorithms rank the features, so the truncation always took those features with the highest magnitude weights (ties broken as discussed above).

This work hypothesizes that if, in step (3), the true-query list generated by a Process A were chosen consistently across queries, it is reasonable to conclude that it was capturing something about the paragraphs more than the true-query lists generated by the other Processes and the decoy lists. This would be evidence that the selected Process generally captures the "image" of their queries as long as, in step (4), the reader is agreeing that the paragraph is about what the labeling algorithm thought it is. The closer the step (4) responses are from 100% agreement, the less need to worry about how to interpret the results of the step (3). For this reason, the analysis does not include any item where the subjects did not believe the paragraphs were about the query, or were about both the query and the decoy-query.

Description of the Survey Data

Data from the two rounds were not combined, due to differences in censoring and other aspects discussed above. Only tabulate the second round results are here tabulated. Though not shown here, it was found the analysis using all paragraphs, regardless of quality scoring as assessed by the survey respondent in step (4), also gives results of similar character to below. Additionally, the first pair of trials, conducted with expert readers, had results nearly identical in form to those presented below.

Table 4.6 shows the distribution of how many lists were picked across all items and subjects. Most of the time the subjects did not pick two lists; this trend was generally shared by all subjects. Table 4.7 shows the number of times each algorithm was picked first, picked second, not picked at all ("skipped"), or lost out to a selection of a decoy-query list ("bad"). This last case is truly an indication of poor imaging: subjects don't even prefer the algorithm's word list to a word list based on a separate query word entirely!

Generally, Table 4.8 suggests the first three processes are good performers. BNS seems particularly bad. In investigating why, a first tentative theory points to how classification

# lists picked	% of items
0	13 %
1	60 %
2	27 %

Table 4.6: Number of times where subject picked 0, 1, or 2 lists.

	L1LR	co-occur	D-tf-idf	FDR	BNS
first	60	50	37	15	10
second	8	12	8	3	6
skipped	21	28	48	78	54
bad	3	9	6	11	29
total	92	99	99	107	99

Table 4.7: Tally sheet for algorithms, collapsed across five human evaluators.

might not be enough to capture human meaning. BNS was a decent classifier, performing similarly to the other processes when measured by a variety of classification rates.

Its word lists, however, were dominated by "negative words"—in this data set, it apparently classifies by kicking things out of the class rather than pulling things in. The word lists, therefore, involve a lot of negative thinking (this topic is not about this or that) which seems to be considered poor by the human readers. Furthermore, BNS is not innately sparse—many words have high (negative) weights, all tied or nearly tied. Taking the top 12 may not provide a complete list. Avoiding these outcomes was indeed why methods such as L1LR were sought for this work.

A first test examined whether the given algorithms were producing results beyond simple noise: was a given algorithm's list selected more than 25% of the time when presented to the subject? Under the null hypothesis of no difference for any reviewer of a given algorithm from any other algorithms or decoy-query, assume a binomial distribution (this also requires assuming the items are drawn and answered independently of each other).

Table 4.8 shows the number of first picks, the total number of times the algorithm was presented, and the final p-value for a one-sided Binomial test with $p_0 = 0.25$ being the probability of being picked first. Note that if an alternate algorithm is superior, overall, to the algorithm being tested, then this test becomes conservative. In this analysis, trials where the subject indicated that the paragraphs were not in fact about the target topic (10%), or were about both topics (6%), were dropped.

The methods based on D-tf-idf, L1LR, and co-occurence are all significantly better than chance. Neither BNS or FDR seem to be.[1] Having established the statistical significance of

[1]Although they lie below 25% due, possibly, to the strength of the other processes.

	picked	n	$\hat{p}$	P-value
L1LR	60	92	0.65	0.000
cooccur	50	99	0.51	0.000
D-tf-idf	37	99	0.37	0.004
FDR	15	107	0.14	0.998
BNS	10	99	0.10	1.000

Table 4.8: First column is number of trials process was picked first for valid paragraph sets. Second column is number of trials where the algorithm appeared at all. $\hat{p}$ is proportion of time process was picked first. P-value is for a one-sided binomial test against $p_0 = 0.25$.

the results, their relative performance can now be assessed.

Comparing the Processes

To discover whether one or the other feature selection model is a superior performer, this chapter now tests the hypothesis that a given process is preferred by human judges over all the other processes. (Note: this is distinct from it being preferred to the *average* of the others.) Each trial pits two processes against each other (what can be called a "head-to-head" contest). For each of the 10 possible pairings of process, six trials were conducted.

In a given trial, a process can be picked first, picked second, or not picked. Say a trial pits Process X against Process Y. The contest was scored as being for X if X is picked first, for Y if the reverse, and a tie otherwise.[2]

There are two ways Process A can be, on the whole, superior to Process B. First, it can be picked more often than the other in the head-to-head contests between the two. Second, it can be picked more often than B when pitted against Processes C, D, and E – i.e., it performs better against the *other* processes than B does against those processes. These results are combined two ways to create an overall test statistic of performance.

Given a desired comparison of A vs. B, there exist ten independent slices of the data, assuming the trials themselves are independent. One slice is the head-to-head contests of A and B. Three slices are A vs. C, D, and E. Three are B vs. C, B vs. D, and B vs. E. (Three are unrelated to either A or B.)

For the head-to-head contests, under the null hypothesis of no difference between A and B, there is a 50% chance of A being picked given that one of them was picked. This hypothesis can thus be tested with a binomial test with $p_0 = 0.5$. Let the resultant P-value be p_1.

[2]Alternatively, one could have ranked the picks (so if Process X was picked second and Y not at all, then Process X "won"), but if neither were picked first then a decoy list was picked first, suggesting both X and Y were simply bad.

Under the null hypothesis of no difference between A and B, there is some shared chance p of either A or B being picked over C, D, or E in those contests.[3] This is tested using the (approximate) chi-squared test on the two-way table of process vs. being picked first. Note that the data determining As performance over the others is independent to those of B, so this test is valid by construction. Let the resultant P-value be p_2.

These later data are independent of the head-to-head data, and thus these two hypothesis tests are independent of each other. Let the test statistic of difference be the product of the two P-values, $B \equiv p_1 \cdot p_2$. If the tests behind p_1 and p_2 were continuous and exact, then under the null the P-values are independent random variables distributed uniformly: $p_i \sim Uni[0,1]$. The cumulative density function of $B = p_1 p_2$ under the null of "shared null" is

$$\text{Prob.}\{B \leq b\} = b(1 - \log b)$$

The smaller B is, the more extreme the difference between the two processes. The P-value associated with b, i.e. the chance under H_0 of B being as small or smaller than the seen b, is Prob.$\{B \leq b\}$.

Although both tests are asymptotically exact and continuous, P_1 is a binomial test, which is exact and discrete and P_2 is a χ^2 test on the 2×2 table of Process $\times$ win-loss count, which is approximate and asymptotically exact.[4] The final P-values for the B-statistics are also somewhat approximate. Following this line of reasoning, the results were verified with a permutation test on the distribution of B. Results did not substantively change.

Interpreting a significant result with this test requires a modicum of care. The overall test is built out of two bi-directional tests. If these are in different directions, and there exists a significant result, then the processes could be said to *differ*, but it could not be said that that they are ordered.

Furthermore, such a situation would call into question whether the processes had a significant overall ranking. If both sub-tests point to the same conclusions then they can be interpreted as any normal test of difference—the larger-value bearer is significantly larger or better. But it is important to remember this is a test of difference in something potentially more complex than a single dimension, and so ordering is not necessarily well defined.

Results

Using the above test statistic, results are shown in Table 4.9. For readability, B has been log-transformed (the scale is otherwise tiny) and so higher values are more extreme. Ten of the hypothesis are definitely related. Under a Bonferroni correction, all tests are significant except for L1LR vs CNT, CNT vs tf-idf, and FDR vs BNS. It can be concluded that L1LR is better than all the other methods except, possibly, CNT. All the sub-tests for significant B are in alignment, so interpretation is more straightforward.

[3]Actually, this p could be different depending on the opponent being C, D, or E (assuming topics are random). There is thus a mixture, but the marginal probabilities will be the same when integrating out the

Pr A	Pr B	% A	n	P_1	$\%A - \%B$	n_A, n_B	P_2	$-\log B$	P-value
L1LR	CNT	70	23	0.115	9	69,76	0.300	3.4	0.151
L1LR	tf-idf	60	22	0.503	31	70,77	0.000	8.3	0.002
L1LR	FDR	95	26	0.000	46	66,81	0.000	17.1	0.000
L1LR	BNS	94	21	0.000	50	71,78	0.000	15.8	0.000
CNT	tf-idf	62	25	0.383	11	74,74	0.257	2.3	0.327
CNT	FDR	95	27	0.000	26	72,80	0.001	17.0	0.000
CNT	BNS	75	24	0.077	43	75,75	0.000	10.2	0.000
tf-idf	FDR	67	26	0.302	25	73,81	0.001	8.1	0.003
tf-idf	BNS	79	26	0.057	26	73,73	0.001	9.8	0.001
FDR	BNS	80	28	0.109	-2	79,71	0.785	2.5	0.297

Table 4.9: Comparing Process Performance. Processes scored by counting the number of times they were picked first. Direct trials between A and B with no first-place winner are dropped. Trials between A or B vs. the others are counted as win if A or B was picked first and a loss otherwise. Trials where the content validation step (on the paragraphs) failed (as in the paragraphs were deemed to not be about the target topic) are also dropped. Thus the ns (and power) vary by test.

The direction of all head-to-head comparisons produces the order L1LR, co-occurence, D-tf-idf, FDR, BNS. All pairwise comparisons follow this ordering. Although the top three are not entirely separated, given the original hypothesis that L1LR would produce better lists due to its consideration of interfeature collinearity, the above data do suggest its superiority. Furthermore, under the P-value for the comparison of L1LR to co-occur went down to 0.04 under the permutation test. And again, these results are echoed in the first pair of trials.

4.6 Conclusions

Sparse text classification can be a valuable tool for social scientists engaged in media studies. By allowing quick summarization of words and concepts as they are portrayed in the media, it can provide a useful starting point on the analysis of how news media may or may not shape their readers's perceptions of the world. But for the purposes of media analysis require an assessment of these instruments by means other than simple classification performance. As an alternative, this work used human evaluation of sample paragraphs and produced word lists. It is important to not only evaluate algorithms based on classification error when the object of interest are the features themselves.

other processes.

[4]Although the values for P_2 are made more exact using Monte Carlo methods.

The algorithms studied perform more or less equally well in terms of predictive performance, with slight edge to sparse logistic regression (L1LR), Delta-TF-IDF and FDR. Of perhaps greater interest, it was shown that L1LR seems to produce lists that are judged by humans to be superior summaries. Further research will concentrate on solidifying the human evaluation surveys with larger sample sizes. It is hoped that the approach can guide the development of machine learning algorithms and performance metrics that are well attuned to human requirements in word imaging tasks.

Chapter 5

Sparse PCA for Text Summarization and Exploration

5.1 Corpus summarization

The volume of content is ever increasing. When an investigator faces a large body of documents, two useful objectives appear:

1. It would be nice to select a small number of documents which typify the corpus, documents which are the least "outlying" from the corpus overall.
2. The corpus's many documents are themselves written in a vocabulary of many words and multi-word phrases. It would be nice to select a small number these word (or phrase) vocabulary tokens which explain much of the diversity in the corpus.

