Topic Analysis and the News

Creation of an RSS feed reader

By

Adam Sanders

Advised by David Walker

Department of Computer Science

Princeton University

Abstract

In this paper, I present a new RSS feed aggregator. This program differs from a majority of the current RSS feed readers in its ability to parse RSS feeds by topic, the use of modern internet technologies to view feeds, and the simplicity with which visualizations can be created from the aggregated information. As a result, this system can be used as a platform to better understand the relationships between different documents in a text corpus. This aggregator also provides an excellent base on which to experiment with information visualization and large text corpuses. I have chosen to analyze news RSS feeds to test the system for a number of reasons. The use of this program for other types of content is discussed further in Future Directions.

Introduction

1.1 Beginnings

It is difficult to overestimate the importance of the news in the modern day. With each step we take toward a more globalized society, we find occurrences in far off lands more likely to affect our lives. Strife in the Middle East has an immediate affect on gas prices in this country, while Political calculations made in the United States have clear effects on nations all around the globe. Perhaps no more germane an example of this globalization can be seen than the credit crisis of 2008 and 2009, where an abundance of easy credit and later plummeting house values combined to create a global financial constriction in a way that would be impossible to understand without the diligence of national and international news organizations.

In addition to traditional news outlets, the exponential growth in the number of Internet users and the adoption of a number of online syndication protocols has allowed for an explosion of news published on the web. The interactivity, immediacy, and limitless space provided by the Internet have rendered it an ideal medium for the publication of news and have brought about unprecedented change to the information industry, fundamentally altering the way many individuals learn about their surroundings. Changes found in the news industry as a result of this increased interest in online content range from differences in revenue models, to a diversification of reader demographics as well as an increasing emphasis on multimedia (Matheson 2004). Perhaps the most obvious to news consumers resulting from the growth of online syndication is a marked increase in the amount of news information available. While traditional forms of content publication like the newspaper often limit news distribution to small and well-defined areas, online content publication can deliver information to users from all over the world. It is likely that we have yet to fully realize some of the long-term implications of this shift for news consumers, however one result that does become clear is that these consumers now have the ability to experience a far greater variety of perspectives than was previously possible. (Robinson 2006).

The potential advantages from gaining these new readers have been by no means lost on content providers. A study published in 1998 found over 2,800 different newspapers published online. More than 1,700 of those newspapers were based in the United States (Peng, Tham and Xiaoming 1999). In the years since that study, the number, form, and readership of online publications have risen dramatically (Sigmund 2008). The attention paid to traditional sources of news finding a presence on the web has even paved the way for the development of new forms of published content. Online blogs whose focus is predominantly or entirely on news and current events have become increasingly popular. Varying remarkably in format, many of these sources blur the line between blog and newspaper. Newspaper/blogs like Think Progress and The Huffington Post exist solely in digital form but have begun to acquire widespread readership (Robinson 2006). During the months between May and November of 2008, the Huffington Post averaged over 4 million unique views per month (Lipsman 2008). This impressive figure represents not only a significant number of blog views, but also a significant portion of news viewership on the web. Thus it becomes clear that online publication will increasingly become part of the way in which Americans, and the entire world, receive information about current events. Given the differences between online and traditional forms of content publication, it may behoove us to consider the potential benefits and drawbacks of an increased reliance on web-based news.

The increasing number of online publications can benefit the news consumer in a number of ways. Consumers given the ability to access stories from different publications may gain an increased appreciation for possible biases or flaws found in news stories or even entire publications. The ability to "cross-examine" information and perspectives from one source with the information from other sources could be an invaluable tool for consumers (Robinson 2006). Using the abundance of news sources could allow the news consumer the ability to peruse news reports local to a given event, rather than relying on the interpretation of that same event by national or even international news organizations. This ability has the potential to illuminate aspects of news events not found in the news coverage of a national news organization. It seems clear that the wealth of information consumers now have at their disposal provides

a great deal of potential for a better informed and even more engaged society. However, these benefits do not arrive without caveats.

This wealth of news information is useless without an effective means of mentally organizing and physically navigating the data. Websites created by news organizations are typically decent at allowing users to peruse information from that particular news source, but are specifically designed to keep consumers from examining news at other organizations. As one might expect, a given news organization will try at all costs to avoid consumers leaving their site for one of their competitors (Peng, Tham and Xiaoming 1999). The result of this phenomenon is that content navigation on the web for a majority of websites his highly introspective. Sensing this introspection, a number of applications and applications have been created in order to bring together content from a number of different sources and display that information with a unified interface. In this paper I describe a system that allows individuals to better navigate news stories as well as to examine the scope of the news in new and more interactive ways. I accomplish this task through the use of state-of-the-art topic analysis tools and leverage new Internet technologies to display this information through an easy-to-use interface. While the creation of a content aggregator is not novel, this system is aimed specifically at addressing the shortfalls found in many of the current aggregators and news navigation software. As a final benefit, this program provides a novel platform on which further study or visualization of the news or other RSS feeds can easily be accomplished.

1.2 An overview of online syndication

While we have mentioned online content syndication, it is difficult to fully comprehend the trajectory of this technology without first examining its past. We first illuminate the circumstances surrounding its inception and the role it served in the growth of the Internet. We next examine the current state of the protocol as well as a number of the programs currently used to interact with this format. This

analysis is aimed at providing context for my content aggregator, as well as motivating the design decisions made in its creation.

1.2.1 A Brief History of Data Syndication

During the early years of the 1990s, a number of software developers became interested in the creation of technologies that bridged the gap between television and the Internet. The main difference concerning these developers lay in the means by which the content arrived at the screen of the user. Televised content was "pushed" from the source to the user. This system provided control over the content for the publisher, as well as relieving the user of the continual responsibility to search for content. On the other hand, the model used in computers and the Internet specifically required users to "pull" content from a web server, and view that content inside of a web browser (Cockburn and Jones 1997). The vision was to create a technology bridging the gap between these two methods capitalizing on the advantages found in each model. The two largest constituents rallying behind this concept were Netscape and Microsoft. Each took a different approach toward solving this problem. Microsoft allowed users of its popular operating system to place snippets of web pages on their desktop. Dubbed "Active Desktop," pages displayed on this system would be continuously updated, and allowed users the ability to monitor a website without the need to manually navigate the site. As an added benefit, websites would find an increased capacity for advertisements pushed directly to the desktop under this system.

Taking a different tact, Netscape instead chose to develop a specification through which websites could provide content to users. This protocol, the "RDF Site Summary" (RDFSS) would allow for the development of novel browser plug-ins able to collect content from several web pages and display that content in new and interesting ways. The specification made this scenario possible by allowing content publishers to release a file annotated with meta-information about web content. Each of these files contained the standard fields one might expect to find in a website: author, publication date, title, content, as well as many more pieces of information that users would find useful. As a result, creators of browser plug-ins would have some assurances about the content they were re-publishing and would not have to

deal with the mess of anonymous paragraph tags, lists and other elements that comprise the typical HTML document.

Taking on the challenge of developing these new browsing applications, Apple developed a web-browser plug-in called Hot Sauce. This plug-in can be seen working inside the Netscape Navigator below in Figure 1. Hot Sauce used a meta-content system similar to RDF Site Summary to browse the web in a 3D "flythrough" interface. While the concept was interesting, the hardware requirements of developing this 3D visualization were beyond the computers of the age (Cockburn and Jones 1997). Apple dropped the project in 1997 (Meta-Content Framework 2005). Netscape's RDF Site Summary survived slightly longer, but was abandoned by the company in 2001.

