MEDIA MONITORING USING SOCIAL NETWORKS

Submitted by:
Wayne Chu (100280442)

Honours Project
COMP 4905
Carleton University

Supervised by:
Dr. Tony White
School of Computer Science

April 8, 2005

ABSTRACT

With the rapid rise in the number of weblogs, or blogs, on the World Wide Web, there is a growing need to be able to quickly search for discussion on specific topics. While keyword searches using tools such as Google or Technorati can yield some useful results, we run into the problem of having to enter contextualizing keywords to filter out unwanted and irrelevant search results. This has the unfortunate consequence of make the search process more complicated and possibly filtering out search hits that we would want. This paper outlines an approach to narrow search results to only relevant hits, while allowing for general keyword queries. Since the blogosphere constitutes a social network, the solution, BlogCrawler, attempts to use the properties of social networks to narrow the focus of search queries to only those blogs that the user is interested in.

ACKNOWLEDGEMENTS

I would like to thank Dr. Tony White for supervising me and providing the support needed to successfully complete this project.

I would also like to acknowledge Bart Ramson who got me blogging in the first place.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Background Information

Increasingly today, corporations, governments, and private citizens are demanding to know what others are thinking. Corporations would like to know what the latest trends are in society today, governments need to know how policies are being received amongst the grassroots, and private citizens often are simply interested in knowing what their peers are thinking about. Weblogs, or blogs for short, are a recent phenomenon in cyberspace that has emerged which offers an incredibly useful collection of information for media analysis. Blogs offer individuals the ability to read what people on the Internet are thinking right now. Indeed, the nature of blogs, online journals written by individuals around the world that are updated frequently, often daily, present an unfiltered view of world, discussing whatever the author feels like talking about. The content found on blogs vary greatly from topics such as technology tips and tricks, politics, arts and entertainment, and even personal accounts of the author's daily life. The unedited, uncontrolled nature of this media means that there are no limits to what blogs can talk about.

The rapid growth in the number of blogs on the Internet also means that this phenomenon is not something restricted to a small subset of the most technologically savvy individuals in society. This makes blogs highly relevant and an attractive target for data mining. While, only a handful of blogs initially existed in 1998, the numbers have grown exponentially over the years to the point that they now number in the millions, according to leading blog tracker, Technorati [Lindahl and Blount, 2003; Technorati, 2005]. Many blogs also have high readership numbers, such as the popular technology

blog, Slashdot, which has an audience numbering in the hundreds of thousands [Slashdot, 2005]. With millions of blogs and their associated authors, or "bloggers", expressing ideas, the amount of information one could find in this subset of the World Wide Web is clearly immense.

More important to this paper, however, is the fact that blogs are not simply self-contained journals on the web. Like any website, blogs contain links to other sites – this includes other blogs. Bloggers will link to others with similar interests. These linkages allow sites to interact with each other, forming an online community. What emerges from this is the formation of social networks within the so-called blogosphere [Herring et. al., 2005], creating order amongst the chaos of the web. As Section 2.0 of this paper outlines, the power of social networks is great, so exploiting the inherent social networks in the blogosphere is something of great interest.

1.2 Problem Description

The project described in this report attempts determine if we can search the blogosphere to see what people are saying about any particular topic. We will attempt to harness the power of the blogosphere determining if useful, relevant search results can be returned using simple keyword queries. This may seem simple like a simple task since keyword searching is something that even the simplest search engines are capable of. However, with the ability for anyone, anywhere to start publishing a blog, finding the information one wants while minimizing the complexity of a query may not be as simple as searching every single blog for a specific search term. The meaning of a keyword is dependent not only on the dictionary meaning of the word, but also the meaning the human user placed on it and the context in which it is found. For example, if one were to

want information regarding the recent Canadian federal budget, a simple search for the term "budget" would be inadequate. Searching for the term on the popular search engine Google produces over 93 million results – far too many for anyone to sift through. Even restricting the search to blogs using the blog search engine Technorati produces over 180,000 results with posts about a disparate range of topics. The trick for conventional searches, then, is to expand the keyword search to provide the needed context. For example, searching for the terms "Canadian federal budget" on Google reduces the number of hits to 3.2 million. Table 1 outlines the results of various keyword searches performed on the blog search engine Technorati. As we can see from the table, providing context clearly narrows the amount of hits returned. The trade off, however, is that search queries must become more complex.

Table 1: Search Results of Various Keyword Queries

Query	Hits	Query	Hits
budget	193,061	canadian federal budget	2211
senate	163,333	canadian senate appointments	152
hotel	440,388	moby cd hotel	326

Source: http://www.technorat.com on 2005/03/31at 7:04pm.

The question that this paper attempts to answer, then, is whether or not it is possible to return concise, relevant search results from the blogosphere while minimizing the complexity required in the actual search query. To do this, we will use the power of social networks to limit the scope of search queries such that a simple keyword such as "budget" will return only pages that the user will want to see. More specifically, if we search only within blogs located within, to continue our example, a social network of Canadian political sites, then the need to provide contextualizing keywords in the search

query will no longer exist. This is analogous to searching for books only within a specific genre. It may be helpful first, however, to outline what a social network is in the context of computer networks and the web.

2. SOCIAL NETWORKS AND BLOGS

In the physical world, a social network is a "network of friendships or other acquaintances between individuals" [Girvan and Newman, 2002]. As friends or acquaintances, these individuals often share common interests and backgrounds with each other. At a basic level, everyone participates in a social network, defined by the everyday interactions one goes through. The concept is one that spans disciplines such as sociology, political science, and in our case, computer science, since in all cases the nature of how we form and keep human relationships is of great interest. We can easily transfer the idea of the social network to that of computer networks, and more specifically, to the blogosphere. In this case, each blog acts as an individual, or a node in the network, and the hyperlinks on each blog act as the connections between the nodes. Figure 1 demonstrates the basic structure of a social network in the blogosphere as an undirected graph, showing how the network of blogs is analogous to a physical network of friends and classmates. We can further transfer the concept to computer networks when we observe that the people who use computer networks "have social relationships with each other that are embedded in social networks" [Wellman, 1996]. From this, we can infer that the network of blogs within the blogosphere has an inherently intelligent human-based organizational structure. This structure has several important properties which we will apply here to solve our searching problem.