Low-rank matrix approximation can help achieve these two objectives. Specifically, this chapter proposes that sparse principal component analysis (PCA) provides a complete framework for accomplishing both tasks in a way that keeps the investigator fully cognizant of the meaning of the results. Many familiar metrics, e.g., fraction of total variance explained, may suffer given that these goals require production of exceedingly simple models. One may hypothesize that this sparse formulation presents a good candidate for these objectives and conditions.

The Salton matrix

A corpus can be represented as a matrix $X = \{x_{ij}\}$, with each row $i = 1, \ldots, m$ corresponding to a particular document, each column $j = 1, \ldots, n$ corresponding to a particular text token, and each element encoding patterns of how each token appears in each document, resulting in a large, sparse matrix. This encoding of value x_{ij} is open to a variety of methods. One could use a count of the number of times token j appeared in document i, or a binary matrix where x_{ij} is a 0/1 indicator of which tokens appeared in which documents. The TF-iDF approach

[50] is a popular representation for many text processing applications, and the effects of it and of several other candidate representations are tested in [43]. It's important to remember that choice of this encoding can greatly impact the results. This paper refers to this matrix (under any encoding) as the "Salton matrix", and it is with low-rank approximations of this matrix that one could hope to achieve the corpus summarization and exploration goals from above.

Latent semantic indexing and its descendents

Latent semantic indexing (LSI) [16] is an application of linear dimensionality reduction via principal component analysis to the Salton matrix. By projecting the document vectors onto the first k principal components, a "reduced matrix" is constructed. The columns of this reduced matrix now correspond, not to particular text tokens, but to specific linear combinations of tokens.

This technique is a commendable addition to many text processing algorithms, but given these goals, there is an acute drawback. The components produced are too dense to function as a keyword list or ranking. For example, the model parameters are each sensitive to all the tokens, and interpreting them in light of this is frustrating. Though LSI has inspired more improved models such as probabilistic LSI [28], the latent dirichlet allocation (LDA) [4], etc., there remains an implicit constraint being violated. The number of tokens used by any model attempting to achieve this paper's goals need to be few enough in number – perhaps dozens, at the most – for an investigator to conceive of them all together at once.

Sparse PCA

Principal component analysis can be reformulated to find a low-rank approximation of a Salton matrix (by finding directions of greatest variance) alongside a penalty for non-zero model parameter values. Finding the solution to this regularized objective can be posed as a semi-definite program [15]. This sparsity regularization will necessarily bias the solution away from the best possible approximation, but to the benefit of a human-interpretable model. The regularization parameter can be increased until the model found has sufficiently few tokens in the components. When comparing how much variance is explained by selecting a few tokens in this fashion versus, e.g., how much variance is explained if one uses only the top few components derived by normal PCA, the benefits of sparse PCA become clearer.

Finding this approximation can be greatly sped up by using safe feature elimination. It can be shown that any token feature with sample variance less than the regularization parameter must be zero in the final solution, and so can be dropped from the Salton matrix when calculating the sparse principal components. This holds for many cases of ℓ_1 regularization, as seen in [17]. The number of features so dropped is a conservative lower bound on the total number of zero-weight features in the final solution. While dropping low-variance features is a common first step in ordinary PCA, this necessarily means the final solution reached must be an approximation; that is not the case here. Dropping low variance features in sparse PCA has no downside; an exact final solution can be calculated much faster than in the dense case.

5.2 Data Exploration

The keywords included in the sparse principal components are interesting and informative, but the model also presents more exploratory opportunities:

Visualizing the corpus

When two or three principal components are used in the approximation, the document data points can visualized in a plot. Because of the few tokens involved in producing the principal components, the user has an immediate and complete grasp of the word use distinguishing documents from different areas of the plot. The same visualization for dense components introduces more ambiguity: is a high value along a component due to the use of a few strongly-weighted tokens, or the accumulation of many weakly-weighted ones? This is an easier question to answer when it's known the number of tokens utilized is small.

Well-approximated documents

The span of the k sparse principal components of the Salton matrix form a subspace of the token-space defining the original document data points. A residual can be found representing the difference between the original data point and its projection into the sparse-PC span. It is then easy to identify and highlight documents with small-magnitude residuals. PCA has been used to implement anomaly detection by identifying points which are far from this subspace [36]; this chapter's hypothesis merely takes the corrollary that points close to the subspace should be broadly representative.

5.3 Demonstrations

International News

For a first demonstration, this procedure was conducted over the collection of 1,288 news articles published by the *New York Times*'s International section mentioning the word "China." The documents were tokenized by unigrams, removing no stop words, and performing no stemming. The Salton matrix encodes the binary 1/0 appearance/non-appearance of tokens. Table 5.3 contains the tokens and respective weights used in the two dominant principal component. The first component implies much variance occurs in terms of China's international standing, especially vis a vis the United States and the United Nations; the second component of keywords have a domestic component, countered by another international agent (Russia). Table 5.3 shows the headlines for articles closest to their sparse-PCA projections, along with the magnitude of this residual. It is interesting that all these articles are of short length.

1st comp. token	*Weight*	*2nd comp. token*	*Weight*
`states`	0.3929	`chinese`	-0.5788
`united`	0.3913	`beijing`	-0.5578
`american`	0.3195	`chinas`	-0.5394
`obama`	0.3155	`russia`	0.2507
`president`	0.2858		
`washington`	0.2792		
`countries`	0.2633		
`nations`	0.2618		
`administration`	0.2457		
`international`	0.2271		
`would`	0.2058		
`nuclear`	0.1976		

Table 5.1: Sparse principal components for articles mentioning “China.”

In Figure 5.1, there is a depiction of the news articles projected onto these two principal components. There remains a healthy diversity in the energy accrued in each dimension. The banding seen along the second principal component is due to the limited number of tokens found for this direction combined with the necessarily-integer number of appearances of those tokens in their parent articles.[1]

Validation by Reader

Much prior work in [52], [12], etc., suggests using human survey respondents to evaluate how well an approach such as ours works. Respondents could be instructed to read many dozen example documents (drawn at random), then solicited for scores of a) how reasonable they find the word lists generated by looking at the sparse principal component features and b) how representative they feel the well-approximated documents are, given their reading. This is then repeated for word lists and exemplary documents generated by another method, whether thresholded LSI or LDA.[2]

It will be interesting to compare the results of this survey to other, more quantitative metrics. One would naturally expect a rather poor approximation (using only a few tokens

[1]The DSPCA algorithm of [15] requires binary search of the sparsity penalization parameter to produce a sparse principal component of exactly k non-zero elements; it can be difficult to precisely hit a small number before machine precision forces it to elect a direction of all-zeros.

[2]It is of course important to get validation as well for all these approximation methods run on different Salton representations, as mentioned in the above section.

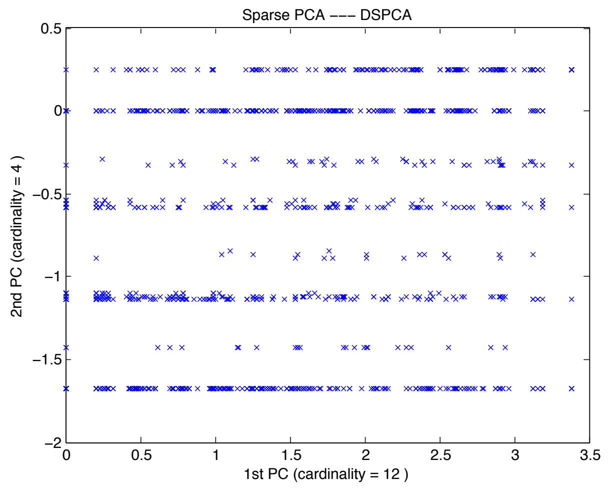


Figure 5.1: Corpus as projected onto its two sparse principal components

in the projection is going to mean only a low percentage of the overall data variance is explained), but it would be intriguing to see what relationships obtain between the survey respondent scores and the effectiveness of the sparse approximation error.

Tweets about Cancer

A text dataset was collected from the social network Twitter. Each text unit was a sole message posted to the service (a "tweet"), constrained to be of maximum 140 characters in length. These tweets were found through querying the service for four phrases: *cancer*, *cancer prevention*, *cancer screening*, and *cervical cancer*. Together, a total of 85,504 distinct postings were collected between April 17, 2012, and April 25, 2012.

A Salton matrix X was constructed, such that the element at the ith row and jth column obeyed:

$$x_{ij} = \frac{c_{ij}}{\sum_j c_{ij}} \tag{5.1}$$

where c_{ij} encodes the number of times the token j appeared in tweet i. This is akin to projecting the bag-of-words counts vector into a simplex; all rows of X have unity ℓ_1 norm. x_{ij} can be thought to encode the concentration of keyword j in tweet i.

The results of performing sparse principal component analysis on this matrix can be found in Table 5.3. The first component encodes a commonly repeated ("retweeted") message:

A 15 year old girl who is terminal with cancer has made a bucket list. One of her wishes is to trend on Twitter.Give a RT #alicebucketlist

Res. Mag.	*Article Headline*
66.593085	"World Briefing — Europe: Vatican: New Language for Web Site"
75.671066	"Aid Sought for Students"
78.128291	"World Briefing — Asia: Taiwan: Agreement With China Opens a Rare Diplomatic Door"
79.107418	"World Briefing — Asia: China: Use of Controversial Software to Filter Web Is Optional, Official Says"
79.108141	"World Briefing — Asia: China: Political Site Is Shut Down"
80.052708	"World Briefing — Europe: Global Arms Spending Up, Study Shows"
80.132263	"World Briefing — Asia: China: Court Upholds Sentences Stemming From Riots"
80.532884	"World Briefing — Asia: China: Ceiling Collapse Kills at Least 11 Workers"
80.723861	"World Briefing — Asia: China: Border to Korea Reopens"
81.543736	"World Briefing — Asia: India: Dalai Lama on Hacking"

Table 5.2: Headlines of the best approximated articles, along with the magnitude of the residual left by subtracting the projection of the document vector onto the span of the two principal components.

Over 4,500 of the dataset tweets were repeats of this message, verbatim. The phrases define a large component in the overall corpus variance, and split the corpus into two: tweets lending support to Alice, and the rest.

The second component is dominated by another highly retweeted comment:[3]

> *Leo, Capricorn Aries , Cancer, Gemini, Sagittarius, Scorpio, Virgos* <3<3<3<3<3<3<3<3<3<3<3<3<3<3<3<3<3<3[...]<3<3<3 *#TheBest RT If your one*

This particular message appeared in over six hundred of the scraped messages, and in general astrology appeared quite often (cancer being a sign of the zodiac and all).

In this way, the principal components are building an approximation of the overall dataset. They settle on word groups which define large swaths of retweeted material. It's an additive style: starting from nothing, what tokens could be used to build a tweet which would fit in with this corpus? The first two principal components choose two widely repeated families; Alice's bucket list and the astrology scene.

[3]Many of the ASCII hearts have been omitted for reasons of formatting. The original tweets contained 49 in all.