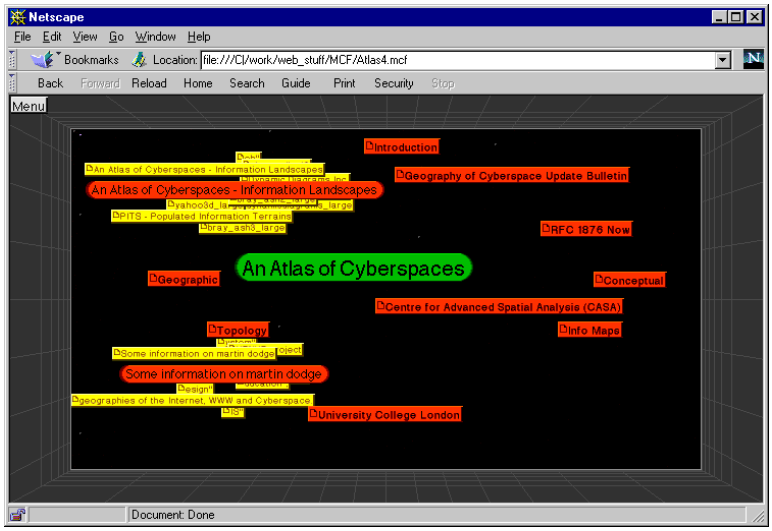

Figure 1: Apple's Hot Sauce

Unlike users of Hot Sauce and Active Desktop, a small community of individuals who had been using RDFSS prior to its abandonment by Netscape felt it to be a useful product. These users adopted the abandoned specification, simplifying the name from RDFSS to Rich Site Summary and modifying the specification through a number of revisions. The format eventually became known colloquially as Really Simple Syndication, reflecting the primary purpose acquired by the technology in its later revisions: online syndicated content publication. Since the abandonment of RDFSS by Netscape in 2001, the use of content syndication files like RSS and its relatives (Atom) have become an important means of

distributing text content on the Internet. The success of this format for the dissemination of text has even led to the creation of the syndicated audio and video formats known as pod casts. The widespread adoption of these technologies has allowed numerous websites to easily provide content to their users. However, these files would be useless without the effective means to navigate and display them. Providing the means for users to interact with these files is clearly an important aspect of this use. A number of different applications have been developed to aid individuals in navigating the myriad of RSS and Atom feeds that are available on the Internet.

1.2.2 NetNewsWire

The programs used to collect and display RSS feeds have become known as feed aggregators, and function primarily to download and present content from several RSS files in a single location with a unified interface. There are both web-based and stand-alone programs, each of which provides a unique array of features. Some will offer the ability to create an account in order to sync news stories across multiple sources, while others allow for the creation of a single news feed out of many individual news feeds. Unfortunately, a majority of these news aggregators suffer from some of the same difficulties presented by the vast number of newspapers currently available in website form. An example of this difficulty is the requirement that articles be browsed according to their source. While this is beneficial in many contexts, it does little to address the difficulties presented by access to an increasingly large amount of online news content.

Examining one stand-alone web aggregator in particular, NetNewsWire[1], we can illustrate these shortcomings. This feed reader presents data separated in folders, each of which represents a single news source. Clicking on an individual folder brings one to the list of articles published by that source sorted by the date of their publication. Clicking on an individual news story will display the summary text and title of that news story, as well as provide appropriate links to the story on the actual news source website.

[1] http://www.newsgator.com/individuals/netnewswire/

The interface of this program functions similar to an email client like Thunderbird and the user interaction in this program is not dissimilar to reading traditional print publications like a newspaper. Continuing the analogy, each individual folder can be viewed as different print publication, while the stories in these folders are the different news topics for the day. Browsing through news in this format bears a close resemblance to wading through a stack of newspapers laid out on a table. One would likely read all or some subset of stories in each individual paper before turning to the next paper. However, this method of reading the news does not make a great deal of sense given the high degree of duplication found in most major news publications. For example, on the day that Obama won the election, it is likely that almost every single newspaper published some commentary on the event. Reading through traditional newspapers for these stories would require the user to wade through a number of unrelated articles in each source before coming across the particular article or articles of interest. This same search would have to be reproduced for users of NetNewsWire in almost the same fashion as users of traditional print publications. In this sense, NetNewsWire and similar types of RSS feed readers fail to take advantage of the numerous differences between print and computerized content.

The result is that in both of these formats, the ability to navigate news stories by their subject matter is lacking. One might suggest that this task can be accomplished by simply looking more specifically into the different RSS feed topics presented by each news source. It is true that most news sources do provide a different RSS feed for each of a number of topics. Standard topics might include politics, business, international news, or even more specific subjects such as horticulture or sports. Even with these demarcations, it can be a challenge to isolate related stories between different news sources. This problem stems from the fact that many stories fail to fall neatly within any particular topic. Every news article has the potential to be categorized differently by each news source. In addition, there exist topics in the news that generally relate to multiple topics. For example, some recent and very important news at the time of this writing is information on the auto bailout in Washington. Stories pertaining to different aspects of this bailout might occur in a number of different sections: Politics, Business, Cars, National News, and even International News. Individuals subscribing to any subset of these different

news topics would find themselves to be underserved by the model of news provided by NetNewsWire and similar feed aggregators.

1.2.2 Google News

It should be noted that not all RSS readers are missing the features lacking above. One important exception is the web-based Google News[2]. The initial concept of this aggregator was to develop a metric similar to Google's famous PageRank formula, but modified appropriately to categorize news stories (Bharat 2006). This aggregator provides a large list of the features you might expect from any news aggregator, but also includes a number of additional niceties. By scanning an assortment of old news sources, Google News actually allows the user to browse news sources predating RSS feeds. Users can search through the data from these sources and seamlessly transition from current stories to these older scanned sources. Google also allows users the ability to view this information on a timeline. However, the feature of most interest to this discussion is that Google News allows users to search for articles by topic from a large library of potential sources. When reading any story in the newsreader, Google provides the ability to examine similar stories published by other news sources. This is precisely the functionality lacking in NetNewsWire and many other RSS feed readers. Google pulls this information from over 4,000 sources; though the company has kept the exact list of sources used a secret (Jarboe 2007). In fact, the main difficulty with the Google News reader is that so much of the inner workings are not publicly known. The algorithms used to find and index similar stories, the exact sources Google uses to search other sites, and many more things are both controlled entirely by Google.

While many of these design decisions are inconsequential for news users, they do become a problem when trying to develop new visualizations. Because a great deal of information about these news stories is accessed exclusively by Google, providing the consumer with visualizations based partially or even completely on the relationships between documents is impossible. In addition to the lack

[2] http://news.google.com/

of access to the raw data behind the algorithms, users of Google News have little control over the sources found in the related articles, and no knowledge of how those sources and their articles are selected. Google provides no means by which a user could modify the specificity of the article selection. For instance, if a user wanted not only stories on an individual piece of news, but also news stories tangentially related to the original, there is little that user can do within Google News to accomplish this task.

1.3 Challenges of text-based visualization

While a great deal of attention has been paid to analyzing text sources, relatively less attention has been paid to making that data easier to visualize and understand (Wise, et al. 1995). Few online publishers provide visual means by which to navigate and understand their often vast collections of information. While a great deal of attention will be paid to the shortcomings of newspaper website design in a later section, it seems worthwhile to speak more generally about the difficulties in visualizing text.

The fundamental process of data visualization is the creation of spatial representations of inherently non-visual information. As an example, the mapping of a series of numbers onto a bar-graph accomplishes this task by giving the inherently non-visual magnitude of these numbers a spatial representation: the height of the bars. This mapping of abstract information to a physical representation reduces the mental load necessary to understand the data (Wong, Whitney and Thomas 2000). This can be accomplished because our brains have evolved to cope with spatial information. As a result we are generally very efficient at decoding this type of information. On the other hand, the comprehension of numbers and symbols had fewer evolutionary pressures, and as a result requires more mental effort. The result is that we are much better at dealing with recognizable elements of the world, than we are at dealing with symbols. The point of this discussion being that information visualization, when well done, takes advantage of our innate ability to grasp and interact with spatial information and allows humans to gain a better appreciation for previously difficult-to-comprehend data.