The first significant property is that of "community structure" in which we see similar nodes densely clustered together within a wider network. In the context of a physical social network, these clusters of nodes represent social groupings of those with common interests or backgrounds [Girvan and Newman, 2002]. Applying this to the

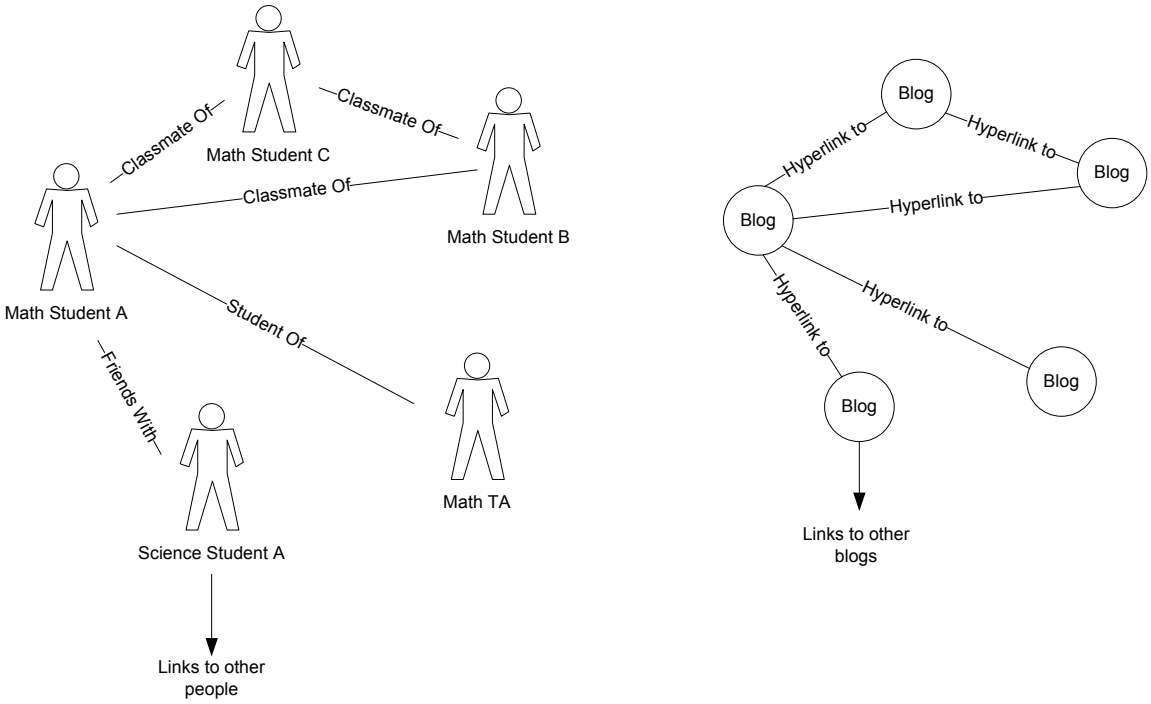
blogosphere, and treating the hyperlinks between blogs as connections in the network, we would expect blogs to be clustered together such that they all cover similar types of topics. Therefore, we can infer that if we were to start at a specific blog and search within it and it's neighbours, we would be predominantly searching blogs of a similar nature and type, providing the context that we were seeking earlier in our problem description.

The second property is the "small world effect", which states that the distance between two vertices in any network is short. Indeed work has suggested that this property is "pervasive in networks arising in nature and technology, and a fundamental ingredient in the structural evolution of the World Wide Web" [Kleinberg, 1999]. The basis of this is an experiment performed by Stanley Milgram in which complete strangers in Nebraska were tasked with getting a letter to a stockbroker in Boston. These strangers could only pass the letter on to someone with whom he or she was on a first-name basis. Milgram found that the average number handoffs required for a letter to be received by the stockbroker were only about six [Newman, 2001]. If one were to take each handoff of the letter as a connection in a network, we can see that within a social network of any sort, one can expect that the distance between any two nodes would be small. As a corollary, this means that a web crawler would only need to crawl very few levels before a sizable amount of the social network is covered. Moreover, Girvan and Newman discuss that many networks display a property of transitivity, in that nodes which share a common neighbour are likely neighbours of one another [2002]. As will be discussed later in Section 3.1 this makes extracting a community of blogs from within the greater

blogosphere much less complex since the depth one must crawl to retrieve a suitable number of blogs is low.

A third property is the idea of trust relationships in the social network. When a human decides to form a relationship with another, this forms a social exchange. In one regard, the exchange occurs to fulfill a purpose of some kinds. In another regard, "exchanges involve investments, gains and losses of time, money, energy, emotions, expectation, and many other energetic and motivational elements" [Rodrigues et. al., 2003]. Put simply, when someone in a social network creates a connection with another individual, then there is an implicit recognition that making that connection was worthwhile. In other words, an individual trusts the other enough to make a connection. Within the blogosphere, connections are made in the form of hyperlinks to other blogs. The fact that a blog has linked to another means that there is some value in the other blog. Extending this idea, if a blog is linked to by many people, then, theoretically, this means that the blog is found to be worthwhile by many, improving its level of trustworthiness. This concept, as will see later in the paper, is important in determining which blogs are more valuable than others.

Figure 1: Graph of Social Network



In the same way that a network of friends are connected by their relationships with each other, blogs in the blogosphere are connected by the collection of hyperlinks located on each page.

3. SOLUTION

3.1 Overview

Let us return, now, to the original problem outlined in this paper. Namely, that of narrowing search results while maximizing the amount of generalization possible within a keyword search query. To accomplish this, we will implement a blog search engine, called BlogCrawler which will use the inherent social networking properties of the blogosphere, ensuring that to the end user searching for information is as simple as using any other search engine available on the web. In the perspective of the end-user, he or she will still be required to enter at least one keyword to search on. However, we will eliminate the need for context specifying keywords, such as "Canadian" and "federal", to use our previous example, by limiting the number of sites that we will search to those of a specific topic. Instead of searching millions of sites for a specific keyword, we will only search hundreds.

This obviously leads to the question of how we determine which sites to include in our query and which to ignore. This is where we use the properties of social networks to aid us. We know that because of the property of community structure, neighbours of a blog will be of similar type. We also know that due to the small-world effect, the members of the social network of blogs we want to search will all be situated near each other in terms of number of links traversed to get from one to another. Finally, in the context of our problem, we know what types of blogs we are interested in. For example, we will know beforehand that we want information about Canadian politics. Therefore, in our solution we will pre-select a small number of "expert" blogs deemed to be representative of the type of blogs we want to search. We will treat these selected blogs

as the root nodes of the social network being traversed, and then crawl through the blogs connected to them. The end result is that after traversing only a few levels of links, we will have indexed a sizable number of blogs, most of which should be similar in type to the root blogs.