Uncentered Data		*Centered Data*	
First sPC	*Second sPC*	*First sPC*	*Second sPC*
cancer	<3<3<3<3<3<3<3	rt	cancer
rt	#thebest	old	cervical
old	scorpio	girl	prostate
year	bigfreaks	#alicebucketlist	prevention
#cancer	sagittarius	terminal	cure
girl	sagattarius	bucket	lung
#alicebucketlist	capricorn	trend	*hurt
terminal	#rt	year	*home
trend	leo	list	*best
bucket	scorpios	wishes	*emotional
list	libra	twitter	*like
wishes	virgos	retweet	*heart
twitter	pisces	twitter.give	*signs
breast	taurus	*list.one	*love
retweet	gemini	*screening	*#cancer's
give	aries	*breast	*#cancer

Table 5.3: Two Sparse Principal Components for the cancer-tweets dataset, with and without centering of bag-of-words representation

Centering the data

Table 5.3 contains two additional principal components found after repeating the procedure on a transformed version of the simplex Salton matrix X:

$$\hat{X} = X - \mathbb{1}_m \mu^T \tag{5.2}$$

where $\mathbb{1}_m$ is an m-dimensional vector of 1's and μ is a n-dimensional vector with elements defined:

$$\mu_j := \frac{1}{m} \sum_{i=1}^{m} x_{ij} \tag{5.3}$$

(Note that though $\hat{X}$ is dense, it is unlikely it would ever need to be explicitly formed. The operations used to calculate the principal components involve only using the product of the Salton matrix and a vector. This can be performed in two steps, distributing the vector to be multiplied by X and μ^T, and taking the difference of the resulting vectors, saving both processor time and memory space. Code implementing this is presented in an appendix.)

The analysis now seeks collections of words which capture much of the variance as a function of deviation from the average word concentration across the tweets. One consequence of this is that the coefficients assigned to the keywords are no longer monolithically positive. (The table distinguishes negative-signed terms with an asterisk.)

The principal components are describing directions of diversity in the corpus, and keyword tokens can define opposite poles of this direction. The first principal component with centered data recovers the #alicebucketlist tweets, but we now also see `breast` and `screening` with an opposite sign. We're being told that a major swath of the corpus can be defined either by the sentimental chain letter, or by the more clinical vocabulary.

A similar result appears in the second principal component: medical issues are addressed in one direction (*prevention, cure, cervical, lung*) while the astrological terms define the others.

A worthwhile course of study could examine in what situations one or the other principal component analysis – centered or uncentered – is appropriate for what domains. When would media scholars prefer to see dichotomies arise (centered), and when more straightforward summaries (uncentered)?

Chapter 6

Conclusions and Future Directions

Here this work will now discuss the pertinent findings of the previous four chapters before exploring some of the frontiers this work suggests.

6.1 The preceding chapters

Document interpolation

An example text mining task was presented: given two document units on a related theme, find a series of documents which help link these two input items from a large set of candidate documents. The vector space model was employed, and from it, an algorithm based on finding the documents whose vector representations were nearest the convex combinations of the inputs.

This approach was compared to two other known algorithms over articles taken from the *New York Times* international section. The first alternative was a known substandard approach; the other was a state-of-the-art method. The results of this nearest neighbor approach was shown to significantly match the sophisticated method, at a lower computational complexity.

This should overall stress the usefulness of working with text-as-vectors.

Databases for text mining

This chapter stepped through the process of designing a relational database for the sorts of text mining tasks approached in this dissertation. The architecture traded off the organizational and data quality advantages of normalization versus the performance boosts from replicating entity attributes across tables.

The vector space model of text was implemented explicitly as a table: the sparse Salton matrix was built in a table with a column for a document ID, a word token ID, and a count value reflecting the importance of that word to that document. (And since news analysis so reliably hinges on changes over time, a column of dates associated with each document was added to allow easy selection of only the pertinent rows.)

Feature selection and sparse modeling

A predictive framework was established for summarizing text. A query term's word image was crafted by comparing documents which mention the query to those which lack it. By selecting a small number of word tokens which help distinguish these two classes of document, the query can be succinctly summarized.

A series of algorithms were proposed, four of which were independent feature models – models whose coefficients for each keyword could be calculated independently of the others, taking the image as a topset truncation of these coefficients. The fifth was an implementation of ℓ_1-regularized logistic regression, a model which can be fit with convex optimization algorithms, and which formulates each coefficient weight with reference to the other keyword coefficients.

To evaluate the effectiveness of these approaches, the *New York Times* international section dataset was again prepared. A set of queries was built from the names of frequently mentioned countries and regions. The word images found by the feature selection methods were tested for predictive value; no model defined by these keywords performed well.

Additionally, volunteers were convened to assess the quality of the word images as news summaries. Cooccurrence and ℓ_1-regularized logistic managed to perform well by this metric.

Sparse principal component analysis and text exploration

For text mining tasks that do not lend themselves to a predictive framework, a sparse variant of latent semantic indexing/principal component analysis was investigated. The *New York Times* international pages was again mined, this time for articles mentioning "China." Sparse principal components were found as a summary of China in these documents. The residual distance between the plane defined by the sparse principal components and the actual documents in higher-dimensional space was used as a statistic to extract representative documents from the set.

Additionally, a series of tweets relating to cancer were mined from the social network Twitter. Sparse PCA was applied to this dataset twice: once with the text data uncentered, and once with the mean keyword-appearance counts subtracted from each document vector. (This subtraction did not need to disrupt the sparse-matrix data structure that had allowed efficient storage of large corpora and powered the other analyses up to this point.)

Comparisons were drawn between the two styles of output. Uncentered results suggested an additive model, where each principal component defining a class of tweets; adding more principal components could narrow down more and more families of tweet. The other approach, with centered data, appeared to identify dichotomies, where two major strands of document crowded the other out.

6.2 Investigating Text Representation

A key factor in the performance of these sparse methods for summarization is the choice of vector space model. Anecdotally, it was clear in the dry runs of the experiments in this dissertation that a poor choice of text representation could make even a solid algorithm look terrible.

A large class of text representations begins with a standard Salton matrix, C, with elements c_{ij} encoding the number of times term $j \in \{1, \ldots, n\}$ appears in document $i \in \{1, \ldots, m\}$. This is frequently updated with the use of two matrices, $L \in \mathbb{R}^{mxm}$ and $R \in \mathbb{R}^{nxn}$ to generate a matrix X:

$$X := LCR$$

The most famous example of this is the tf-idf representation, with L and R diagonally set to depress values found in well-populated rows or rarely-nonzero columns. This dissertation has featured several such representations:

Rescaling rows and columns In Chapter 2, X was formed by rescaling each column of C to have unity ℓ_1 norm. In Chapter 5, X was formed by rescaling each row of C to have unity ℓ_1 norm. Each is equivalent to setting L or R to diagonal matrices, with elements set to the inverse of the sum of the respective row or column.

Stop word removal Dropping stop words, as in Chapters 2, 4, and 5, (and query words as well, for Chapter 5), is akin to setting R to a diagonal matrix where each diagonal element is a 1 (for non-stop words) or 0 (for stop words).

Stemming The stemming performed in Chapter 2, where multiple words are mapped to a shared root term, can be thought of as constructing R as a binary matrix. Each column of R would correspond to a root, each row to a keyword token, with elements of R only set to 1 when the keyword of that row mapped to the stem of that column.

PCA Projection The projection of documents into the plane defined by the sparse principal components of Chapter 2 was found by building an R matrix by concatenating the two sparse principal component vectors (with coefficients normalized to ℓ_2 unity).

Each of these manipulations can lead to substantial swings in qualitative and quantitative aspects of the trained models. To investigate this further, colleagues and I established a larger-scale version of the experiment in Chapter 3, currently available at [41].

Figure 6.1, taken from this work, demonstrates the interactions between term-document matrix encoding, feature selection algorithm, and scores awarded by volunteer readers to the resulting word images.

At top left, average reader score is shown for three encodings (X as tf-idf, as rescaled-ℓ_2-unity columns, or as simple stop word removal) against four feature selection methods (cooccurrence and L1LR, as in Chapter 4, plus correlation of keyword column with label

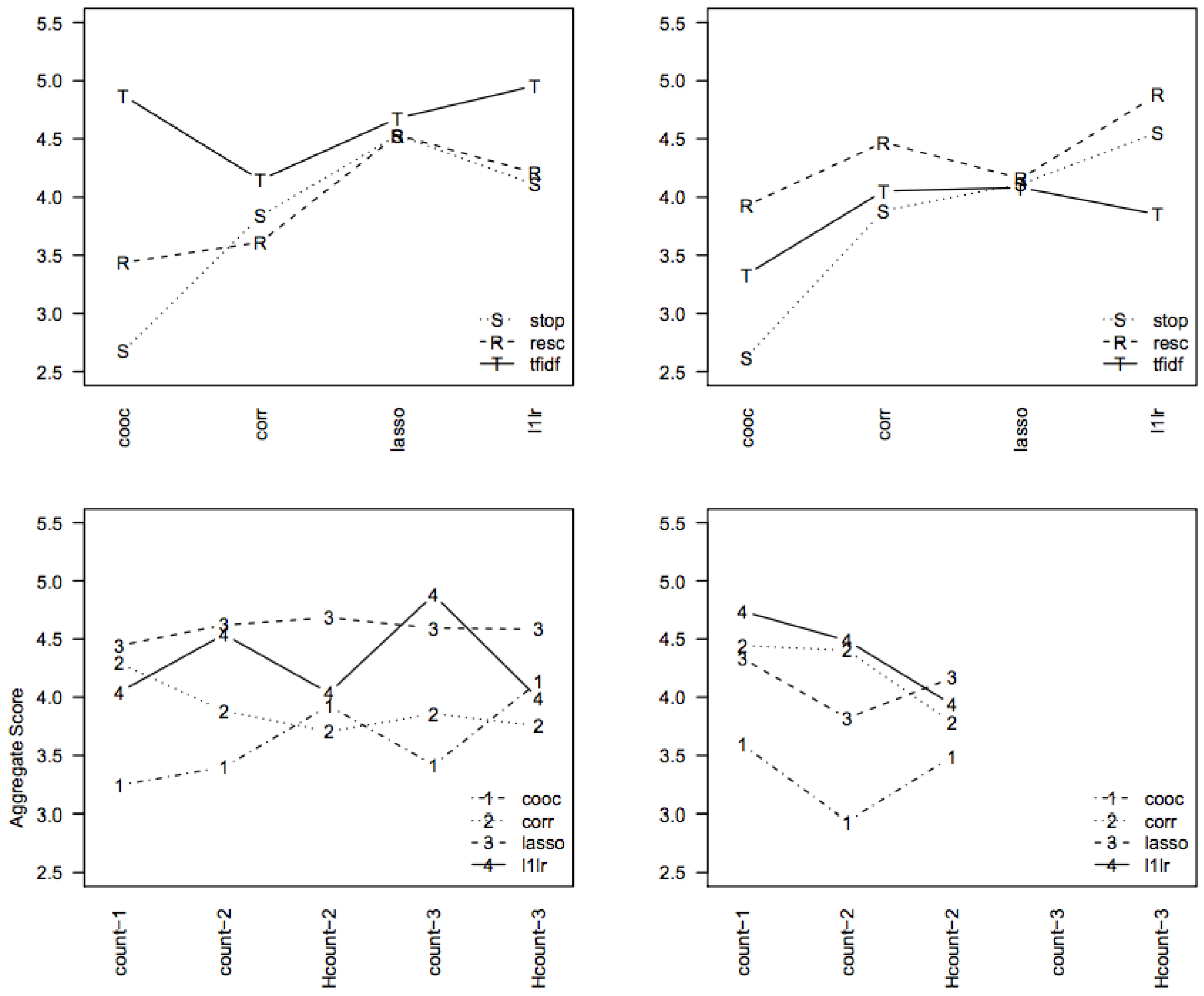


Figure 6.1: Interactions between vector space representation, word-imaging algorithm, document unit labels, and mean human validation preference scores (the vertical axis in all plots). Plots at left demonstrate interactions for doc. units taken as full articles; at right, doc. units as single paragraphs. Plots at top demonstrate interaction between feat. sel. method and vector space rep.; at bottom, between feat. sel. and labeling stringency. (Taken from [41])

vector, and the LASSO), with document units defines as full article texts. At top right, this interaction is replotted, for results when using single paragraphs as the document unit.