Unfortunately, the realm of text visualization is more challenging than the example just mentioned. The difficulty with creating visualizations of text stem largely from the fact that most text is inherently non-spatial (Rohrer, et al. 1998). That is to say that it contains no intuitive mapping onto the visual plane. While the magnitude of words easily maps to the size of a bar in a bar chart, there is no consensus measure for the magnitude of a word. There seems to be no obvious way to visualize strictly textual information. Fortunately, the information used in this project does contain some spatial information that will aid this process. The news articles I set out to visualize contain several pieces of meta-information that can be directly utilized in data visualization. For example, publishing dates can readily be understood along a timeline, while authors and sources can be used to construct trees and graphs. These and other examples permit a greater deal of flexibility and expression for the creation of spatial representations of the news.

There have been a number of precedents aimed at creating visualizations from large bodies of text. By virtue of its size and importance to our daily lives, the Internet has often been the target of such efforts. By 1997, there were several companies actively working on novel ways by which to navigate the complexities of the Internet at least one of which we already mentioned: Apple's Hot Sauce. These projects typically involved the creation of some type of graph in which nodes represented documents and edges represented hyperlinks. On a small scale, these visualizations were an attempt to endow the user with a mental model of the website that would both be simple to remember and improve the ability of the customer to navigate the site effectively. Unfortunately, these solutions did not weather the dramatic increases in complexity brought about by the evolution of the Internet. At the time of their creation in the mid to late 90s, the number of hyperlinks on the average website remained relatively small. These graph-based visualizations functioned well for small websites with limited link structures. It is not uncommon in modern websites for a single page to contain several-hundred clickable. These increases in complexity resulted in indistinguishably complex graph representations. To compound matters, the markup language used to create websites provides no intrinsic information about the importance of one hyperlink relative to the others. That is to say that HTML contains no method of differentiating between important hyperlinks

(for example main navigation hyperlinks) from any other type of link. The use of these graph-based or other visualization programs to browse the modern web would likely lead to more rather than less complex web browsing. This is because those who created the website would be far better equipped to create a meaningful navigation system than would a naive computer algorithm, making a majority of attempts to create a general purpose visual web browser unnecessary and even counterproductive.

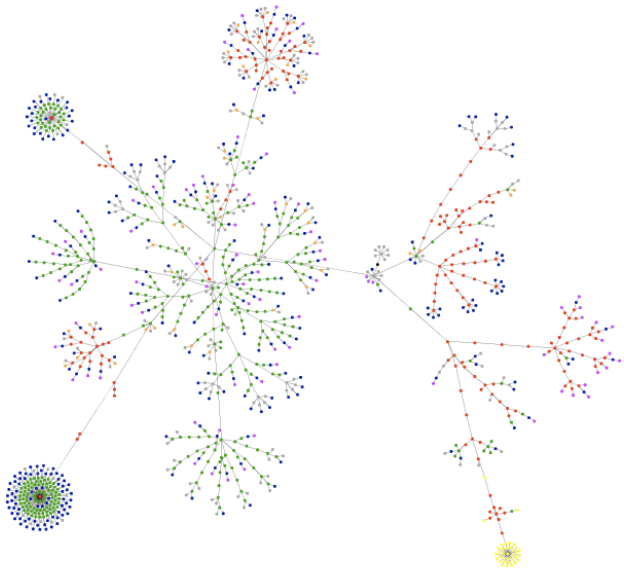

Figure 2: HTML Structure of Amazon.com

A general theme in the criticism of previous web visualization work stems from the unstructured nature of the Internet. Creation meaningful and intuitive visualizations of this amorphous collection of tags is challenging as there are few rules as to what can occur in a given webpage. Fortunately, many of these criticisms do not apply to the visualization of highly structured content like RSS feeds. With the RSS specification, and those syndication protocols that resemble it, there are guarantees for the location of the title, content, and many other points of information. This format even introduces a nice hierarchy that can be used in further visualizations. Unlike the web, which is a large directed graph, an RSS XML feed is guaranteed to be a tree. At the top level, we find the source website, for example cnn.com. Under this main heading there are a number of different RSS feeds, like politics, business, or all news. Finally, we have an array of articles underneath each of these sub-headings. This required structure is identical

for every RSS feed and allow for a greater degree of freedom in creating useful ways to navigate and visualize information stored in these files.

1.4 My Project

Given the limitations of the previous work in this area, and need to develop more efficient means for individuals to interact with news information, I created my own news aggregator. The main goals of this project were to create an aggregator capable of sorting articles by their topics, and presenting those stories in an interface that was both easy to navigate and visually pleasing, while addressing the difficulties found in previous works. The secondary goals of the project were to create a platform for the development of news visualizations. This platform will allow for the rapid development of news visualizations.

We begin by describing the look and feel of the system that users see online today. With a grasp on the finished product, the explanation of its evolution will be easier to grasp. The visual design decisions made in the creation of the news browser were made with simplicity and efficiency in mind. We then address the technical evolution of this system. We examine first the decisions made on which news sources to cover initially. These decisions outlined, we define the processes by which they are collected and stored as well as the interfaces developed for their retrieval. The next phase in the design, information analysis, is covered in depth at this point. Once we have examined the technical means by which these stories are analyzed, we turn to their final storage in a medium suitable for the news browser website. A final point of the project is the development of visualizations that allow individuals to better understand the news.

Visual Interface

The massive amounts of information available on the Internet demanded the creation of a system that was both robust and reliable. On the other hand, this reliability had to be balanced with the simplicity necessary for any user to easily navigate the site. In order to create the best possible interface, we first examined the current interfaces for news information found on the Internet, specifically those of major news publications. While thousands of news organizations have developed a web presence, I intend to focus specifically on newspapers. This decision was made primarily because these sources serve as the largest outlet of online news. In addition, many of the navigation issues and design problems with these sites can be generalized to the other forms of publication found online. Through this analysis I show the motivations behind a number of the design decisions made in the final web browser.

2.1 Newspapers on the web

Traditional means of navigating news sources give consumers very little choice as to the format in which they see stories and articles. The authors of any publication are responsible for defining all the possible parameters: article locations, section headings, even the number of articles available in a given publication. These restrictions are placed on paper by necessity; it would be cost-prohibitive to print personalized newspapers for each individual. As a result, news organizations attempt to create the best decisions possible in order to satisfy the largest majority of their readers. During the transitioning to the new format of online content publication, the majority of newspapers simply replicated the format used in their print publications. This duplication is apparent in the example of the New York Times website shown in Figure 3: The New York Times below. This figure demonstrates the transition from the printed version of the publication on left, to the modern web format on the right. A quick glance at these images reveals the majority of the components of the printed version were perfectly maintained when the switch to the digital version was made. In both formats, the user is presented with a large image, or series of images on the main page, as well as accompanying snippets of text representing article summaries. Those

interested in reading further about the articles in the printed version can follow the information at the bottom of the article- a function accomplished in the web format by hyperlinks. As a result, interaction with this web form of the newspaper is left largely unchanged when compared to the printed format. The continuity in format found in this publication is by no means restricted to the New York Times. A majority of the major newspapers have maintained most elements of their printed formats in a similar fashion.

Figure 3: The New York Times

While the familiarity associated with these components is important, the duplication of this format fails to take advantage of the substantial differences between printed formats and computerized content. Those who developed these newspaper websites failed to utilize a large degree of potential for greater interaction with their users and the flexibility afforded by their new format. It is important to note that there have been a handful of news organizations that have begun to utilize some of this wealth of underutilized potential. For example, many websites have created commenting systems allowing users to post their thoughts about articles found online. These systems add a degree of interactivity to the site taking advantage of the flexibility afforded by this new medium. Despite this point, a majority of these websites still fail to take advantage of important advances in Internet technology[3].

[3] Shortly before this paper was written the New York Times did release just such an application. This prototype, dubbed the New York Times news "skimmer," provides the user with an entirely AJAX interface to the NYT's collection of daily news. (http://prototype.nytimes.com/gst/articleSkimmer/).