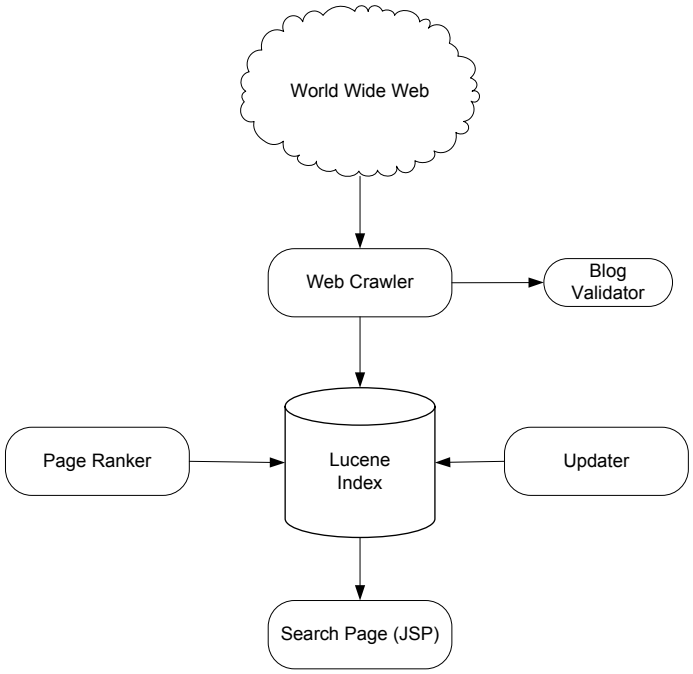
Once we have a collection of blogs to search from, then finding the information we want is simply a matter of entering a general keyword and performing a text-based search of the content of the blogs collected. If we were successful in limiting the type of blogs in our collection, then the results returned should be limited to only those topics that we are interested in. There is one caveat. Because of the uncontrolled nature of the blogosphere, anyone can write anything they want. As well, some blogs have more relevant information and are more trustworthy than others. This is where the idea of trust relationships within a social network comes into play. Each blog will be assigned a rank, similar to that of the PageRank score given to websites on Google [Nanno et. al., 2004]. This rank will represent a blog's level of trustworthiness. Hence, the results of a keyword search will be sorted such that those sites with the highest rank are situated at the top of the search results, ensuring the most relevant and trustworthy hits are the first ones the user sees.

There are several components we will need to create to implement such a solution. Those are a web crawler, a ranking module to calculate the expert level of blogs, and a front-end user interface for the user to access when performing search queries. Figure 2 outlines the architecture of the system.

3.2 Software

BlogCrawler will be implemented using a pure Java solution for all the modules, including a Java Server Pages (JSP) application for the front-end web-based search engine. Java was chosen for a variety of reasons, including the fact that it is cross-platform compatible, allowing all types of machines to use the same code base. This simplifies the task of ensuring that the code can run on the widest range of machines as possible. In addition, the standard Java library also includes tools to easily connect to, retrieve, and tokenize HTML web pages, which is helpful in the implementation of the search engine.

Figure 2: System Architecture



Another deciding factor to use Java was also the decision to use the Apache Lucene library, a full-featured text search engine. Lucene allows developers to efficiently store and search large amounts of any type of data, including web sites

[Apache Jakarta Project, 2005]. Since a large portion of the project involves performing keyword searches on text, it was felt that using Lucene would ensure the overall efficiency of the system. Because the engine is written entirely in Java, and the fact that every element of the project would need to access the index, it was felt that the use of Java was the most appropriate choice of environment.

3.3 Web Crawler

The web crawler is responsible for gathering the blogs which we want to search and is the most complex component in the system. Unlike standard web crawlers, the requirements of the system mean that it cannot blindly follow every single link on a blog. Since the blogosphere is only a subset of the greater World Wide Web and individual blogs often post links to sites that are not blogs, doing so would lead to a massive amount of web sites crawled that are outside the scope of our search parameters. Clearly, the web crawler needs to be intelligent enough to crawl only those sites we want. To solve this issue, the crawler needs to be equipped with a validation module that verifies whether or not a web page is a blog. Only if the validation determines that a page is a blog will it allow the crawler to index the site. The end result should be that only those sites situated within the blogosphere will be searched.

The basic algorithm for web crawling is a simple one. Simply access a site, store its contents in the index, extract the links on the site, and recursively crawl those links. Note, however, that due to the nature of the web, actually using a recursive algorithm would create an exponential number of instances of the crawler. We can easily solve this problem by using a queue instead as seen in Algorithm 1.

Algorithm 1: Basic Web Crawler

```
Algorithm Crawl(firstSite)
   Queue queue := new Queue;
   queue.enqueue(firstSite)
   While queue not empty
      currSite := queue.dequeue();
      siteContents := getContents(firstSite);
      outgoingLinks := getLinks(firstSite);
      index(currSite, siteContents);
      queue.enqueue(outgoingLinks);
   Loop
```

[Blum et. al, 1998]

As we noted earlier, we cannot follow every single link. There are several issues when programming a web crawler that we need to be cognisant of. First, we need to ensure that the web crawler does not continue to run for an indeterminate amount of time. This is a distinct possibility with millions of blogs on the web online today and the number of web hosts doubling every year [Kobayashi and Takeda, 2000]. To resolve this issue, we simply need to limit the depth that our web crawler will crawl. For example, assume that the initial list of blogs we crawl represent depth zero. Then every blog linked from those at depth zero would be depth one, those linked from depth one will be depth two, and so on. Therefore, we simply place a condition on our crawler to stop indexing blogs that exceed a user defined depth.

The second issue we need to address is that of ensuring that only blogs are crawled. We must remember that the blogosphere is not self-contained within the World Wide Web since blogs not only link to other blogs, but to other websites that we do not want to index. If we allow the crawler to exit the blogosphere into the greater web, then the chances are slim that the crawler will return to where we want it to. This problem can be addressed by validating each page the crawler retrieves before indexing it. If the page validates as a blog, then the crawler will index it, extract the links on the page, and

continue crawling. If it does not validate it, the crawler will ignore the page. Section 3.4 goes into further details on how the validation works.

Finally, we need to ensure that we do not index pages twice, since most blogs, and web pages for that matter, maintain a many-to-many link relationship with other blogs. This is resolved by maintaining a list of websites already crawled and skipping those links which lead to previously traversed pages.

From our basic algorithm, then, we now have a more intelligent and efficient crawling algorithm, as seen in Algorithm 2. This algorithm forms the basis of the **Crawler** class implemented in our solution, traversing the social network of blogs we are interested in.

Algorithm 2: Intelligent Crawling Algorithm

```
Algorithm IntelligentCrawl(rootSites, maxDepth)
   List visited = new List;
   Queue queue = new Queue;
   queue.enqueue(rootSite)
   while queue not empty
      currSite := queue.dequeue();
      if currSite.depth <= maxDepth then
         visited.add(currSite)

         if not visited.contains(currSite) then
            if isBlog(currSite) then
               siteContents := getContents(currSite);
               outgoingLinks := getLinks(currSite);
               index(currSite, siteContents);
               queue.enqueue(outgoingLinks);
            end if
         end if
      end if
   loop
```

Looking at each individual blog to be indexed, we now have to determine what properties of an individual blog we need to store. We have already mentioned that we will be using the Lucene library to index the pages that we crawl. As part of its

implementation, Lucene allows us to index any number of fields with whatever content we wish. The obvious fields to index are the ones the end-user is interested in, namely the address, title, and contents of the page. These are not the only things that need to be indexed, since we also need to store attributes that relates the page to its position within the social network. Our discussion of social networks identified two key aspects of blogs that we are interested in. Firstly, we need to know what other blogs the page is connected to. To that end, we also index the complete list of outgoing links contained on the page. Secondly, we need to know the rank of the page to determine which blogs are the most trustworthy. Therefore, we will also index the page's rank. Since a page's rank is dependent on the rank of other pages in the index, we will not be able to calculate the rank during the web crawl, so we index the page with an initial rank of zero. All of this is accomplished with the **PageIndexer** class of the project.