Below, interactions are shown for the four feature selection methods against five labeling rules. *count-*{*1,2,3*} define a document unit as matching the query if the query appears in it $\{1, 2, \text{or } 3\}$ times, respectively. *Hcount* does as well, but only includes the document unit in the negative example partition only if the query never appears in the document. These again are repeated at both the article and paragraph level.

The LASSO results were found to be of consistently high average, with special consistency against both labeling style, document unit designation, and vector space representation. It seems a good choice for new domains of text corpora looking to implement a convex approach to text summarization. Tf-idf, as well, seems a resilient choice. Since often the length of document unit is non-negotiable (e.g., Twitter corpora), it's important to plan for methods that are robust in many conditions.

6.3 Real World Use

It's clear that compared to the status quo of hand-coded, human-driven analyses, the results of these summarizers are a) much faster to compute and b) much less informative than what an informed, well-read human could explain. A compelling use case, then, would be a situation where cursory analysis is required (simple analysis), but at a large scale (many times over).

Figure 6.2 presents one possible visualization of many passes over the same Salton matrix. In turn, each of the top 10,000 most-frequent keywords were taken as a query. A short image was created. These 10,000 images now help define a graph: word tokens as vertices, with edges between nodes defined if either appears in the other's image list.

Figure 6.3 is a screencapture of a real-world deployment of a similar concept. Taken from `http://infomous.com/site/economist/`, it is a visualization of the comments left on the website of the *Economist*. A control widget at bottom right allows the user to specify a time frame, from one day to one month – suggesting multiple Salton matrices, all updating at least once per day.

It's unclear what exact approach Appinions is using to build their word-association edges in the graph, or how they would compare to those tested in this dissertation. It remains an instructive case of where large scale word associations can be useful. The corpus large, spread out, and not as easily accessed as the archived text of major newspapers.

It suggests other promising applications of this technology should likewise look for corpora of documents with rapid and broadly-distributed authorship. Between the *New York Times* and Twitter corpora used in this dissertation, the Twitter dataset may be more indicative of future need.

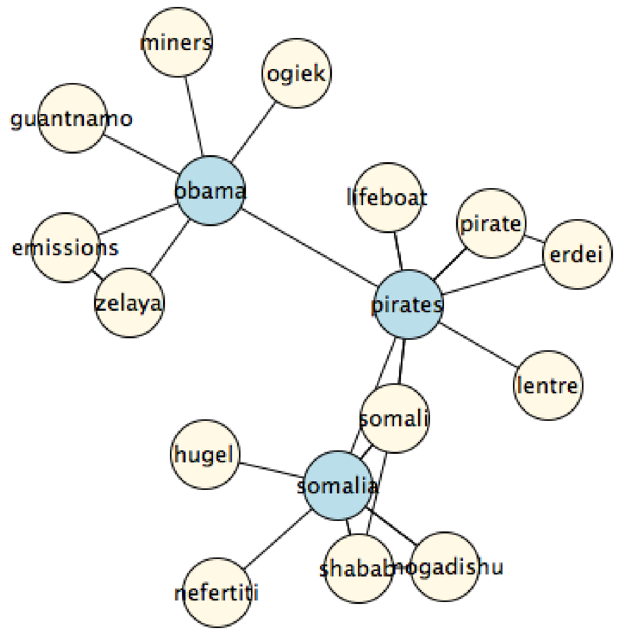


Figure 6.2: A network of predictors produced from *NYTimes* data.

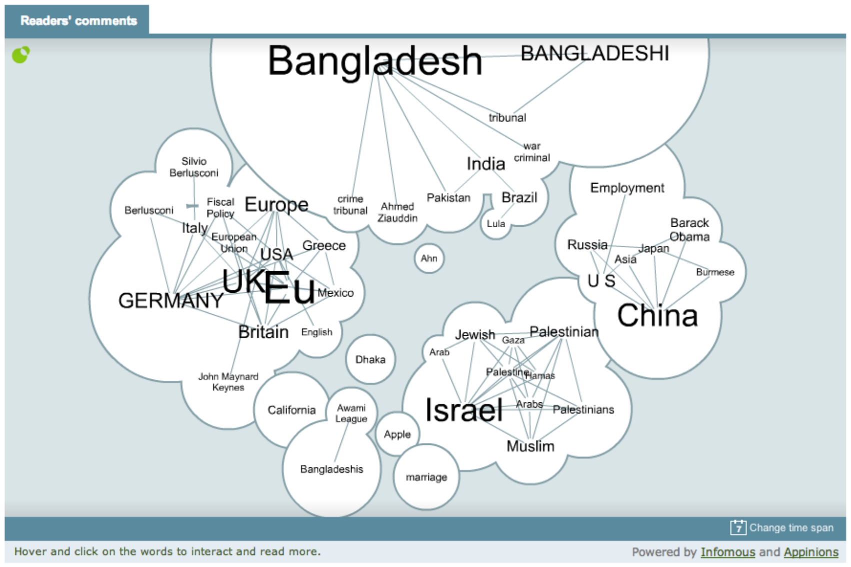


Figure 6.3: A visualization of reader comments from the Economist, by Appinions and Informous.

6.4 Scaling Up

The experiments performed here have been premised on the idea that news today is of great volume, and approaches to mining this text should be designed to respect that as a given. However, many of the techniques described are largely built around traditional, local-machine computing.

At 700,000 articles per day, at 500 word tokens per article, at (an estimated average of) 16 bytes to store a word token-document-count Salton matrix entry, it would take roughly 200 days to fill a terabyte of storage. Just a few years of the modern news is enough to eclipse what can be reasonably dealt with by a lone machine, configured as the ones in this experiment were.

Storage itself would need rethinking. Much of the vogue in databases of late has been to eschew relational databases, for key-value style approaches. Applications such as Google's BigTable [11] have found that moving to "NoSQL" storage solutions have allowed for easier distribution: to share the data across several machines. With volume growing rapidly, this gives hope for allowing this dissertation's techniques to run on full-sized datasets.

Algorithmically, it becomes important to allow the computation to also scale. For the independent feature models considered in Chapter 5, this is fairly straightforward. (In fact, the "word count" exercise is practically a canonical tutorial demonstration of distributed computing for frameworks like Hadoop or Spark, and it's nearly identical to the cooccurrence feature selection method.)

For LASSO, L1LR, sparse PCA, and other more complicated summarization techniques, the path forward is somewhat murkier. Stochastic gradient descent methods have been proposed for distributed solvers [58]: stream over multiple partitions of the data stored on each machine, regularly stopping to average the current state across worker nodes. More recently, a promising algorithm has been put forward for exact solutions via the alternating direction method of multipliers. [8]

6.5 Final Thoughts

These techniques discussed in this dissertation are intriguing. Repurposed machine learning algorithms seem to be able to learn enough from their source data to turn around and point out insights for their designers. The results here are simple alone, but their potential to spread to many domains, learning from considerably more text than any one person could consume themselves is inspiring.

I hope that the study of the news can be assisted with these techniques. It's important to society to trust what we're hearing, to know we're safe in knowing what we think we know. I hope the techniques grow accessible, easily plugged into anything spouting new reporting or archiving the old, so that when there's concern or skepticism about what's been broadcast, we'll have a broadly available source of evidence to use in supporting solutions.

Appendix A

Database Implementation Examples

In chapter 3, this dissertation discussed how a database for news mining should be designed. This appendix presents several examples of scripts interacting with such a database. This Python software uses the MySQLdb module, available at `http://mysql-python.sourceforge.net/`.

A.1 Inserting articles

Here is a script to initially populate the database with news data. This code presumes that a series of marked-up articles are stored in many files in the local directory `./data`. An example file layout for files scraped from a websource's opinion section, displaying articles on January 1, 2010, and November 9, 2010:

```
<ART>
<URL>http://websource.com/2010-01-01/opinion/piece1.html</URL>
<PAR>This is the first paragraph in the first article.</PAR>
<PAR>This is the second paragrpah in the first article.</PAR>
</ART>
<ART>
<URL>http://websource.com/2010-11-09/opinion/piece125.html</URL>
<PAR>This is the first paragraph in the second article.</PAR>
<PAR>This is the second paragrpah in the second article.</PAR>
</ART>
```

Files like this would traditionally be the result of some preprocessing, e.g., scraping from a website or RSS feed, then transformed to meet this simple standard. Note that the URL performs double-duty, encoding both the article source and the article publication date (this is common for content management systems that power web news sources).