2.2 Browser Design

There were several guiding principals when I sat down to develop this project. The first of these was simplicity. It was imperative that this project be easy to understand for anyone who was interested in its use. Additionally, this system should allow users to get at the news in which they are interested without requiring them to wade through unrelated articles. Given the difficulties faced when dealing with a large number of sources, another goal in this project was to provide an interface through which an individual could browse a number of sources at once. Finally, the news browser should leverage visualization as a means to convey information while reducing the mental strain necessary to read all that news.

Figure 4: News Browser

2.3.1 Look and Feel

The looks of the browser were developed to provide a simple and clean interface. The HTML and CSS of the browser are both standards compliant, and the look of the finished product is compatible with the three major browsers: Internet Explorer, Safari, and Firefox[4]. All navigation is accomplished using AJAX, and no further page loads are required. There are a number of interface features that improve the use of the site: tabs indicate the current page you are viewing, arrows aid this process showing the direction the content box will move when the button is clicked. All news articles in this project are presented in a standardized format, modeled to provide the simplest possible interface.

[4] Internet Explorer 8, Safari 3 and 4.0 beta, and Firefox 3.0. Google Chrome was not tested at the time of writing.

The moving transitions on the site accomplished using the very popular Scriptaculous JavaScript library[5] as well as Prototype JavaScript Framework[6] for its AJAX functionalities. The combination of these two libraries allows objects to fly around the screen smoothly and without interruption. The result is uninterrupted news in an interface that is enjoyable to use. Each individual top button will drop down a menu of further options. This menu does not inhibit your reading the article, as the current news article simply drops down with the menu. Once you are finished with the drop-down menu, one can simply click the button again to remove it from the screen.

2.3.2 Navigation

The navigation bar on top has a sequence of links: corpus, documents, topics, visualization, about, and search. The first of these options allows users to select the list of sources they would like to browse. This list can be a user-defined entity. For example, some users might prefer to browse only blog titles about politics, while others would like to browse only major news publications' sports pages, still others might enjoy the option of browsing all possible sources for all possible kinds of information. Each of these sets of news RSS feeds is updated regularly, so that those interested can always find the exact sources they are looking for. It should be noted that this flexibility is an advantage over Google News, whose algorithms do not allow one to specify the sources from which related articles can be culled.

The topics tab allows the user to browse through the 100 most relevant topics of the day. Each topic is made of a representation of roughly 100 words, 8 of which are shown in the topic browser, and 10 of which appear above any article. Users can scroll through the available topics, viewing subsections of the list 10 at a time. This navigation is accomplished seamlessly through AJAX. Each topic has a colored circle as a header. The color of this circle represents the strength of the topic. This strength is a measure of the fit for the top ten articles within that topic to the topic words themselves. Clicking on any individual topic will load the highest related article for that topic in the topic browser.

---

[5] http://script.aculo.us/

[6] http://www.prototypejs.org/

The Documents tab allows users to view those documents listed under the current topic. When the browser first loads, this value is set to topic 1. Articles listed in this tab are sorted by their relevance to the given topic, though this particular sorting metric was intentionally developed to be very easy to change. An alternate form of the browser used a different distance metric, allowing users to browse not by articles related to the current topic, but rather by articles related specifically to this article. This was later changed for the purpose of continuity between the topic browser and the document browser. Clicking on any individual article will load the article in the reading window. This window provides the title of the article, topic most closely related to the article, source, and feed, as well as the summary text of the article. Clicking on the article title redirects the browser to the actual source of the article, allowing those interested in reading further about any given article, the ability to do so rapidly. In addition, users who become interested in visiting the source can do so simply by clicking the source URL. This link will redirect the user to the source RSS feed of the article, for example: CNN National News.

The news visualization tab shows a list of visualizations on the current dataset. Clicking the link can access these visualizations. They load inside the drop-down menu, allowing the user to continue reading an article while simultaneously examining a visualization of that same article. There is a great deal more to be said about these visualizations in section 5.1, and I will leave the remainder of that discussion until that point. Clicking the About document tab brings up a brief description of the project as well as contact information should the user feel the need to learn more information about the project. Finally, the search tab allows users to input a number of search terms. These terms are then matched against all possible topic words, and a list of the most relevant topics is returned.

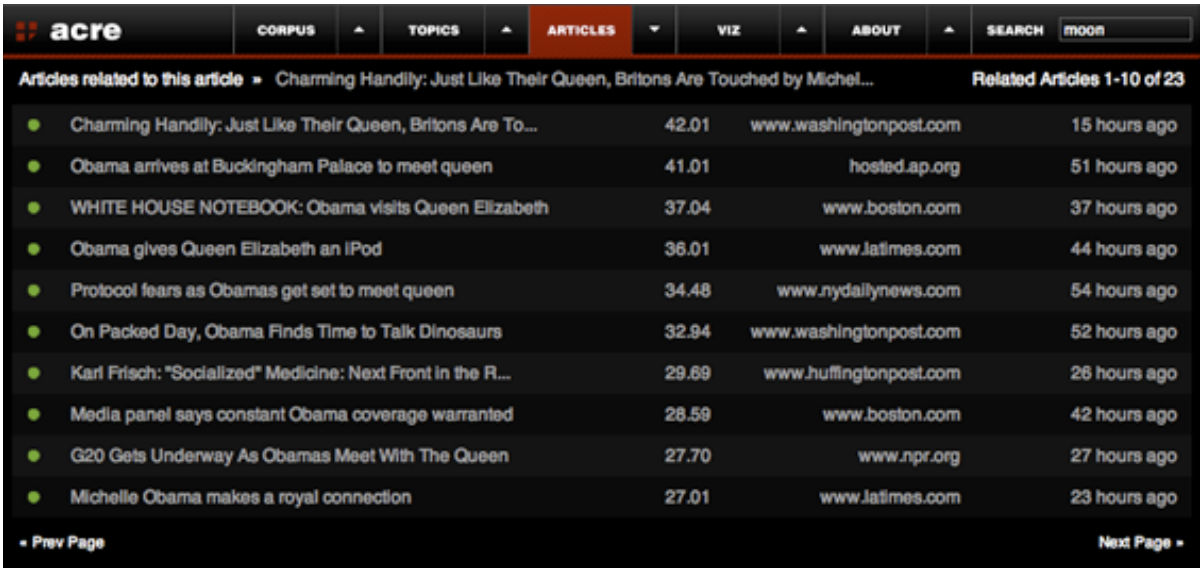

Figure 5: Articles Tab

2.3.4 Topic Browsing

The ability of this browser to navigate by topic is one of the most important aspects of its user interface, but the question of exactly how well this analysis functions is certainly important. Perusing a sample of the 100 topics collected and sorted each day reveals an algorithm very competent at accomplishing this task. An example topic can be seen in the image in Figure 5 above. We find the first topic for this particular day involved Barack Obama's first meeting with the Queen of England. All the articles under this topic on the first page are related to this story. The strength of these relationships is also indicated by the row of green circles, representing articles that match well to a given topic. The user is given information about article precedence in the form of a publishing time, which allows this user to rapidly determine which article represents the most recent coverage of a given story. Finally, it is important to note that this particular story is well covered by a large number of different news organizations. This coverage is represented in the news browser with seven different publications listed on the first page. Users interested in examining the coverage of each of these sources would be able to rapidly search through each of these publications and to compare the conclusions drawn by each

Data Collection

Finding the appropriate sources to use for this project and reliably filling a database with new articles from those sources proved to be a far more difficult proposition than originally anticipated. Obstacles that held up progress in this area stemmed from a number of places, from complications with improperly formatted sources, to unanticipated difficulties with various data tools. This aspect of the project presented more than its share of challenges.