Now that we have the overall structure of the crawler module designed, we turn to ensuring that the module is as efficient as possible. Like any algorithm that we design, we are particularly concerned with time and speed efficiency and memory usage. In both cases, implementing the web crawler created many challenges in ensuring that our algorithm was as efficient as possible.

With respect to speed efficiency, we needed to ensure that crawler indexed as many pages as possible in a minimal amount of time. Particularly troublesome was the fact that as the crawler traversed deeper into the network, the number of links queued to crawl grew exponentially in the same way that the number of leaves on a tree grows exponentially at each height. Since a particular instance of a crawler can only realistically crawl the web one page at a time, we attempt to reduce the required time to

crawl the network by making our crawler multithreaded. Therefore, at any given moment, several sites are being retrieved, validated and indexed at once. These threads share a common queue and visited list so that we reduce duplication of work.

In terms of memory usage, we immediately see that if the number of links we need to crawl grows exponentially, then that means that the storage space required to contain the queue of links grows at a similar rate. Therefore a disk-based queue was used to maximize the amount of storage available for the queue. As implemented in the **DiskQueue** class of the project, we store the queue in a text file. Data access is accomplished by maintaining two file pointers: an output stream that appends to the end of the file where we write to when queuing an item, and an input stream that begins at the top of the file and reads each line successively whenever we remove an item from the queue. The two pointers minimize the seek time required to access the data. Although memory access on a hard drive is slower than through volatile memory storage, we benefit by ensuring that we will not run out of memory when crawling.

3.4 Blog Validator

To validate whether a site is a blog or not, we created a **BlogValidator** class which is designed to filter out unwanted sites. We accomplish this by recognizing the fact that blogs share common formatting characteristics, including a consistent sequence of date-entry pairs [Nanno et. al., 2004]. Figure 3 shows an example of a blog with this characteristic. Particularly, we note that the date on each entry is consistently formatted in terms of date expression (such as “dd/mm/yy”) and formatting style (such as font size, bolded). The validator takes advantage of this by analysing the structure of an HTML document, extracting the dates on the page and determining if the sequence of dates is

consistent with that of a blog. Specifically, the validator determines that a site is a blog if and only if:

1. There exists a sequence of dates, spaced out by a user defined minimum number of characters.
2. The sequence of dates is ordered in ascending or descending order.
3. The HTML tag sequence surrounding each date instance is uniform for all entries in the sequence.

To accomplish this, the BlogValidator class extracts all the dates on a page that match one of a number of predefined regular expressions. If a date follows another date within a specified number of characters, however, it is ignored. Once extracted, the dates are sorted into bins based on the regular expression that the date matches. For example, the dates "2005/02/14" and "2004/12/01" would go in the same bin, having matched against the date format "yyyy/mm/dd". The bins are then further split by the HTML tag sequence the date was found in. For example, a date may be found following an tag sequence of " <p>". We then take the largest list of dates, and assume that the list represents the sequence of date entries for the articles on the potential blog. If those entries are found to be in ascending or descending order, then the page is determined to be a blog and the validator returns true.

Using this algorithm, we are able to detect blogs produced by any type of blog software and presented using any type of template. While thought was given to simply using a precompiled list of "acceptable" blogs, this approach of determining whether a page is a blog in real-time allows for much more flexibility in discovering lesser known blogs. In fact, as will be discussed in Section 4.1, this algorithm is remarkably successful in identifying blogs.

Figure 3: Sample of a Blog with Date Entries

Screenshot of http://401blog.blogspot.com illustrating a sequence of date-article entries.

3.5 Page Ranker

After we use our web crawler to extract and index the blogs were are interested in, we now turn to ranking the blogs to ensure that the most important ones are given the most weight when performing keyword searches. Doing this prevents blogs that are relatively new and those that are unpopular from being returned ahead of the blogs that the end-user actually wants to see. We accomplish this by assigning a rank to each blog which represents its relative importance or trust level compared to other sites in the index. When performing a keyword search, blogs with higher rankings will appear first in the results listing, followed by those with lower ranks. This is not a new concept, as demonstrated by Google, which returns search results based on the rank of a page calculated by the highly successful PageRank algorithm [Eiron et. al., 2004]. So, rather

than create our own algorithm, we will use a modified version of the PageRank algorithm to calculate the ranks for the blogs in our index.

PageRank is premised on the idea that the more important a web page is, the more other pages will link to it. Therefore, the more links to a page, the higher its rank is and the higher its importance. Mathematically, PageRank begins by assigning each page an initial rank of $1/N$, where N is the number of pages in the index. Let N_u be the outdegree, or number of outgoing links, on page u, and let $Rank(p)$ be the rank of a page p. Also, let B_v be the set of all pages with a hyperlink to page v. The rank of a page, v, at iteration i is calculated as follows:

$$Rank_{i+1}(v) = \sum_{u \in B_v} Rank_i(u) / N_u \quad (1)$$

Since the rank of a page is dependent on the rank of others, we iterate through all of the pages in the set of pages until the ranks stabilize to within a specified threshold. The rank vector that is calculated from this formula is calculated once and the results are used for every search query.

The PageRank algorithm, however, is susceptible to the problem of assigning pages with little actual authority a high rank simply because the page was heavily linked to [Haveliwala, 2002]. This is a problem if we want to generalize keyword searches as much as possible while maximizing the relevancy of the search results. To overcome this obstacle, we bias the rankings by creating a "topic-sensitive" PageRank. As Haveliwala argues, biasing the page ranking towards specific pages allows for personalization in the ranking. In this case of BlogCrawler, we will personalize the rankings so that blogs which we deem are experts or important will bias the ranking of all the blogs in the index towards them. Take the example of someone searching political blogs. If a user is

specifically interested in conservative blogs, then it would make the most sense to bias the rankings towards those sites which present a conservative viewpoint. If the user is interested in liberal blogs, then using same index, the user can bias the rankings towards those blogs with a liberal viewpoint. This allows us to further narrow the search results to what the end-user wants.

Biasing the ranking algorithm is actually quite simple. Essentially, we want to ensure that those blogs which the user finds important have high scores. To do this, we modify the initial rank given to a page, such that a page has a rank of 1 if it is in the list of "expert" blogs, and a rank of 0 if the blog is not in the list. On each iteration, the PageRank is calculated as outlined in (1). The difference is that on each iteration, the initial rank given to the expert blogs is further diffused across the entire network, meaning that a blog's ranking is almost entirely dependent on its proximity to an expert blog.

Implementing this algorithm using the Lucene index is simply a matter of updating the appropriate rank field for each blog in the index. One final issue that needs to be resolved is that often times, there will be multiple pages from the same blog site indexed. To ensure that these internal linkages do not affect the final rank of a blog, we exclude outgoing links that point to pages on the same web host when calculating the rank. Once the ranks are calculated the index is now ready for keyword searching via a web-based search engine.