The code

```
import string, sys, os, MySQLdb
from math import log10
from time import localtime
from time import asctime

def popArt(articles):
    if '<ART>' not in articles:
        return []
    artBegin = articles.index('<ART>')+5
    artEnd = articles.index('</ART>')
    urlBegin = articles.index('<URL>')+5
    urlEnd = articles.index('</URL>')
    return [articles[artBegin:artEnd], articles[urlBegin:urlEnd],
     articles[artEnd+6:]]

def popPar(archive):
    if '<PAR>' not in archive:
        return []
    parBegin = archive.index('<PAR>')+5
    parEnd = archive.index('</PAR>')
    return [archive[parBegin:parEnd], archive[parEnd+6:]]

def stripPar(par):
    count = 0
    unicodecount = 0
    out = ""
    for k in range(len(par)):
        char = par[k]
        if unicodecount == 0 and char == '&':
            unicodecount = 1
        if char == '<'  or par[k:k+4] == '&lt;':
            count = count + 1
        if (count == 0 and (char.isalpha() or char == ' ') and
          unicodecount == 0):
            out = out + char.lower()
        if char == '>' or par[k-4:k] == '&gt;':
            count = count - 1
        if count < 0:
            return "ERROR ERROR NEGATIVE BRACKET COUNT"
        if unicodecount == 1 and char == ';':
            unicodecount = 0
    return out

def getDate(url):
    dateBegin = url.index('.com/') + 5
    dateEnd = url.index('opinion') - 1
    return string.replace(url[dateBegin:dateEnd], '/','-')

##############################################
########### END FUNC DEFS ####################
##############################################
########### BEGIN DB INSERT ##################
##############################################

stopfile = open('stop_words', 'r')
stopdata = stopfile.read()
stopwords = stopdata.split()
stopfile.close()
for k in range(len(stopwords)):
    word = stripPar(stopwords[k])
    stopwords[k] = word.lower()

db = MySQLdb.connect(host='localhost', user='root', db='nytoped')
c = db.cursor()

lex = []
artcount = 1
parcount = 1
filenames = os.listdir('./data')
print filenames
for filename in filenames:
    fid = open('./data/'+filename, 'r')
    articles = fid.read()
    fid.close()
    author = filename[:-4]
    artList = popArt(articles)
    while artList != []:
        article = artList[0]
        url = artList[1]
        date = getDate(url)
        articles = artList[2]
        c.execute("""insert into ARTICLES (ArtID, Author, Section,
         Source, ArtText, URL, Date)
             values (%s, %s, %s,
%s, %s, %s, %s)""", (artcount, author, "Opinion",
"NY Times", article, url, date))
        parList = popPar(article)
        while parList != []:
            par = parList[0]
            article = parList[1]
            c.execute("insert into ARTPAR (ArtID, ParID, ParText)
             values (%s, %s, %s)", (artcount, parcount, par))
            words = string.split(stripPar(par))
            seenit = []
            for word in words:
                if word not in lex:
                    lex.append(word)
                    wordNum = lex.index(word) + 1
                    c.execute("insert into WORDS (WordID, Word, StopCode)
                        values (%s, %s, %s)", (wordNum, word,
                         word in stopwords))
                wordNum = lex.index(word) + 1
                if wordNum not in seenit:
                    seenit.append(wordNum)
                    c.execute("insert into PARWORD
                     (ParID, WordID, Count, Date)
                     values (%s, %s, %s, %s)",
    (parcount, wordNum, words.count(word), date))
            parList = popPar(article)
            parcount = parcount + 1
        artList = popArt(articles)
        artcount = artcount + 1
print str(parcount) + " paragraphs across "+str(artcount)+" articles."
```

A.2 Generating Reports

With the database populated, it can now be used to generate many useful reports just from SQL interactions. (And of course, the sparse Salton matrices described in the previous chapters can also be easily dumped, for executing the more sophisticated convex problems that yield summarizations.) The following subsections provide sample reports.

Author’s Favorite Words

When provided an author’s name, this script returns the number of articles on file for that author as well as the most commonly used words inside those articles (ignoring stop words).

Code

```
import string, sys, os, MySQLdb

db = MySQLdb.connect(host='localhost', user='root', db='nytoped')
c = db.cursor()

author = string.strip(raw_input("Enter author name: "))
if ';' in author or '%' in author:
    print "WARNING: DANGEROUS INPUT DETECTED."
    sys.exit()
c.execute("select Count(*) from ARTICLES where Author = %s", (author,))
artCountTup = c.fetchone()
artCount = artCountTup[0]
if artCount == 0:
    print "Sorry, "+author+" does not appear in our records."
    sys.exit()
print str(artCount) + " articles located.\n"+author+"'s most commonly" +
    + " used words (stopwords ignored):"

c.execute('''select WORDS.Word, Sum(Count)
    from (((ARTICLES inner join ARTPAR
                on ARTICLES.ArtID = ARTPAR.ArtID)
            inner join PARWORD on ARTPAR.ParID = PARWORD.ParID)
        inner join WORDS on PARWORD.WordID = WORDS.WordID)
    where ARTICLES.Author = "'''+author+'''"
        and WORDS.StopCode = 0
    group by WORDS.WordID
    order by Sum(Count) desc limit 10;''')
wordTups = c.fetchall()
for tup in wordTups:
    word = tup[0]
    count = tup[1]
    print word.upper()+": "+str(count)+" appearances."
```

Execution

```
gawalt$ python report1.py
```

```
Enter author name: Thomas L Friedman
6 articles located.
Thomas L Friedman’s most commonly used words (stopwords ignored):
OBAMA: 32 appearances.
PRESIDENT: 24 appearances.
IRAN: 21 appearances.
NEED: 18 appearances.
GOING: 16 appearances.
LIKE: 16 appearances.
JUST: 14 appearances.
OIL: 14 appearances.
PRICES: 14 appearances.
BUSH: 13 appearances.

gawalt$ python report1.py
Enter author name: Bozo The Clown
Sorry, Bozo The Clown does not appear in our records.

gawalt$ python report1.py
Enter author name: Fake Name"; drop table ARTICLES;
WARNING: DANGEROUS INPUT DETECTED.
```

Date-range’s Frequent Words

Given two date strings (YYYY-MM-DD) as input, finds the words most frequently used across this timespan.

Code

```
import string, sys, os, MySQLdb

def datecheck(date):
    if len(date) == 10:
        if date[0:4].isdigit():
            if date[4] == '-':
                if date[5:7].isdigit():
                    if string.atoi(date[5:7]) < 13:
                        if date[7] == '-':
                            if date[8:].isdigit():
                                if string.atoi(date[8:]) < 31:
                                    return 1
    return 0
```

```
db = MySQLdb.connect(host='localhost', user='root', db='nytoped')
c = db.cursor()

start_date = string.strip(raw_input("Enter start date: "))
if ';' in start_date or '%' in start_date:
    print "WARNING: DANGEROUS INPUT DETECTED."
    sys.exit()
if datecheck(start_date) == 0:
    print "Please try again with a valid date string (YYYY-MM-DD)"
    sys.exit()
end_date = string.strip(raw_input("Enter end date: "))
if ';' in end_date or '%' in end_date:
    print "WARNING: DANGEROUS INPUT DETECTED."
    sys.exit()
if datecheck(end_date) == 0:
    print "Please try again with a valid date string (YYYY-MM-DD)"
    sys.exit()

c.execute('''select WORDS.Word, Sum(Count)
    from (((ARTICLES inner join ARTPAR
                on ARTICLES.ArtID = ARTPAR.ArtID)
            inner join PARWORD on ARTPAR.ParID = PARWORD.ParID)
        inner join WORDS on PARWORD.WordID = WORDS.WordID)
    where ARTICLES.Date >= %s
        and ARTICLES.Date <= %s
        and WORDS.StopCode = 0
    group by WORDS.WordID
    order by Sum(Count) desc limit 10;''', (start_date, end_date))

print "TOP TEN WORDS DETECTED IN TIME SPAN:"
wordTups = c.fetchall()
for tup in wordTups:
    word = tup[0]
    count = tup[1]
    print word.upper()+": "+str(count)+" appearances."
```

Execution

```
gawalt$ python report2.py
Enter start date: 2008-10-27
Enter end date: 2008-11-07
TOP TEN WORDS DETECTED IN TIME SPAN:
OBAMA: 137 appearances.
MCCAIN: 87 appearances.
AMERICAN: 57 appearances.
NEW: 55 appearances.
PRESIDENT: 55 appearances.
CAMPAIGN: 52 appearances.
LIKE: 51 appearances.
PEOPLE: 50 appearances.
MR: 48 appearances.
TIME: 44 appearances.

gawalt$ python report2.py
Enter start date: No.
Please try again with a valid date string (YYYY-MM-DD)
```

Imaging by Joint and Disjoint Word Sets

This script computes a vocabulary Venn Diagram between two authors a list of words used only by Author A, a list of words used only by Author B, and the list of words used by both. It's a somewhat-crude approach to the predictive-framework models where the image of an author are taken to be those known in the training set to be strongly correlated with authorship by A as opposed to B – though in a strongly overfit way.

code

```
import string, sys, os, MySQLdb

db = MySQLdb.connect(host='localhost', user='root', db='nytoped')
c = db.cursor()

author1 = string.strip(raw_input("Enter first author's name: "))
if ';' in author1 or '%' in author1:
    print "WARNING: DANGEROUS INPUT DETECTED."
    sys.exit()
c.execute("select Count(*) from ARTICLES where Author = %s", (author1,))
artCountTup = c.fetchone()
artCount = artCountTup[0]
if artCount == 0:
    print "Sorry, "+author1+" does not appear in our records."
    sys.exit()
```

```
author2 = string.strip(raw_input("Enter second author’s name: "))
if ’;’ in author2 or ’%’ in author2:
    print "WARNING: DANGEROUS INPUT DETECTED."
    sys.exit()
c.execute("select Count(*) from ARTICLES where Author = %s", (author2,))
artCountTup = c.fetchone()
artCount = artCountTup[0]
if artCount == 0:
    print "Sorry, "+author2+" does not appear in our records."
    sys.exit()

numresults = string.strip(raw_input("How many results "+
"would you like to see? "))
if numresults.isdigit() == 0:
    print "Please enter a number next time."
    sys.exit()
numres = string.atoi(numresults)

c.execute(’’’select WORDS.Word, Sum(Count)
    from (((ARTICLES inner join ARTPAR
                on ARTICLES.ArtID = ARTPAR.ArtID)
            inner join PARWORD on ARTPAR.ParID = PARWORD.ParID)
        inner join WORDS on PARWORD.WordID = WORDS.WordID)
    where ARTICLES.Author = "’’’+author1+’’’"
        and WORDS.WordID not in (select distinct PARWORD.WordID
            from ((ARTICLES inner join ARTPAR
                    on ARTICLES.ArtID = ARTPAR.ArtID)
                inner join PARWORD
                on ARTPAR.ParID = PARWORD.ParID)
            where ARTICLES.Author = "’’’+author2+’’’")
    group by WORDS.WordID order by Sum(Count)
    desc limit ’’’+str(numres)+’’’;’’’)
oneNotTwo = c.fetchall()

c.execute(’’’select WORDS.Word, Sum(Count)
    from (((ARTICLES inner join ARTPAR
                on ARTICLES.ArtID = ARTPAR.ArtID)
            inner join PARWORD on ARTPAR.ParID = PARWORD.ParID)
        inner join WORDS on PARWORD.WordID = WORDS.WordID)
    where ARTICLES.Author = "’’’+author2+’’’"
        and WORDS.WordID not in  (select distinct PARWORD.WordID
            from ((ARTICLES inner join ARTPAR
                on ARTICLES.ArtID = ARTPAR.ArtID)
            inner join PARWORD
            on ARTPAR.ParID = PARWORD.ParID)
        where ARTICLES.Author = "'''+author1+'''")
    group by WORDS.WordID order by Sum(Count)
    desc limit '''+str(numres)+''';''')
twoNotOne = c.fetchall()

c.execute('''select WORDS.Word, Sum(Count)
    from (((ARTICLES inner join ARTPAR
                on ARTICLES.ArtID = ARTPAR.ArtID)
            inner join PARWORD on ARTPAR.ParID = PARWORD.ParID)
        inner join WORDS on PARWORD.WordID = WORDS.WordID)
    where ARTICLES.Author = "'''+author2+'''"
        and WORDS.StopCode = 0
        and WORDS.WordID in  (select distinct PARWORD.WordID
            from ((ARTICLES inner join ARTPAR
                    on ARTICLES.ArtID = ARTPAR.ArtID)
                inner join PARWORD
            on ARTPAR.ParID = PARWORD.ParID)
        where ARTICLES.Author = "'''+author1+'''")
    group by WORDS.WordID order by Sum(Count)
    desc limit '''+str(numres)+''';''')
both = c.fetchall()

print author1+" has used the following words, but "+author2+" has not:"
for tup in oneNotTwo:
    word = tup[0]
    count = tup[1]
    print word.upper()+": "+str(count)+" appearances."

print author2+" has used the following words, but "+author1+" has not:"
for tup in twoNotOne:
    word = tup[0]
    count = tup[1]
    print word.upper()+": "+str(count)+" appearances."

print "Both authors have used the words:"
for tup in both:
    word = tup[0]
    count = tup[1]
    print word.upper()+": "+str(count)+" appearances."
```