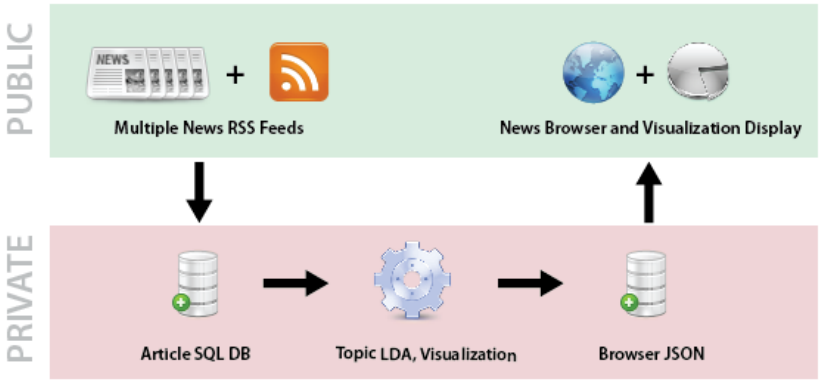

Figure 6: Project Outline

3.1 Choosing the Sources

Selecting the different news sources to be used in the trial run of this project proved surprisingly difficult. My first impulse was to create a web spider that crawled the web in search of RSS feeds to add to a database. This program, written in Python, took in an address, and then attempted to find other websites of a different domain from that original address. Once these new domains had been downloaded, the program then searched for either links with RSS in the title, or RSS feeds. The volume of RSS feeds found by this simple program was impressive. Running for several minutes from Reddit.com and Digg.com produced hundreds of RSS feeds. Unfortunately, a majority of these feeds represented less than reputable news organizations. To compound matters, it was difficult to determine which feeds were from reasonable news organizations, and which feeds were not. Undeterred, I modified

this program to take in a single webpage and find all the RSS feeds from this page. All that was necessary with this new program was to find the news sources in which I had interest, and point the Python script at that URL. The reason for the use of a Python script to accomplish this task was that many of the news organizations hosted between 50 and 100 different RSS feeds. Manually copying these feeds down for ten or more news organizations would have been an arduous and time-consuming process, especially if one wished to gather a collection of all the news feeds for any given site. However, the results of this approach were also dissatisfactory. While many of these news sources hosted a large number of news feeds, a majority of these feeds represent topics in which I had little interest: fashion, sports, gardening, yoga. These feeds were far from conserved across major news publications, making their inclusion problematic. For these and other reasons it became clear that the best way to collect the sources for this project would be to manually choose a subset of these RSS feeds for each news publisher. In order to make these selections, I began by defining the criteria that would be useful for rating possible news publications. Several aspects of each potential source were important. Each source was to have a sufficiently high rate of publication. This requirement was important to ensuring the news in my system was expansive and up-to-date. In addition, the sources with high rates of publication were more likely to cover all the news stories in each news cycle, allowing for a greater likelihood of article overlap between sources - a feature useful in comparing the stories from several news organizations on the same topic. A second criterion was that the news sources have specific sub-feeds for as many of the following topics as possible: Political News, National News, World News, and Business News. This is a criterion that was done primarily for testing purposes. These article listings would become very useful to determine the accuracy of any topic analysis algorithms used later in the project.

With these criteria in mind, I was able to gather a list of 13 major news publications. Each of these publications had a reasonably high rate of publication, covered a subset or the entire set of topics in which I was interested. In addition to the major traditional news publications, I also was able to collect RSS feeds on a number of new news sources. The complete listing of these sources used in this project can be seen in Figure 2 below. Later several other news sources were added to the project, and can be

found online at the project website. Adding or removing articles from these databases is an extremely simple process.

Figure 7: News Sources

While selecting some of the largest news organizations did prove advantageous in terms of the number of articles I was able to collect and the relative trust I had in each of those stories, these RSS feeds did not come without disadvantages. The major complication with these major news publications was the lack of full-text RSS feeds. Downloading the RSS information from all but one or two of these sources meant downloading just the summary text. This text, typically only a few sentences long, is only a short description of the article. This would later prove to be an important decision for the use of different topic analysis algorithms, as many of these algorithms are designed to work with large bodies of text for analysis. While the question of whether or not this decision would be advantageous for topic analysis was unclear, it did allow for the storage of many more articles. As a majority of the articles listings in the database did not contain full-text, the database was able to hold a great many more articles. With reasonably sized articles, the size of each article entry in the database would increase between three and five-fold if one were required to store all of the article text.

3.2 Storing the Data

3.2.1 Parsing RSS Feeds

In order to collect the information from this set of RSS feeds, I created a program in Java that would interface with both a database for storing information about the different RSS feeds, as well as a special library that was capable of parsing all the major variations of RSS feeds[7]. The importance of having a library actually serve as the interface with the RSS feeds comes as a result of the sheer number of online syndication protocols, each differing slightly in format. Were I to develop my own RSS feed parser, it would be necessary to develop components of that parser capable of dealing with each different kind of syndication protocol family, and all the versions of that family. Rather than developing an RSS feed parser for each individual type of RSS feed, I chose to utilize this library.

Once the RSS feed had been downloaded, it became necessary to find a subsection of the possible meta-information that would be important to keep. At first blush, there was an abundance of information for every article in an RSS feed. Examining the specification for the different syndication protocols, one might expect to see many of the potential fields completed on the web. Most formats allowed for the publisher to specify the author, locations, images, summary text, full text, and many more interesting pieces of data. Unfortunately, a majority of these fields are left empty by even the largest professional news publications. Further complicating this process, the specific subset of fields used by each publication also differed. While CNN might provide an image with most or all of their stories, FOX might provide none at all. As a final hindrance, the format of each field for one publication often conflicted with the format used by another publication. For example, the New York Times completes the source link field with a hyperlink to their logo in GIF format. This method, linking to an appropriate logo element, was the anticipated use of the field. On the other hand, Associated Press links straight to its homepage, an HTML document. These inaccuracies reduce the amount of meta-information the program can reliably extract from an RSS feed to only a small subset of the possible information. After an

[7] https://rome.dev.java.net

exhaustive search, I was able to determine the fields that were almost always completed in a cross-compatible manner, narrowing the large list of potential data points to a mere ten.

3.2.2 Designing an SQL Database

In order to store these data points for later analysis, they are stored in an SQL database. Given the original format of the content is already a form of database, one might point out that this method of storing seems unnecessary. However, the use of SQL allowed for a much greater deal of flexibility in the manipulation and examination of the source data than would an XML file. In addition, this information can be more easily accessed by a number of client programs in an SQL database than would be possible using an XML file, as the former is designed for exactly that purpose. Perhaps the most important component of the choice to use an SQL database was the ease with which one could query the data and extract points of interest. This query language allows one to rapidly search through date ranges for articles from a specific source, or collection of sources. This operation, and those like it, was integral to many of the data analysis components of my project about which I will speak in the next section.

The specific flavor of SQL I chose for this project was SQLite 3. This lightweight SQL implementation works by creating databases as file objects on the operating system; removing the requirement held by some other SQL flavors of hosting a server to access the database. This proved to be a strikingly important feature when developing applications to work with the database. The ability to rapidly delete, create, and move databases is useful for debugging while applications are routinely damaging databases. After this process of database development, it might eventually make sense to switch to a more solid SQL variant. This is a task further discussed in the future directions of this paper. The end result of database development was a single SQL file containing two tables. The first of these tables, Sources, contained all the necessary information about a specific RSS feed. This information included the title of the feed, a short description, the URL of the feed, the domain found in that URL, a link to the homepage of the feed, and finally an image representing the feed. The image element of the source was included optimistically, though it is unlikely that it will ever see use in my applications as a

result of the formatting problems mentioned earlier. The second table, Articles, contains the permanent article URL, the URL of the feed from which this article was collected, the title of the article, a brief description, and the date. The creation statements for all of these tables can be seen in Figure 8 below.

```
CREATE TABLE Articles(arturl PRIMARY_KEY, srcurl, title, date DATE, desc);
CREATE TABLE Sources(srcurl PRIMARY_KEY, domain, title, desc, link, image);
```