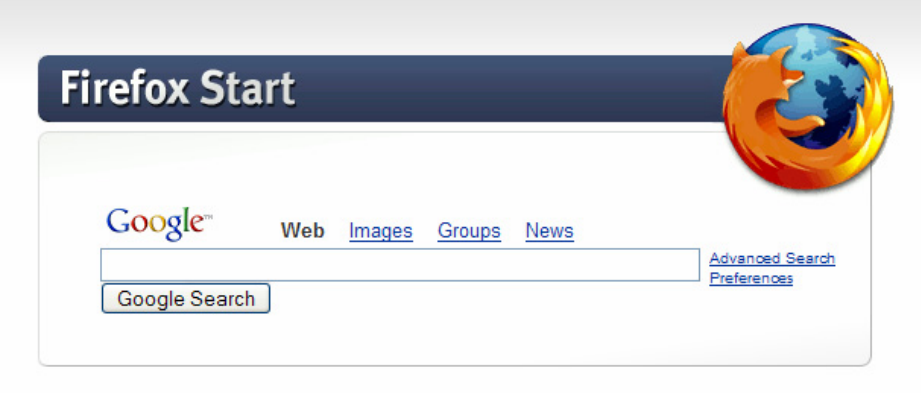
3.6 User Interface

To the end-user, searching through the blogs we have crawled should be as simple as using any other search engine. Figure 4 shows what a standard search box looks like,

and Figure 5 shows what the BlogCrawler search box looks like. The common element necessary present is a text box to enter the keywords the user wishes to search for. To search for information, all the user is required to do is enter his or her search query in the textbox, click Search and, as seen in Figure 6, wait for the search engine to display a list of appropriate blogs. Also present are two options to further refine the query. The user is able to enter a variety of keywords, separated by a space. Selecting "All Terms" will search through the index for items containing all of the keywords entered in the text box. Selecting "Any Terms" will return items that contain one or more of the keywords entered. In any case, if results are found, the search engine will return a sorted list of blogs, complete with an excerpt from the blog that matches the keywords entered.

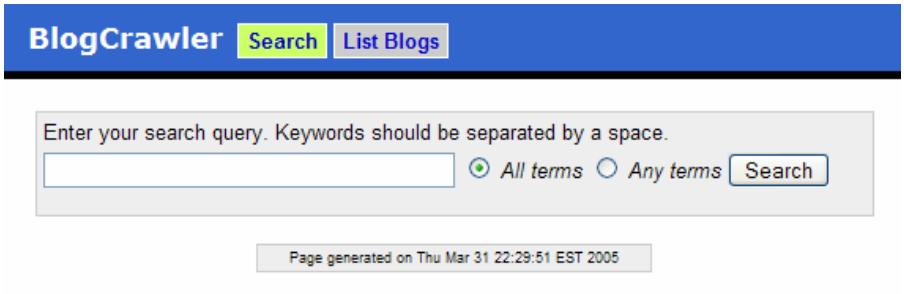
Programmed using JSP, the search page interfaces directly with the Lucene index and uses the Lucene's built in boolean search query objects to return a list of blogs. Furthermore, the list is sorted by the blogs' calculated ranks (see Section 3.5), and then by the query score which is calculated by Lucene and reflects the amount a blog matches the keywords in the query.

Figure 4: Google Search Interface



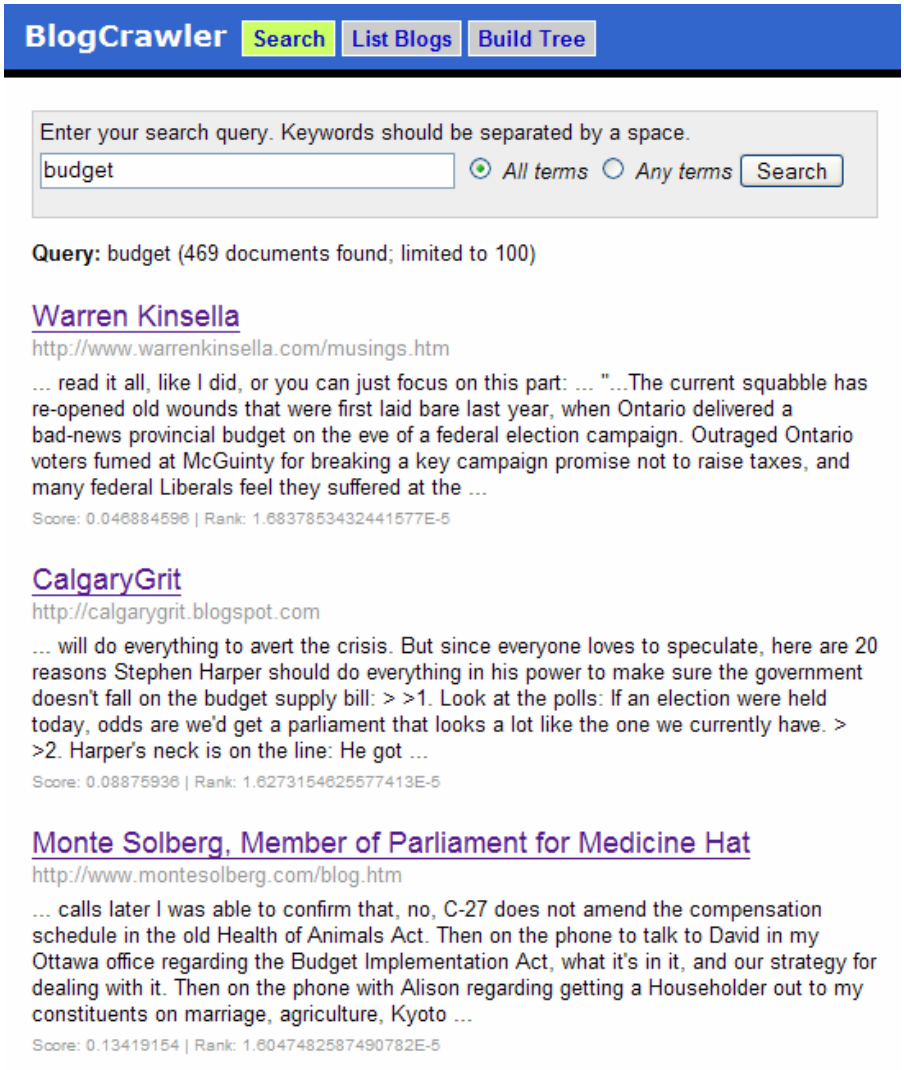
Main search page for Google (via the Firefox web browser)

Figure 5: BlogCrawler Search Interface



Search interface for BlogCrawler search engine

Figure 6: BlogCrawler Search Results



Screenshot of results page after searching for keyword "budget"

4. RESULTS

With implementation of BlogCrawler complete, we now move to evaluating the system. Recall that in the original problem statement, we wanted to know if it were possible to use the inherent social networking properties of blogs so that relevant hits would be returned using a generalized search query. Based on these parameters, we will focus on two main issues. We will first look at the web crawler's effectiveness in extracting the blogs we want, while filtering out everything else. Success in this regard is paramount since the success of the system requires that we actually have blogs to search from. Secondly, we will look at the actual results of keyword search queries performed on the index of blogs we constructed. We will determine whether or not we can retrieve relevant results from generalized queries and, if this is the case, relate this back to idea of social networks.