Execution

```
gawalt$ python report3.py
Enter first author’s name: Roger Cohen
Enter second author’s name: William Kristol
How many results would you like to see? 5
Roger Cohen has used the following words, but William Kristol has not:
CULTURE: 12 appearances.
IRAN: 12 appearances.
FAZLIN: 11 appearances.
BETTER: 11 appearances.
DEBT: 8 appearances.

William Kristol has used the following words, but Roger Cohen has not:
LIBERALS: 11 appearances.
CONSERVATIVES: 11 appearances.
DOG: 10 appearances.
SHELTER: 8 appearances.
CONGRESS: 7 appearances.

Both authors have used the words:
OBAMA: 42 appearances.
MCCAIN: 31 appearances.
PERCENT: 17 appearances.
DEMOCRATIC: 15 appearances.
NEW: 13 appearances.
```

Appendix B

Additional Details: Feature Selection over the International News Data

B.1 Stop Word List

a	an	been	computer	empty	forty	hers
about	and	before	con	enough	found	herself
above	another	beforehand	could	etc	four	him
across	any	behind	couldnt	even	from	himself
after	anyhow	being	cry	ever	front	his
afterwards	anyone	below	de	every	full	how
again	anything	beside	describe	everyone	further	however
against	anyway	besides	detail	everything	get	hundred
all	anywhere	between	do	everywhere	give	i
almost	are	beyond	done	except	go	ie
alone	around	bill	down	few	had	if
along	as	both	due	fifteen	has	in
already	at	bottom	during	fill	hasnt	inc
also	back	but	each	find	have	indeed
although	be	by	eg	fire	he	interest
always	became	call	eight	first	hence	into
am	because	can	either	five	her	is
among	become	cannot	eleven	for	here	it
amongst	becomes	cant	else	former	hereby	its
amount	becoming	co	elsewhere	formerly	herein	itself

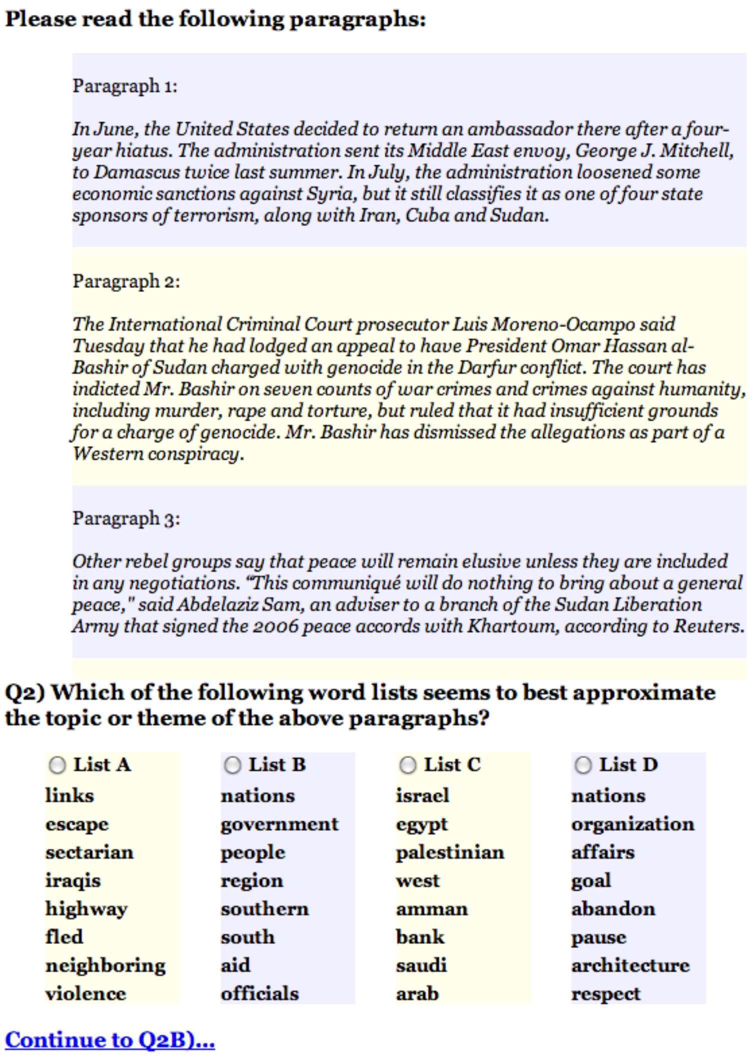
Table B.1: Stop words dropped from consideration (‘A’ through ‘L’)

made	noone	please	somewhere	through	whence
many	nor	put	still	throughout	whenever
may	not	rather	such	thus	where
me	nothing	re	system	to	whereas
meanwhile	now	same	take	together	whereby
might	nowhere	see	ten	too	wherein
mill	of	seem	than	top	whereupon
mine	off	seemed	that	toward	wherever
more	often	seeming	the	towards	whether
moreover	on	seems	their	twelve	which
most	once	serious	them	twenty	while
mostly	one	several	themselves	two	who
move	only	she	then	un	whoever
much	onto	should	there	under	whole
must	or	show	thereafter	until	whom
my	other	side	thereby	up	whose
myself	others	since	therefore	upon	why
name	otherwise	sincere	therein	us	will
namely	our	six	these	very	with
neither	ours	sixty	they	via	within
never	ourselves	so	thick	was	without
nevertheless	out	some	thin	we	would
next	over	somehow	third	well	yet
nine	own	someone	this	were	you
no	part	something	those	what	your
nobody	per	sometime	though	whatever	yours
none	perhaps	sometimes	three	when	yourself

Table B.2: Stop words dropped from consideration (‘M’ through ‘Z’)

B.2 Survey Format

The survey questions as presented to our volunteer respondents:



Please read the following paragraphs:

Paragraph 1:

In June, the United States decided to return an ambassador there after a four-year hiatus. The administration sent its Middle East envoy, George J. Mitchell, to Damascus twice last summer. In July, the administration loosened some economic sanctions against Syria, but it still classifies it as one of four state sponsors of terrorism, along with Iran, Cuba and Sudan.

Paragraph 2:

The International Criminal Court prosecutor Luis Moreno-Ocampo said Tuesday that he had lodged an appeal to have President Omar Hassan al-Bashir of Sudan charged with genocide in the Darfur conflict. The court has indicted Mr. Bashir on seven counts of war crimes and crimes against humanity, including murder, rape and torture, but ruled that it had insufficient grounds for a charge of genocide. Mr. Bashir has dismissed the allegations as part of a Western conspiracy.

Paragraph 3:

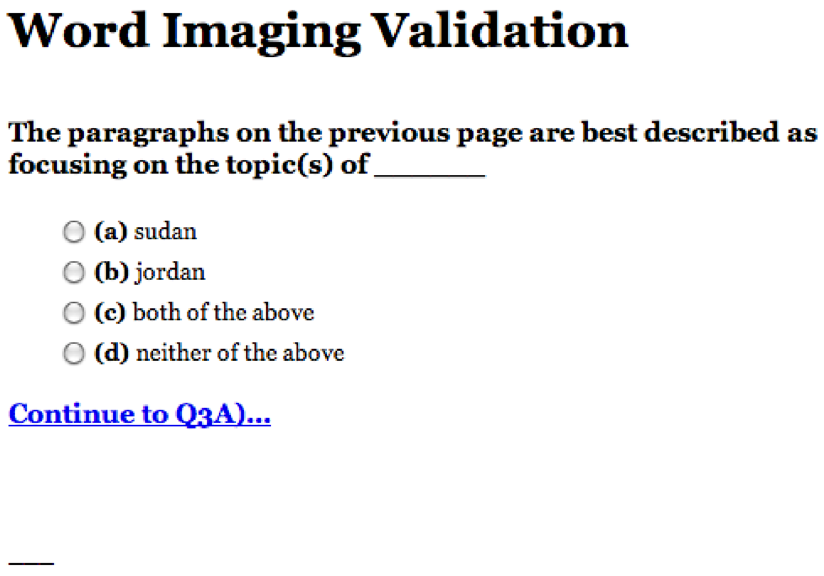
Other rebel groups say that peace will remain elusive unless they are included in any negotiations. "This communiqué will do nothing to bring about a general peace," said Abdelaziz Sam, an adviser to a branch of the Sudan Liberation Army that signed the 2006 peace accords with Khartoum, according to Reuters.

Q2) Which of the following word lists seems to best approximate the topic or theme of the above paragraphs?

○ List A	○ List B	○ List C	○ List D
links	nations	israel	nations
escape	government	egypt	organization
sectarian	people	palestinian	affairs
iraqis	region	west	goal
highway	southern	amman	abandon
fled	south	bank	pause
neighboring	aid	saudi	architecture
violence	officials	arab	respect

Continue to Q2B)...

Figure B.1: Word list selection survey item. Users chose between a set of document units pertaining to a target country q, followed by (in random order): a word image of target q (here, Sudan) using feature selection algorithm x, a word image of target q using feature selection algorithm y, a word list of a *different* target country r (here, Jordan) using feature selection algorithm x, and a word list of target y using feature selection algorithm x.



Word Imaging Validation

The paragraphs on the previous page are best described as focusing on the topic(s) of ______

- ○ **(a)** sudan
- ○ **(b)** jordan
- ○ **(c)** both of the above
- ○ **(d)** neither of the above

Continue to Q3A)...

Figure B.2: Immediately after making a selection as in Fig. B.1, subjects were asked to pick out what target country the preceding paragraphs had in common, from a list containing the actual target q, and the decoy target r.

B.3 Queries and L1LR Imaging Results

Presented here is a collection of the target countries queried for the feature selection summarization summary, and the word images produced by the L1LR method.