Figure 8: Create statements for the SQL database

3.2.3 Keeping the database updated

Clearly an important aspect of this project was the ability to quickly and reliably update the information stored in this database. As the news on a given day is constantly evolving, it would be necessary to keep the content on the news browser website constantly updating. In order to keep this information updated, I created a Java program that takes as a parameter the location or potential location of a database. If this file does not exist, the database is created in that location with all the correct fields mentioned above. This feature allows for one to rapidly develop new RSS feed storing databases, a component that I will speak further about in future discussions. The program then searches that database for any feed URLs. Finding any of these URLs, the program than begins to download all the stories that are not currently in the database. The image in Figure 9 shows a run of this program in verbose mode. Each new article added to the database will print the title of that article, while articles discovered that are already in the database will output a period. Simply watching this program run is very similar to watching a news ticker. When set to run every few hours, it can become obvious when there are periods of increased news activity. Days on which a great deal of news is generated will result in a greater number of articles added to the database.

```
Added article: Studies fail to settle prostate testing debate
.
Added article: Southern group to campaign for poor

Added article: US prison officials will ease restrictions on 'American Taliban' Lindh
....
Added article: Spain's SEAT workers vote for pay freeze
....
Added article: UK soldier killed in Northern Ireland laid to rest

Added article: Palestinians adjourn Cairo talks without government deal
```

Figure 9: Example output from Update.java

A final step to the process of keeping this file updated was to find some means of automating the update process. In order to accomplish this, I wrote a crontab file that originally ran the Update.java code every three hours. However, problems began to arise as the database grew larger and the runs for each portion of this program respectively slower. As more programs began to work with this database, some overlap began to occur, as two processes worked with the database at the same time. On occasion, two of these programs would end up touching the database at the same time, locking the database. Alternatively, one program might encounter an error as a result of a badly formatted RSS feed, and be unable to safely close its connection to the database. One might presume that this problem would be trivial to solve: simply unlock the database and continue. Unfortunately there is no simple way to unlock these databases in some instances without restarting Apache, or rebooting the machine. As this system was running on a server to which I did not have root access, these two options were out of the question. Thus it became necessary to find some alternative method of unlocking these databases. In order to solve this problem, I created a simple script that creates a copy of the given database. Even if the original database was locked, this new copy will be unlocked. Appending this to any code that attempts to use a database, we can always be assured that our program will have access to an unlocked version of the database.

```
#!/bin/bash

echo '.dump' | sqlite3 news.db > news.dump; cat news.dump | sqlite3 news2.db
rm -rf news.db
mv news2.db news.db
```

Figure 10: Unlocking Script

3.2.4 Database size

The project outlined above is completely automated and with the 13 original sources is able to download between 1000 and 3000 articles per day. This figure adds rapidly, and after running for only 15 days the article database contains over 50,000 articles. For the purposes of comparison, the very popular AP news corpus contains 2248 news articles. I have provided my database of news articles online at the project website in SQL format[8]. Those interested in working with large contemporary article databases can easily download this file and will hopefully put it to good use. While the size of this database does provide a number of advantages in terms of completeness, it is clear that allowing this database to update continuously will eventually result in a prohibitively large database. As a result, one future direction this project might consider is creating a time limit for articles to remain in the database. This limit can be imposed based on the target size of the database. As an example, roughly 15 days worth of news articles represents an SQL file size of 50MB. A reasonable idea might be to store just 100MB of this data, and delete all articles aged over 30 days. An alternative solution would be to modify the sources collected in this database to save only articles from a specific source. This step would allow for a dramatic reduction in the size of the database, while still keeping a large number of stories from any given time period in the database. As it is unclear at the time of writing the exact purposes I have planned for this database, I am currently keeping all articles.

[8] http://www.princeton.edu/~acsander/school/news/data/sql/news.db

Data Analysis

With the large database of news articles, there still remained the problem of how to analyze those articles and prepare them to be presented on the website. An important component of this analysis was the development of a system that organized these articles on their relatedness to other articles. Once these relationships had been mapped, all that remained was to find a suitable format in which to store the final set of information for the browser. The first of these problems was taken care of by Latent Dirichlet Allocation (LDA).

4.1 Topic Analysis

The goal of topic analysis is to develop shorter representations of the members of a collection of documents. These short representations can improve efficiency and allow for the processing of large data collections all while preserving the essential statistical relationships that are useful for document classification, summarization, similarity and relevance assignments (Blei, Ng and Jordan 2003). A great deal of progress has been made in the area of Information Retrieval (IR). The methodology most used in this field is to reduce each document in the text corpus to a vector of real numbers, representing the ratio of the number of occurrences of each word in the corpus. From the list of total words used in the corpus, a small representative set of terms is selected using the term frequency-inverse document frequency (tf-idf) weights. The use of the inverse document frequency is an important addition that diminishes the weight of terms that occur very frequently in the entire collection of documents (Landauer, Foltz and Laham 1998). The rationale behind this method is that words like “the” and other frequently used terms do not convey a great deal of meaning to the document, while words that occur less commonly or only in a smaller subset of the documents are important. The end result of this process is a matrix of terms by documents. An example of this term by document matrix can be found in Figure 11 below. Each numerical value in this matrix is actually the tf-idf value for the sample document p_n and term $word_m$.

	p1	p2	p3	p4	p5	p6	.....	pn
word 1	-0.0120	-0.0104	0.0040	0.0032	-0.0036	-0.0025		-0.0051
word 2	-0.0103	-0.0021	0.0007	-0.0168	-0.0004	-0.0144		-0.0177
word 3	-0.0072	-0.0054	-0.0068	-0.0093	-0.0129	0.0109		0.0166
word 4	-0.0047	0.0184	0.0041	0.0186	-0.0161	0.0130		-0.0198
word 5	-0.0085	0.0033	-0.0081	-0.0028	0.0140	-0.0069		0.0070
word 6	-0.0073	0.0130	-0.0064	0.0084	0.0199	-0.0106		0.0011
⋮								
word n	-0.0035	-0.0006	-0.0074	-0.0073	-0.0166	0.0168		0.0015

Figure 11: Term-by-document matrix

This matrix reduces documents of arbitrary length to fixed lists of numbers, increasing the speed with which documents can be compared. However, this particular methodology is incomplete for a variety of reasons. The most important of these for the purposes of topic creation is that the model does not provide any direct information on inter or intra-document relationships. The standard model presented above does an excellent job at producing documents for search queries, but does little to segregate documents into reasonable collections that can be browsed. It is in light of this problem that several more complex models were developed that took into account the concept of a "topic."

The model used in this project is Latent Dirichlet Allocation, created in part by Professor David Blei. LDA is a three-level hierarchical Bayesian model in which each document are represented as random mixtures over latent topics. Each of these topics is characterized by a distribution over words. The Final result of the calculations in this model is two matrices. The first of these is a document by topic matrix, representing the likelihood of any given document to be associated with a given topic. The second of these matrices is a topic by term matrix. This second matrix represents the probability of any given word being associated with a given topic. An advantage of this model is that these two matrices can be used on novel data, allowing researchers to classify documents not found in the original training set. For the purposes of this project, this would allow us to develop these matrices over a large time period using large datasets, and simply update the current news stories as they come along.

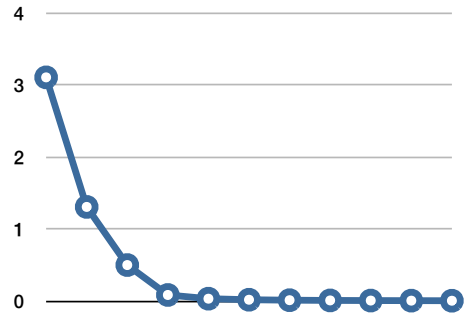

Figure 12: LDA clustering coefficient over 100 iterations on real news data

The implementation of this program provided by Blei calls for just a single input file[9]. Each line in the file represents a single document, while the items on that line represent the unique ID of a word found in that document, as well as the frequency of occurrence of that document. In order to create this file, it was necessary to develop a program that queried the database for articles from a given time period and a specific set of sources. The ability to filter sources was an important addition to my program as all the articles were stored in a single large database.