4.1 Crawler Effectiveness

The web crawling portion of the BlogCrawler system can be evaluated in terms of two criteria. Firstly, how efficient was the actual crawling process. For this criterion, we can look at how fast web sites are crawled, and examine where the bottlenecks were during the crawling process. The second criterion is the accuracy of the validator and whether or not we were successful in extracting only blogs.

In terms of the efficiency of the crawler, we were relatively successful in producing a web crawler that maintained a consistent rate of operation in that regardless of how long the system was running, a web page was examined at least once every 10 seconds. This is in contrast to previous versions of the web crawler that appeared to slow

down the longer the system was crawling. This slow down was predominantly caused by lengthy seek times as the disk-based queue grew, since we were only using one file pointer. For example, at one point, we observed that the file storing the list of sites already visited was approximately 500 kb. The file storing the list of sites waiting to be crawled, however, was approximately 80 Mb, which caused lengthy seek times when appending new sites to the end of the file. We were able to overcome this limitation by implementing a disk-based queue with two file pointers; one pointing at the head of the file and the other pointing to the end of the file, ensuring that the seek time required to access either end of the disk-based queue remained constant. With the final version of the web crawler submitted with this paper, our main test run was able to index 1480 Canadian political blogs over the course of 2 days, not including sites that were rejected by the blog validator. This equates to approximately 30 blogs indexed per hour. While this performance rate is not poor, we were hoping to increase the rate of crawl. Section 5 outlines potential improvements that we can make to increase the web crawler's efficiency.

Accuracy is the second criterion in which we can evaluate the web crawler. Indeed, for all other aspects of the system to work, we need to be confident that the sites being indexed are actually blogs. In this regard, we were very successful in implementing a validation module that more often than not, correctly identified web pages as blogs. These blogs include those based on standardized templates (e.g. http://calgarygrit.blogspot.com), and custom designed templates (e.g. http://www.freethought.ca). Using our list of blogs crawled, as outlined in Appendix B, a random sample of 50 blogs revealed the validator incorrectly identified 4 web pages as

blogs. In one case, there would be no way beyond natural language analysis to determine that the site was not a blog. That gives us a success rate of 94%, which for our purposes is more than adequate. Further random sampling returns similar results.

For the most part, then, our web crawler was successful in giving us a good base of blogs to analyze and search upon. Although there were difficulties in overcoming the memory storage problems caused by the massive amount of links that accumulated over days of crawling, we were able to crawl a significant portion of the blogosphere that we were interested in.

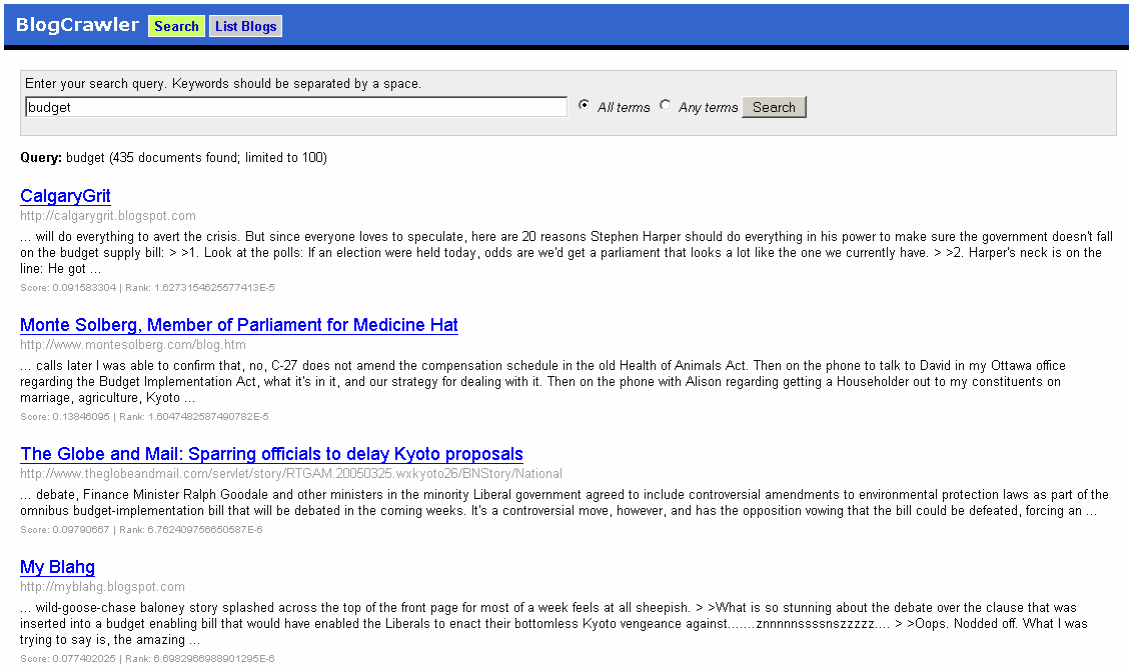
4.2 Relevance of Search Results

Given that we have an acceptable base of blogs to search from, we now turn to evaluating the actual search results returned from BlogCrawler. In Section 1.3, we discussed how generalized search queries were inadequate in narrowing the scope of the returned results. Consequently, when we evaluate the effectiveness of our system, we need to look at how relevant the returned results are when using very general keywords. The main example used in this paper has been Canadian political blogs, so we continue to use this example in our evaluation of BlogCrawler. In our sample index, we begin by feeding two Canadian political sites (http://www.freethought.ca and http://calgarygrit.blogspot.com) into the web crawler and then allow it to run for two days. Afterwards, we rank the blogs using the two previously mentioned blogs to bias the rankings, then run the Updater module to ensure that all the pages in the index are recent. Finally, we perform various keyword searches using the BlogCrawler web application. For the purpose of comparison, we will also examine the results returned by

the blog search engine Technorati (http://www.technorati.com) using the same keyword query.