Query	*Image*
afghanistan	kabul, troops, pakistan, nato, taliban, obama, iraq american, afghan, war, karzai, forces, military, states united, soldiers, president, government
algeria	tunisia, mali, maghreb, libya, arctic, mauritania, ship french, qaeda, russia, group, officials, country, countries militants, oil, morocco, captain
angola	luanda, cameroon, dos, congo, africa, benedict, oil nigeria, african, mrs, war, visit, foreign, clinton angolas, like, civil, mr, said,
argentina	kirchner, brazil, ecuador, buenos, chile, bishop, flu mexico, cup, venezuela, swine, countries, world, latin states, president, aires, united, country
australia	sydney, zealand, australias, melbourne, australian, chinese, britain china, rio, states, japan, south, countries, tinto year, said, officials, military, president, american, mr
azerbaijan	baku, armenia, azeri, soviet, georgia, moussavi, caspian region, station, pipeline, russia, europe, turkey, gas oil, republic, iran, government, town, said
bahrain	manama, fleet, fifth, bahrains, shiite, jewish, gulf king, region, arab, iran, small, jews, navy united, based, said, states, government, mr
bolivia	evo, paz, lithium, venezuela, ecuador, peru, september cocaine, morales, latin, uranium, argentina, governments, s brazil, nicaragua, report, percent, new, people
brazil	paulo, silva, da, rio, countries, latin, brazilian india, france, brazils, oil, china, world, argentina united, air, states, america, president, new, said

Table B.3: Set of queries examined (A through B), along with their L1LR word images

Query	*Image*
cambodia	phnom, khmer, thailand, vietnam, philippines, penh, countries laos, ambassador, tribunal, myanmar, courts, years, corruption research, mr, said
canada	ottawa, canadian, harper, flu, mexico, trade, britain ban, states, italy, united, france, health, countries border, obama, germany, swine, oil, city, said
china	chinese, beijing, chinas, states, russia, countries, global north, united, korea, india, year, economic, world japan, mr, said
columbia	vancouver, university, district, journalism, british, scholar, states studied, dr, global, masters, court, million, professor new, chinese, york, government, taliban, said
cuba	bay, cuban, embargo, castro, guantnamo, travel, latin venezuela, states, cubans, policy, relations, obama, havana island, administration, american, president, cubanamericans, restrictions,
egypt	mubarak, arabia, jordan, egyptian, cairo, gaza, arab israel, hamas, smuggling, border, iran, world, international muslim, ceasefire, president, middle, flu, killed
ethiopia	mohamed, somalia, africa, sudan, union, born, troops dr, somali, leave, million, country, war, countries government, kenya, mr, said
france	germany, britain, french, air, nicolas, sarkozy, spain paris, european, flight, italy, nato, united, countries europe, president, said, government
georgia	tbilisi, ossetia, saakashvili, russia, nato, russias, august war, ukraine, georgias, russian, moscow, abkhazia, georgian soviet, biden, south, military, states, gas
germany	merkel, france, german, britain, berlin, italy, chancellor europe, european, countries, russia, states, poland, east percent, world, war, united, mr, government, said

Table B.4: Set of queries examined (C through G), along with their L1LR word images

Query	*Image*
haiti	portauprince, haitians, haitian, gonaves, clinton, danner, country nations, ban, poorest, united, school, aristide, haitis city, work, help, said, mr
honduras	zelaya, tegucigalpa, facto, coup, organization, return, zelayas states, crisis, political, ousted, president, american, hemisphere micheletti, honduran, country, world, mr
india	mumbai, indias, pakistan, indian, china, countries, delhi pakistani, pakistans, global, world, new, dalai, united like, states, american, president, said, mr
iran	tehran, iranian, nuclear, irans, sanctions, ahmadinejad, obama syria, uranium, united, administration, israel, states, program russia, iraq, north
iraq	baghdad, iraqi, war, afghanistan, american, troops, bush oil, military, invasion, security, forces, violence, united iran, states, said
israel	hamas, palestinians, gaza, israeli, palestinian, peace, rockets arab, jerusalem, iran, netanyahu, jewish, settlements, egypt united, state, israels
japan	tokyo, japanese, korea, japans, china, economy, united global, countries, party, percent, south, states, democratic economic, meeting, russia, military, president, said, mr
jordan	amman, benedict, egypt, arabia, west, israel, islamic palestinian, jerusalem, leaders, bank, pope, arab, visit king, saudi, iran, trip, muslim, said, government
kenya	nairobi, mombasa, embassies, kenyan, tanzania, africa, pirates somalia, violence, somalias, people, african, clinton, american uganda, nigeria, election, said
kuwait	qatar, invasion, iraq, arab, gulf, husseins, bahrain gaza, city, baghdad, arabia, american, iran, political parliament, failed, women, said, states, officials, government
lebanon	beirut, hezbollah, syria, lebanese, israel, armenians, irans southern, war, gaza, political, hariri, israeli, palestinian elections, army, united, military, said, mr, government
libya	qaddafi, italy, prisoner, libyan, megrahi, scottish, return release, migrants, tripoli, transfer, decision, libyas, oil straw, nuclear, britain, almegrahi, african, lockerbie, british

Table B.5: Set of queries examined (H through L), along with their L1LR word images

Query	*Image*
mexico	mexican, city, drug, flu, caldern, health, states cases, travel, mexicos, countries, border, obama, reported violence, president, state, new, said, military, mr
nigeria	nigerian, nigerias, africa, kenya, angola, muslim, countries oil, senegal, violence, niger, killed, worlds, ghana region, western, like, new, mr, said
pakistan	islamabad, india, afghanistan, pakistani, taliban, attacks, holbrooke qaeda, zardari, militants, pakistans, peshawar, officials, american militant, intelligence, united, obama
palestine	liberation, zionists, jews, israel, jewish, organization, court state, iraq, bank, west, palestinian, hamas, israeli arab, fatah, settlements, like, officials, mr, government
russia	georgia, moscow, ukraine, russian, putin, russias, europe china, united, gas, iran, european, nuclear, states soviet, nato, medvedev, countries, government, said
somalia	mogadishu, islamist, pirates, coast, piracy, shabab, somali ships, al, central, government, islamic, nations, yemen kenya, ethiopian, united, aid, ahmed, said, mr
sudan	khartoum, darfur, southern, chad, sudanese, court, peace bashir, region, albashir, nations, south, arab, president policy, aid, crimes, united, groups, war, mr
switzerland	davos, swiss, tax, polanski, italy, germany, zurich countries, britain, states, international, united, accounts, year government, said, mr
syria	damascus, syrian, lebanon, hariri, iran, turkey, iraq administration, israel, egypt, hamas, states, peace, arab syrias, reactor, talks, hezbollah, saudi, east, said

Table B.6: Set of queries examined (M through S), along with their L1LR word images

Query	*Image*
taiwan	taipei, typhoon, ma, mainland, china, taiwanese, taiwans relations, island, chen, beijing, hong, chinas, military chinese, arms, civil, president, said
turkey	armenia, turkish, turkeys, istanbul, muslim, erdogan, syria genocide, kurdish, countries, european, pipeline, israel, eu country, talks, minister, gas, relations, government, said
ukraine	kiev, gas, russia, georgia, europe, yushchenko, dispute russian, soviet, membership, price, gazprom, nato, european supplies, ukraines, countries, said, military, mr
vatican	lombardi, benedict, pope, bishops, federico, bishop, society church, rome, reforms, israel, jews, pius, catholic council, popes, holocaust, cardinal, people, government, mr
venezuela	hugo, caracas, chvez, bolivia, colombia, latin, cuba oil, venezuelan, ecuador, energy, countries, posada, billion russia, states, companies, relations, president, like, said
yemen	sana, aden, somalia, saudi, muhammad, yemenis, qaeda gulf, detainees, yemeni, tribal, south, al, country officials, southern, american, guantnamo, official, president, mr
zimbabwe	harare, mugabe, mugabes, robert, africa, cholera, human bennett, tsvangirai, aid, rights, zimbabwes, political, south african, minister, nations, crisis, government, american, said

Table B.7: Set of queries examined (T through Z), along with their L1LR word images

Appendix C

Sparse PCA Implementation

The following is an implementation of sparse principal component analysis, used in calculating the principal components of the cancer tweet dataset. Centering of the data is applied. This implementation uses a technique of iterative hard thresholding as an approximation to sPCA's semi-definite program.[5]

The input file `cancer_mat.txt` which encodes the sparse Salton matrix in a [row, column, element value] format.

```
MAX_ITER = 10000000

import numpy, sys, operator

from math import sqrt

from scipy.sparse import lil_matrix
from numpy.random import rand
from scipy.linalg import norm
from scipy import sparse

from numpy import ones
from numpy import dot

def proj(v):
    return v.copy()/norm(v)

def trunc(v,k):
    vals = []
    avals = []
    inds = []
    mn = 0
    pos = -1
    for a in xrange(k):
        avals.append(abs(float(v[a])))
        vals.append(float(v[a]))
        inds.append(a)
    mn = min(avals)
    pos = avals.index(mn)
    for a in xrange(k,len(v)):
        val = float(v[a])
        if abs(val) > mn:
            vals[pos] = val
            avals[pos] = abs(val)
            inds[pos] = a
            mn = min(avals)
            pos = avals.index(mn)
    out = numpy.zeros((len(v),1))
    for a in xrange(k):
        out[inds[a]] = vals[a]
    return out

infilename = '../data/cancer_mat.txt'
r = 2
k = 16
cfile = open('../res/cancer_comp.txt', 'w')
pfile = open('../res/cancer_proj.txt', 'w')

print "Loading matrix..."
V = []
I = []
J = []
infile = open(infilename, 'r')
for line in infile.xreadlines():
    spline = line.split('\t')
    #V.append(float(spline[2]))
    V.append(1)
    I.append(int(spline[0])-1)
    J.append(int(spline[1])-1)

m = max(I)+1
n = max(J)+1
print "Building matrix..."
X = sparse.coo_matrix((V,(I,J)),shape=(m,n)).tocsc()
```

```
#mu = X.sum(axis=0)/m

del I, J, V
comps = []
projs = []
for a in xrange(r):
    p = proj(numpy.matrix(numpy.random.randn(m,1)))
    q = proj(numpy.matrix(numpy.random.randn(n,1)))

    print "Finding component", a+1
    diff = 1
    it = 0
    while diff > 1e-10 and it < MAX_ITER:
        oldp = p.copy()
        oldq = q.copy()

        p = proj(X*q - mu*q)
        qu = X.T*p-mu.T*p.sum()
        for comp in comps:
            for key in comp:
                qu[key] = 0  # Remove features active in any other component
        q = proj(trunc(qu,k))

        dp = norm(p-oldp)
        dq = norm(q-oldq)
        diff = max((dp,dq))
        it +=1
    if it >= MAX_ITER:
        print "(Max number of iterations,",MAX_ITER,", reached)"
    projs.append((X*q - mu*q).copy())
    comp = {}
    for a in xrange(n):
        if q[a] != 0:
            comp[a] = float(q[a])
    comps.append(dict(comp))

lexfile = open('../data/cancer_lex.txt', 'r')
lex = []
for line in lexfile.xreadlines():
    spline = line.split('\t')
    lex.append(spline[1])
lexfile.close()
```

```
for a in xrange(r):
    comp = dict(comps[a])
    scomp = sorted(comp.iteritems(), key=operator.itemgetter(1),
      reverse=True)
    for b in xrange(len(scomp)):
        cfile.write(str(a+1)+'\t'+str(scomp[b][0]+1)+'\t'+
         str(lex[scomp[b][0]])+'\t'+str(scomp[b][1])+'\n')
cfile.close()

for a in xrange(m):
    outline = ''
    for b in xrange(r):
        if len(outline) == 0:
            outline = str(float(projs[b][a]))
        else:
            outline = outline + '\t'+str(float(projs[b][a]))
    pfile.write(outline+'\n')
pfile.close()
```

Bibliography

[1] Xavier Amatriain et al. "The wisdom of the few: a collaborative filtering approach based on expert opinions from the web". In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval.* SIGIR '09. Boston, MA, USA: ACM, 2009, pp. 532–539. ISBN: 978-1-60558-483-6. DOI: http://doi.acm.org/10.1145/1571941.1572033. URL: http://doi.acm.org/10.1145/1571941.1572033.