4.2 JSON File Creation

The question of how best to represent the collected data to the browser was a challenging one. The first question that required an answer was the format the browser was to read. There were several options to consider before making a final choice on this issue. The first of these was XML. This format is becoming increasingly popular as JavaScript has many native methods for manipulating and reading from XML files. However, there are certain problems with the format. This format requires a great number of potentially long XML tags, adding to the overall size of any large XML file. This becomes especially problematic when there are many small items to store. In this case, the overhead of creating a tag for each individual item, for example a hyperlink, becomes almost as large as the hyperlink itself. As

[9] http://www.cs.princeton.edu/~blei/lda-c/

all these files would have to be transferred over the Internet to the client's computer, these inefficiencies could easily result in longer load times and potentially loading failure as the file-size grew.

An alternative to the XML file format often used in these projects is JavaScript Object Notation (JSON). This language has a smaller grammar and maps far more directly onto the data structures used in modern programming languages. The largest upshot of this comparison is that in Python, the language used for this portion of the project, it is extremely simple to export a JSON file. Any python dictionary can be directly converted into a JSON file, without the need to work extensively with the DOM building appropriate data structures. The file created in Python can then be read directly into JavaScript as a variable, without the need for any additional processing. The simplicity of this operation allows the changes made to the Python programs immediately appear in the JavaScript browser, without the need for additional modification of either program.

In order to most efficiently update the browser with different articles, a distributed file creation process was utilized. Keeping efficiency in mind, it was important to develop a file specification that would be appropriate for potentially very large datasets. The situation this process was attempting to avoid was having the news browser open all of the articles in the database at once. As most browsers running JavaScript limit the amount of memory that can be used by that language, this option had the potential to cause serious problems. Instead it was decided to break the information gained from the LDA analysis into two segments: global information, and article-specific information. Every piece of article specific information could then be stored in it's own file. These local files would only be opened when the information they held was needed. The global file, on the other hand, would be opened when the user opened the news browser, and then this file would be maintained until that user closed the browser. The actual specifications for these files are appended to the end of this document.

As a final step in this process, it became necessary to transfer the files from my original server running at Princeton, to the remote server that hosted my website. In order to accomplish this, I wrote a short script that automatically compressed all the data for a given LDA run into one file. This file was then uploaded automatically to the remote server using another short script written in Python. The remote

server ran a cron script that checks every ten minutes for any new files uploaded to the server. If a new file is located, the script unpacks that file and over-writes the previous JSON files on which the browser runs. An added benefit of this process is that it allows the website to continuously show news content, even while the new JSON files are being created.

4.4 Time Efficiency

These steps work together to create the files required by my browser to display news information, but they are certainly not cheap. The entire process from selecting the articles from the database to uploading the completed product to my web server takes between five and six hours for the roughly 1,500 articles served each day. The time required to complete this process occurs in part because LDA and the other components of this analysis must be completed from scratch for each new corpus, and there are six of these different corpi total. Running this process on just the largest corpus (news-all) takes between two and three total hours.

There are a number of reasons for this high running time. With regards to LDA, this program simply requires a great number of calculations. In order to reduce the number of calculations, one could reduce the number of topic clustering iterations. Unfortunately this reduction would necessarily result in a decrease in topic quality. There are other areas in which this project could be improved. One in particular relates to the database creation. When I originally devised this database, I created as the primary key of my Articles database, the permanent article URL. This was done because this URL was certain to be unique, as well as to clearly identify the article. While this decision worked well for identification purposes, pulling an item from the database requires SQL to search through the list of primary keys and select the key that matches. Continually sorting this database in order to perform lookups begins to take a greater amount of time as the database grows. The result is that operations with a large database begin to take up a considerable amount of time. To make matters worse, the size of a URL as compared to just a numeric index is very large. Many URLs in the database also contain the title

of the article as can be seen below in Figure 13. The result of these difficulties is that a surprising amount of time in each run is taken up by performing SQL queries on the database. This could easily be remedied by creating a new column in the Articles table called the Article ID. This would serve as the primary key, and would improve the efficiency of this project.

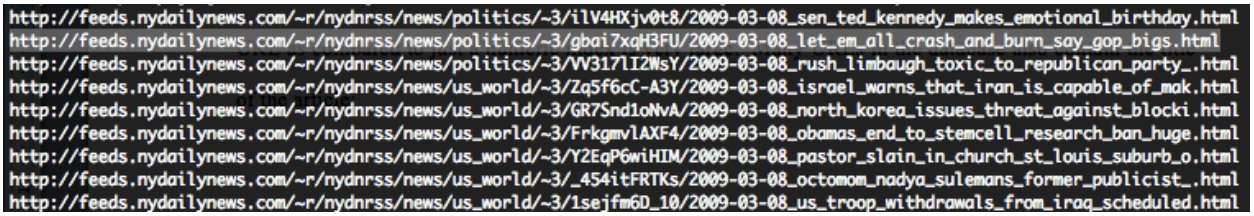

Figure 13: Example set of article URLs

Visualization

A final stage in this project was the development of meaningful visualizations for the news information being downloaded. There were only a handful of different visualizations created for this project. However, the most important point to take from these visualizations is that information required to create these visualizations is publicly available on the project website. Those interested in working with the data in this project can simply download it as well as the specification for that data, and rapidly create visualizations of their own. Thanks to the flexibility purposely built into the browser, these visualizations can be incorporated into the site at any time.

5.1 Stream Graph

The first visualization I created was a stream graph. This form of chart is useful for displaying the relative amounts of an item over time. The purpose of this graph to this project was an attempt to show users the shape of the news. Specifically, this graph would show the introduction and lifespan of a single news story. By graphing the prevalence of topics over the period of time, individuals reading the news can get a better understanding of exactly how a story is covered. One might imagine that some stories break onto the scene and very quickly remove themselves, while others get mention by one news source 24 hours before the remainder of the news sources pick them up. Any number of these possible differentiations is possible in our news cycle, and the ability to really understand how exactly a news item plays out in the media could be a very useful tool to consumers.

The graph in Figure 13 below demonstrates the result of this work. The graph is arranged with time across the x-axis, and the relative prevalence of topics across the y-axis. The prevalence figure represents the aggregate probability of all articles published in this time period being a member of a given topic. It is important to keep in mind that strictly speaking, topics do not represent individual news items. Rather, they represent a collection of words that are indicative of those news items. As we are comparing the aggregate score for those words over all the articles, at each point in this graph the probability of a

topic is non-zero. That said, it is interesting to see that each topic typically has a small span of importance. Typically these topic probabilities remain as thin lines until the story breaks, at which point they take over a considerably larger amount of the news cycle. In the particular graph seen below, we can also see the importance of the weekend, as this chart was made over a three-day span. The area of the chart where the three topics (1, 2, and 3) begin to get greater mention is a Monday. As the volume of news produced on Sunday is typically small, this point in the graph can be seen as the news organizations reporting on information that occurred over the weekend. This coverage then trails off. On the other hand, Topic 5 becomes important later, and is clearly a story that broke on Tuesday of that same week.

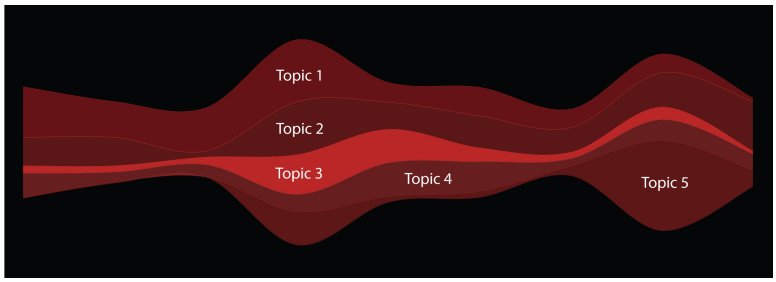

Figure 14: Stream Graph for March 29th - 31st 2009

An interesting extension of this assignment would be to take this stream graph for a three-day span, and create a topic analysis for a much longer period of time. One can imagine examining the shape of the news over a several month period, or even an entire year. While the visualization I have created is capable of doing such an analysis, this visualization requires LDA output to run. As each 3-day span typically contains between 1,000 and 3,000 articles, the number of articles found in a single month would contain close to 100,000 articles. Attempting to do topic analysis on this large dataset would require an incredible amount of computing power to accomplish in any reasonable speed.