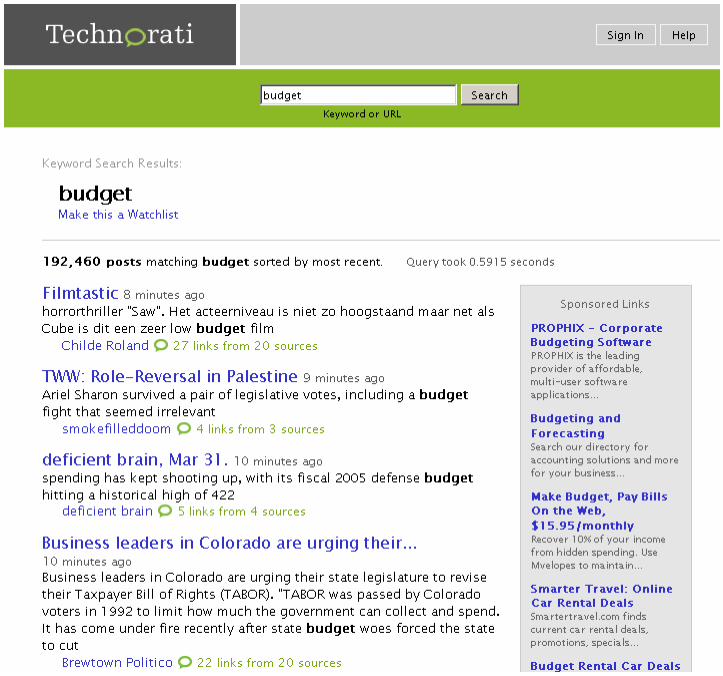
Let us first examine a search on the keyword “budget”, which happens to be a very general keyword that can apply to many situations. To satisfy the requirements of the system, we would like to only receive blogs that discuss issues surrounding the federal budget, an item prominent in the public consciousness at the time writing. More specifically, on March 31, 2005, when the search was run, the main issue was the federal government’s decision to include environmental protection measures in the bill approving the budget. We therefore, would expect that the returned blogs would be discussing this issue. After running a search on BlogCrawler for the keyword “budget”, we see that the returned blogs do discuss this issue for the most part. In contrast, a keyword search for “budget” on Technorati returns a plethora of blogs, with no consistent topic of conversation. Figures 7 and 8 show the results of the two search engines, while Table 2 summarizes the main topics of the top 10 hits returned by both search engines. Searching for another keyword, in this case “senate”, we now expect to retrieve blogs that have information regarding the appointments to the Canadian Senate that occurred in March 2005. BlogCrawler again provides the results we expect as Table 3 demonstrates. Hence, we can conclude that by feeding a specific type of site into the web crawler as the root node in the social network, and following the links to other blogs, we are able to restrict the search engine to a specific topic without needing to enter clarifying keywords in the search query.

Figure 7: Results Returned by BlogCrawler



The results returned for the keyword "budget" predominantly discuss the same issues. Although we have one false positive (Globe and Mail), the topic remains consistent with the other blogs.

Figure 8: Results Returned by Technorati



Results returned by Technorati (http://www.technorati.com) for the same keyword search.

Table 2: Topics of Top 10 Hits for "budget" Keyword Search

BlogCrawler		Technorati	
Blog	**Topic**	**Blog**	**Topic**
Warren Kinsella	Provincial budget	Childe Roland	Film
CalgaryGrit	Federal budget	Smoke Filled Doom	Israel budget
Monte Solberg	Federal budget	Deficient Brain	American budget
Globe and Mail	Federal budget (False positive)	Brewtown Politico	State budget
My Blahg	Federal budget	Brewtown Politico	American budget
Living in a Society	Federal budget	Mac Professionell	(German language post)
TDH Strategies	Federal budget	Oleg Dulin	American budget
On the Fence	American budget	unLively Lives	Personal diary
Freethought.ca	International trade	The Reminisce Mind	Personal diary
Capitalist Pig vs. Socialist Swine	Federal budget	New Leadership Blog	American election

BlogCrawler search performed for "budget" on 2005/03/31 at 2:45pm. Technorati search for "budget" performed on 2005/03/31 at 2:47pm.

Table 3: Topics of Top 10 Hits for "senate" Keyword Search

BlogCrawler		Technorati	
Blog	**Topic**	**Blog**	**Topic**
CalgaryGrit	Canadian Senate	Semidi	American Senate
Peace, order, and good government	American Senate	Swing State Project	American election
My Blahg	Canadian Senate	Gluetree.com	Star Wars trailer
On the Fence	Canadian Senate	Dudes Drivel	American Senate
Capitalist Pig vs. Socialist Swine	Canadian Senate	Dudes Drivel	American Senate
Highway 401 Blog	Canadian Senate	Centerfield	American Senate
Warren Kinsella	Government audit	Uncountable Spoons	American Senate
Lotusland	American Senate	Blast Off!	Obituary
BlogsCanada	Canadian Senate	Citrus Commando	American election
Crawl Across the Ocean	Canadian Senate	LANL: The Real Story	University/student politics

BlogCrawler search performed for "senate" on 2005/03/31 at 3:38pm. Technorati search for "senate" performed on 2005/03/31 at 3:38pm.

We also discussed the idea that blogs returned from searches should also be trustworthy. From our analysis of social networks, we concluded that if we rank blogs based on the number of links to it, then we should be able to order our collection of blogs by relative importance within the greater blogosphere. Since our search engine sorts hits by the rank of a blog, this leaves us to show that the actual page ranks calculated by BlogCrawler are accurate. Qualitatively, we see that the rankings calculated by BlogCrawler, supplemented by the biasing discussed in Section 3.5 appear to reflect this idea (see Appendix B for a sorted list of blogs). Indeed, the sites we deemed "expert" blogs when calculating the page ranks all appear at or near the top of the ranked list of blogs contained in the index. We also note that the problem of the BlogValidator incorrectly identifying sites as blogs is overcome by the fact that those sites tend to have low page ranks relative to actual blogs.

4.3 Overall Results

Based on the results we have observed, we can safely conclude that BlogCrawler is successful in implementing a solution demonstrating the properties of the blogosphere and the power that social networks have. This is not to say that the system is perfect, as will be discussed in the following section. However as a prototype, BlogCrawler was able to provide us with fairly accurate and very relevant hits on our generalized keyword queries. The system was also able to rank blogs according to our personal preferences as defined by the blogs we used to bias the page rank calculations. As a basis for evaluating the use of social networks within the application of searching, the software succeeded.

5. POTENTIAL IMPROVEMENTS

As a prototype, there are several improvements that can be made to BlogCrawler in many aspects of the system. Firstly, changes to make the system more efficient come to mind. These include speeding up the crawling process and making the system easier to use. However, we also note improvements to expand the scope of the system. This stems from the fact that the system was designed to be a personal search engine – that is, the index of blogs stored and the rankings calculated for each page reflect the personal tastes and biases of the individual to which the search results are geared towards. Because of this, the system is not intended for use by many people at the same time, unless they all happen to want search results from the same topic. Therefore, there are issues related to how we can allow for expansion of BlogCrawler to accommodate a wider scope for a wider audience.

In its current state, the web crawler is designed to be run on one machine. Although the crawler is multi-threaded, there still is a processing power limitation created by the validation module. We would suggest then, that the crawler be designed to run on many machines at the same time, parallelizing the crawling process. This is what many commercial search engines do, in fact. To accomplish this, it would simply be a matter of making the disk-based queue of links accessible by multiple machines over a network. A suitable solution could be a database system, for example. Not much more needs to be done, however, since the crawler is already designed to have multiple instances of it running in memory.

The validation module can also be improved. We stated earlier that although we are very successful in correctly identifying blogs, we still occasionally receive false

positives. Ultimately, the ideal solution would be to implement a natural language analyzer to determine that a site was a blog and not, for example, a list of press releases listed in descending order with extracts displayed on the page. The current design of the blog validation module looks at web pages structurally, but does not analyze the language being used in the page. Improving the module so that it can analyze both aspects of a web page would go a long way in ensuring that false positives are kept to a minimum.