[2] Y. Benjamini and Y. Hochberg. "Controlling the false discovery rate: a practical and powerful approach to multiple testing". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1995), pp. 289–300.

[3] B. Berelson. "Content analysis in communication research." In: (1952).

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3 (2003), pp. 993–1022. ISSN: 1532-4435. DOI: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993.

[5] Thomas Blumensath and Mike E. Davies. "Iterative hard thresholding for compressed sensing". In: *Applied and Computational Harmonic Analysis* 27.3 (2009), pp. 265 – 274. ISSN: 1063-5203. DOI: 10.1016/j.acha.2009.04.002. URL: http://www.sciencedirect.com/science/article/pii/S1063520309000384.

[6] Toine Bogers and Antal van den Bosch. "Comparing and evaluating information retrieval algorithms for news recommendation". In: *Proceedings of the 2007 ACM conference on Recommender systems.* RecSys '07. Minneapolis, MN, USA: ACM, 2007, pp. 141–144. ISBN: 978-1-59593-730–8. DOI: 10.1145/1297231.1297256. URL: http://doi.acm.org/10.1145/1297231.1297256.

[7] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[8] S. Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine Learning* 3.1 (2011), pp. 1–122.

[9] R.P. Branton and J. Dunaway. "Slanted newspaper coverage of immigration: The importance of economics and geography". In: *Policy Studies Journal* 37.2 (2009), pp. 257–273.

[10] L. Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[11] F. Chang et al. “Bigtable: A distributed storage system for structured data”. In: *ACM Transactions on Computer Systems (TOCS)* 26.2 (2008), p. 4.

[12] J. Chang et al. “Reading tea leaves: How humans interpret topic models”. In: *Neural Information Processing Systems (NIPS)*. 2009.

[13] E.F. Codd. “A relational model of data for large shared data banks”. In: *Communications of the ACM* 13.6 (1970), pp. 377–387.

[14] X. Dai et al. “SBA-term: Sparse bilingual association for terms”. In: *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*. IEEE. 2011, pp. 189–192.

[15] A. d’Aspremont et al. “A Direct Formulation of Sparse PCA using Semidefinite Programming”. In: *SIAM Review* 49.3 (2007).

[16] S. Deerwester et al. “Indexing by latent semantic analysis”. In: *Journal of the American Society for Information Science* 41.6 (1990), pp. 391–407. URL: `http://dx.doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9`.

[17] Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. *Safe Feature Elimination in Sparse Supervised Learning*. Tech. rep. UC/EECS-2010-126. EECS Dept., University of California at Berkeley, 2010.

[18] Robert M. Entman. *Projections of Power: Framing News, Public Opinion, and U.S. Foreign Policy*. University of Chicago Press, 2004.

[19] D.E. Farrar and R.R. Glauber. “Multicollinearity in regression analysis: The problem revisited”. In: *The Review of Economics and Statistics* 49.1 (1967), pp. 92–107.

[20] George Forman. “An extensive empirical study of feature selection metrics for text classification”. In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 1289–1305. ISSN: 1532-4435. URL: `http://dl.acm.org/citation.cfm?id=944919.944974`.

[21] Herbert J. Gans. *Democracy and the News*. 198 Madison Avenue, New York, New York 10016: Oxford University Press, 2003.

[22] Xiubo Geng et al. “Query dependent ranking using K-nearest neighbor”. In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR ’08. Singapore, Singapore: ACM, 2008, pp. 115–122. ISBN: 978-1-60558-164-4. DOI: `http://doi.acm.org/10.1145/1390334.1390356`. URL: `http://doi.acm.org/10.1145/1390334.1390356`.

[23] A. Genkin, D.D. Lewis, and D. Madigan. “Large-scale bayesian logistic regression for text categorization”. In: *Technometrics* 49.3 (2007), pp. 291–304.

[24] Matthew Gentzkow and Jesse M. Shapiro. *What Drives Media Slant? Evidence from U.S. Daily Newspapers*. Working Paper 12707. National Bureau of Economic Research, 2006. URL: `http://www.nber.org/papers/w12707`.

[25] D. Gillick and B. Favre. “A scalable global model for summarization”. In: *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing.* Association for Computational Linguistics. 2009, pp. 10–18.

[26] A. Gruber, M. Rosen-Zvi, and Y. Weiss. “Hidden topic Markov models”. In: *Artificial Intelligence and Statistics (AISTATS)* (2007).

[27] B. Gunter. “The quantitative research process”. In: *A handbook of media and communication research: Qualitative and quantitative methodologies* (2002), pp. 209–234.

[28] Thomas Hofmann. “Probabilistic latent semantic indexing”. In: *SIGIR ’99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval.* Berkeley, California, United States: ACM, 1999, pp. 50–57. ISBN: 1-58113-096-1. DOI: `http://doi.acm.org/10.1145/312624.312649`.

[29] Mongoose Global Intelligence. “About Us”. In: (2011). URL: `http://www.mongoosegi.com/about.html`.

[30] Robert Jervis. *Perception and Misperception in International Politics.* Princeton University Press, 1976.

[31] T. Joachims. “Text categorization with support vector machines: Learning with many relevant features”. In: *Machine learning: ECML-98* (1998), pp. 137–142.

[32] I. Jolliffe. *Principal component analysis.* Wiley Online Library, 2005.

[33] S.S. Keerthi and C.J. Lin. “Asymptotic behaviors of support vector machines with Gaussian kernel”. In: *Neural computation* 15.7 (2003), pp. 1667–1689.

[34] Deept Kumar et al. “Algorithms for storytelling”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* KDD ’06. Philadelphia, PA, USA: ACM, 2006, pp. 604–610. ISBN: 1-59593-339-5. DOI: `http://doi.acm.org/10.1145/1150402.1150475`. URL: `http://doi.acm.org/10.1145/1150402.1150475`.

[35] H. Kwak et al. “What is Twitter, a social network or a news media?” In: *Proceedings of the 19th international conference on World wide web.* ACM. 2010, pp. 591–600.

[36] Anukool Lakhina, Mark Crovella, and Christophe Diot. “Diagnosing network-wide traffic anomalies”. In: *SIGCOMM ’04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications.* Portland, Oregon, USA: ACM, 2004, pp. 219–230. ISBN: 1-58113-862-8. DOI: `http://doi.acm.org/10.1145/1015467.1015492`.

[37] L. Lee and S. Chen. “New Methods for Text Categorization Based on a New Feature Selection Method and a New Similarity Measure Between Documents”. In: *LECTURE NOTES IN COMPUTER SCIENCE* 4031 (2006), p. 1280.

[38] J. Martineau and T. Finin. “Delta tfidf: An improved feature space for sentiment analysis”. In: *Proceedings of the 3rd AAAI International Conference on Weblogs and Social Media.* 2009, pp. 258–261.

[39] Q Mei and C Zhai. “Discovering evolutionary theme patterns from text: an exploration of temporal text mining”. In: *KDD ’05*. 2005. ISBN: 1-59593-135-X. DOI: `http://doi.acm.org/10.1145/1081870.1081895`.

[40] E. Michelakis et al. “Filtron: A learning-based anti-spam filter”. In: *PROCEEDINGS OF THE 1ST CONFERENCE ON EMAIL AND ANTI-SPAM. MOUNTAIN*. Citeseer. 2004.

[41] Luke Miratrix et al. *Summarizing large-scale, multiple-document news data: sparse methods and human validation*. Tech. rep. Berkeley, CA 94720: UC Berkeley, May 2011.

[42] Francesco Moretti. *Graphs, Maps, Trees: Abstract Models for a Literary History*. Verso, 2005.

[43] Preslav Nakov, Antonia Popova, and Plamen Mateev. “Weight functions impact on LSA performance”. In: *EuroConference RANLP’2001 (Recent Advances in NLP*. 2001, pp. 187–193.

[44] R Nallapati et al. “Event threading within news topics”. In: *CIKM ’04*. 2004. ISBN: 1-58113-874-1. DOI: `http://doi.acm.org/10.1145/1031171.1031258`.

[45] R.C. North et al. *Content analysis: A handbook with applications for the study of international crisis*. Vol. 184. Northwestern University Press Evanston, IL, 1963.

[46] Bo Pang and Lillian Lee. “Opinion mining and sentiment analysis”. In: *Foundations and Trends in Information Retrieval* 2.12 (2008), pp. 1–135.

[47] Y. Pawitan et al. “False discovery rate, sensitivity and sample size for microarray studies”. In: *Bioinformatics* 21.13 (2005), pp. 3017–3024.

[48] M. Porter. “An Algorithm for Suffix Stripping”. In: *Program* 14.3 (1980), pp. 130–137. URL: `http://ontology.csse.uwa.edu.au/reference/browse_paper.php?pid=233281850`.

[49] Amy E. Potter. “Voodoo, Zombies, and Mermaids: U.S. Newspaper Coverage of Haiti”. In: *Geographical Review* 99.2 (2009), pp. 208–230. ISSN: 1931-0846. DOI: `10.1111/j.1931-0846.2009.tb00427.x`. URL: `http://dx.doi.org/10.1111/j.1931-0846.2009.tb00427.x`.

[50] G. Salton and M. McGill (editors). *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[51] F. Sebastiani. “Machine learning in automated text categorization”. In: *ACM computing surveys (CSUR)* 34.1 (2002), pp. 1–47.

[52] Dafna Shahaf and Carlos Guestrin. “Connecting the dots between news articles”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’10. Washington, DC, USA: ACM, 2010, pp. 623–632. ISBN: 978-1-4503-0055-1. DOI: `http://doi.acm.org/10.1145/1835804.1835884`. URL: `http://doi.acm.org/10.1145/1835804.1835884`.

[53] J.L. Solka. “Text data mining: theory and methods”. In: *Statistics Surveys* 2 (2008), pp. 94–112.

[54] M. Steinbach, G. Karypis, V. Kumar, et al. “A comparison of document clustering techniques”. In: *KDD workshop on text mining*. Vol. 400. Boston. 2000, pp. 525–526.

[55] The Pew Research Center for The People and the Press. “Internet Overtakes Newspapers as News Outlets”. In: (2008). URL: `http://www.people-press.org/files/legacy-pdf/479.pdf`.

[56] A. Thompson. “The News Media and International Relations: Experience and the Media Reality”. In: *Canadian Journal of Communication* 13.1 (1988), pp. 53–54.

[57] R. Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[58] J. Tsitsiklis, D. Bertsekas, and M. Athans. “Distributed asynchronous deterministic and stochastic gradient optimization algorithms”. In: *Automatic Control, IEEE Transactions on* 31.9 (1986), pp. 803–812.

[59] Otto E Wahl, Amy Wood, and Renee Richards. “Newspaper Coverage of Mental Illness: Is It Changing?” In: *Psychiatric Rehabilitation Skills* 6.1 (2002), pp. 9–31. DOI: `10.1080/10973430208408417`. eprint: `http://www.tandfonline.com/doi/pdf/10.1080/10973430208408417`. URL: `http://www.tandfonline.com/doi/abs/10.1080/10973430208408417`.