5.2 Force Directed Graph

The next visualization found in my project was a force directed graph. This graph has been seen many times before in the examples of previous attempts at textual visualization. The graph seen below is a representation of an individual topic. Each of these nodes represents an article, while the edges represent relationships over a certain threshold. This threshold is set currently such that all nodes are connected. In order to create this graph, a second Python program had to be written that would parse and re-package the JSON data from a given topic into a format more conducive to graph generation. The actual algorithm on which the graph runs is based on energy minimization. Each of these nodes repels all other nodes, while each edge can be seen as a spring. The graph will slowly make its way towards the least energetic state by separating edges as best it can. It is important to note that Matt Sanders, an Undergraduate at Princeton University, did a majority of the work on this specific visualization. I merely adopted his work for use in my particular project.

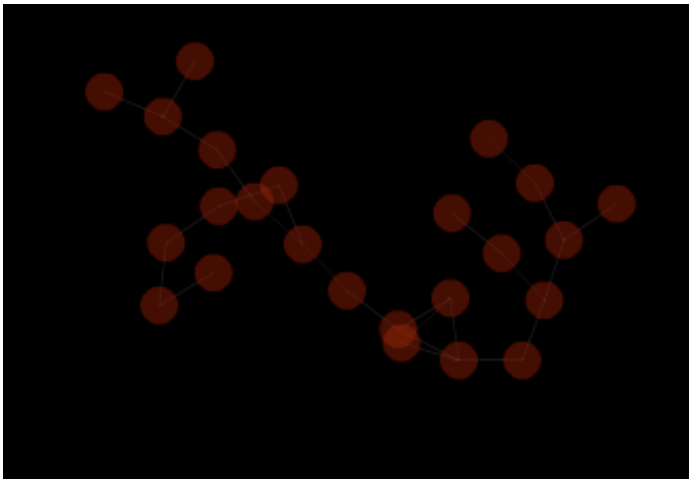

Figure 15: Force Directed Graph

5.3 Visualization Efficiency

One of the largest problems I discovered in this project was that it was difficult to create visualizations on demand. As there were 100 topics and thousands of articles in each run, creating force-directed graphs or other forms of visualizations for all of these different combinations proved largely impractical. This was especially true in light of the surprisingly large amount of time that was required to simply create the data on which these visualizations were run. As a result of these difficulties, these visualizations are not yet fully incorporated into the site. A future direction I am currently investigating is the creation of these images on-demand. This model makes more sense, as it seems unlikely that a majority of the possible visualizations will never be accessed. More importantly, this will allow for a much greater number of visualizations, as the static files will not need to be stored on the drive.

Conclusion

The results of this project were largely positive. I was able to create a system that automated the process of collecting and analyzing news stories from any RSS feed. Building on this accomplishment, I created a browser capable of navigating this enormous dataset efficiently with an interface that is visually pleasing. I also provided the tools for those interested in doing further research into data visualization and more specifically news information. All of these components provided their own challenges, and each contained its own specific benefits.

It is my belief that the end result of this project is more than the sum of its parts. This system provides a looking glass through which individuals can rapidly examine the viewpoints of all different sides to a story via the application of topic analysis. This accomplishment allows individuals powers of media scrutiny previously reserved for those who spend a great deal of time scouring a large number of news publications. At some small level, this system should allow media consumers a path to a more balanced viewpoint on the world. In regards to the browser design, the aesthetics of this project are likely to go overlooked. However, they are an integral component of the project. Creating a website that was as visually pleasing as it was informative will hopefully result in more users, and users who spend more time on the site. With each click, these consumers can learn more about the events in locations from Princeton to Pyongyang and all the places in between. Finally, the creation of visualizations for this project and the development of a platform that will allow yet further research into this important art will hopefully lead to a better informed population and increased interest in the problems presented by the creation of spatial representations of text.

These accomplishments are really only a few steps at the foot of the mountain of work that exists in the field of data interaction. The rapid increase in the amount of news one has available is really only a small subsection of the dramatic increase of data available to an individual in general. With the Internet at the fingertips of billions, information has begun to flow online from all over the world. This information comes in the form of scientific data, published papers, the musings of bloggers, novels,

photographs, and sound files. The potential good that can be accomplished with all of this information is restricted only by our ability to utilize it effectively. These abilities are not innate; rather they must be developed either on an individual level or through the creation of programs similar to this project. Only by putting in the work on better means to understand the data we have, will we be able to realize it and our own full potential.

Works Cited

Bharat, Krishna. *And now, News.* 1 23, 2006. http://googleblog.blogspot.com/2006/01/and-now-news.html (accessed 3 28, 2009).

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." Edited by John Lafferty. *Journal of Machine Learning Research* 3 (1 2003): 993-1022.

Cockburn, Andy, and Steve Jones. "Design Issues for World Wide Web Navigation Visualization." Report, Department of Computer Science, University of Canterbury, Christchurch, 1997.

Godbole, Namrata, Manjunath Srinivasaiah, and Steven Skiena. "Large-Scale Sentiment Analysis for News and Blogs." Google Inc., New York, 2006.

Hearst, Marti A. "Untangling Text Data Mining." School of Information Management & Systems, University of California, Berkeley.

Jarboe, Greg. *Revealing the Sources of Google News.* 5 31, 2007. http://searchengineland.com/revealing-the-sources-of-google-news-11353 (accessed 3 28, 2009).

Landauer, Thomas K., Peter W. Foltz, and Darrel Laham. "Introduction to Latent Semantic Analysis." *Discourse Processes* 25 (1998): 259-284.

Lasica, J. D. "Blogs and Journalism Need Each Other." (Nieman Reports) 9 2003: 70-74.

Lipsman, Andrew. *Huffington Post and Politico Lead Wave of Explosive Growth at Independent Political Blogs and News Sites this Election Season.* 2008. http://www.comscore.com/press/release.asp?press=2525 (accessed 3 28, 2009).

Matheson, Donald. "Weblogs and the Epistemology of the News: Some Trends in Online Journalism." (New Media & Society) 6 (2004): 443-468.

Peng, Foo Yeuh, Naphtali Irene Tham, and Hao Xiaoming. "Trends in Online Newspapers: A look at the US Web." *Newspaper Research Journal* 20, no. 2 (3 1999): 52-62.

Robinson, Susan. "The mission of the j-blog: Recapturing journalistic authority online." *Journalism* 7, no. 65 (2006).

Rohrer, Randall M., David S. Ebert, John L.Rohrer, Randall M. Sibert, David S. Ebert, and John L. Sibert. "The Shape of Shakespeare: Visualizing Text Using Implicit Surfaces." Department of EECS, George Washington University, 1998.

Sigmund, Jeff. *Online Newspaper Viewership Reaches Record in 2007.* 1 28, 2008. http://www.naa.org/PressCenter/SearchPressReleases/2008/Online-Newspaper-Viewership.aspx (accessed 3 28, 2009).

The Meta-Content Format/Framework. 2005. http://downlode.org/Etext/MCF/ (accessed 3 28, 2009).

Wise, James A., et al. "Visualizing the Non-Visual: Spatial analysis and interaction with information from text documents." *Proceedings on Information Visualization*, 3 1995: 51-58.

Wong, Pak Chung, Paul Whitney, and Jim Thomas. "Visualizing Association Rules for Text Mining." *Pacific Northwest National Laboratory*, 2000.

Xu, Zhichen, Yun Fu, Jianchang Mao, and Difu Su. "Towards the Semantic Web: Collaborative Tag Suggestions." Yahoo! Inc, Santa Clara, 2006.