Also in terms of efficiency, the process can be automated much more. Appendix C outlines how to deploy and use the software. The system is currently designed so that the user must manually begin the crawling, ranking, and updating processes. This was done to accommodate the use of shell scripts and other scheduling systems, such as cron jobs in the Linux operating system. As a personal search engine, however, greater steps can be taken to making the system easier for users who are not skilled in server administration, such as creating a graphical user interface, or eliminating the requirement of a web server to access the search interface.

In terms of expanding the system, BlogCrawler can be designed to accommodate many social networks of blogs, not just the one we decided to focus on. Appendix B lists two indexes of blogs we have crawled, however they are kept independent of each other. Therefore, if the search engine is configured to search political blogs, then the end-user can only search those sites. The system can be improved by allowing multiple networks of blogs to coexist within the same index, and indeed, multiple sets of page rankings. This allows the user to not only enter the keywords he or she wishes to look for, but also to select the network of blogs to search and which personalized page rank vector to use.

By implementing these changes, we should be able to transform BlogCrawler from a prototype designed to demonstrate the theoretical properties of social networks, to a fully functional search engine that allows users to search for the information they want efficiently.

6. CONCLUSION

As we noted in the opening of this paper, blogs are increasingly being used by a variety of users in the field of media analysis. Blogs inherently express the thoughts of the public at large, so the importance of knowing what is being said on the many blogs on the internet is important. With this comes the requirement to be able to search for what types of views we want in a more efficient fashion than is usually used. Indeed, unlike searching for a particular piece of information, looking for a specific string of text, media analysis requires people to look for a specific topic, but any type of viewpoint. Searches like this require the use of more general keyword queries, which we showed earlier to be cumbersome in terms of the amount of irrelevant sites the user would have to filter out. Therefore, this paper ultimately needed to answer two questions. First, do blogs exhibit properties of a social network? Secondly, can we use these properties to provide a level of human intelligence to search results while keeping the actual searching process simple? With BlogCrawler successfully implementing a solution that satisfies our requirements we can conclude several things regarding blogs in particular and social networks in general.

First, we can conclude that the blogosphere does constitute a social network with its interconnected set of hyperlinks and references. As our web crawler demonstrated, given any blog, its set of outgoing links linked it to other blogs of the same type, exhibiting the property of clustering, forming a community structure of blogs. The cluster of blogs we crawled all tended to be focus on the same issues, even if the opinions and viewpoints expressed on them differed. Moreover, we observed that the set of hyperlinks from one blog to another exposed embedded trust relationships in the

blogosphere, with the most important and trustworthy blogs being linked to the most. Blogs clearly do constitute a social network.

From our experimental search results, we also can conclude that using social networks in a searching application can make the searching process much more efficient. By limiting the search engine to a specific cluster of blogs, we see that effectively filter out irrelevant blogs that we would normally have had to filter out using contextualizing keywords. This also limits the numbers of blogs to only those with some level of authority since blogs that no one reads or trust will hardly ever be linked to. Social networks, then, add a modicum of human intelligence into the search process by recognizing and leveraging the human efforts made when constructing the social network in the form of linking to other blogs.

Interestingly, the conclusions we have found through our analysis of the blog social network also means that communities of blogs can be hierarchical. Blogs that focus on Jazz music form their own social networks, but may fall under a more general category of music blogs. Indeed, we would go so far as to say that the entire blog community may be hierarchical – using ontologies and intelligent agents to analyze and categorize individual blogs it is entirely possible to generate an overall blog network topology [Heflin, 2004]. We have already seen in a simple manner how we can use the inherent organization of social networks to aid us in implementing practical applications for common problems. Knowledge of how the entire blogosphere is structured would, therefore, bring many benefits.

In all facets of life, we use social networks because we recognize that the relationships we build with others are valuable and useful. In the world of Computer

Science, social networks allow us to add a human element to networking problems by recognizing the inherent organization structure social networks provide and realizing that there is information imparted whenever we decide to connect one resource to another. The common problem with searching is often that our search engine is not intelligent enough to recognize what we want. As we have seen with the BlogCrawler project, by using social networks, search engines can be intelligent enough to ensure that we are always satisfied by what we get back.

REFERENCES

Blom, Thom et. al. (1998) *Writing a Web Crawler in the Java Programming Language*. Retrieved 12 February 2005 from http://java.sun.com/developer/technicalArticles/ThirdParty/WebCrawler/.

Eiron, Nadav et. al. (2004) "Ranking the web frontier," *Proceedings of the 13th international conference on World Wide Web*, pp. 309-318.

Girvan, M. and Newman, M.E.J. (2002) "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, Volume 99, Issue 12, pp. 7821-7826.

Google (2005) *Google.* Retrieved 30 March 2005 from http://www.google.com.

Haveliwala, Taher H. (2002) "Topic-Sensitive PageRank," *Proceedings of the 11th World Wide Web conference*, pp. 517-526.

Heflin, Jeff (2004) *OWL Web Ontology Language Use Cases and Requirements*. Retrieved 4 April 2005 from http://www.w3.org/TR/webont-req/.

Henzinger, Monika. (2000) "Link Analysis in Web Information Retrieval," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*

Herring, Susan C. et. al. (2004) "Bridging the Gap: A Genre Analysis of Weblogs," *Proceedings of the 37th Hawaii International Conference on System Sciences.*

Herring, Susan C. et. al. (2005) "Conversations in the Blogosphere: Analysis 'From the Bottom Up'," *Proceedings of the the 38th Hawaii International Conference on System Sciences*.

Kleinberg, Jon (1999) "The Small-World Phenomenon: An Algorithmic Perspective," *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pp. 163-170.

Krishnan, Sriram (2004) *Writing a web crawler*. Retrieved 15 March 2005 from http://dotnetjunkies.com/WebLog/sriram/archive/2004/10/10/28253.aspx.

Lindahl, Charlie & Blount, Elise. (2003) "Weblogs: Simplifying Web Publishing," *Computer*, Volume 36, Issue 11, pp. 114 -116.

Nanno, Tomoyuki et. al. (2004) "Automatically Collecting, Monitoring, and Mining Japanese Weblogs," *Proceedings of the 13th international World Wide Web conference*, pp. 320-321.

Nardi, Bonnie A. et. al. (2004) "Blogging as Social Activity, or, Would You Let 900 Million People Read Your Diary?" *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, pp. 222-231.

Newman, M.E.J. (2001) "The structure of scientific collaboration networks," *Proceedings of the National Academy of Sciences*, Volume 98, Issue 2, pp. 404-409.

Slashdot (2005) *About Slashdot*. Retrieved 28 March 2005, from http://www.slashdot.org/about.shtml.

Wellman, Barry (1996) "A Sociological Perspective on Collaborative Work and Virtual Community," *Proceedings of the 1996 ACM SIGCPR/SIGMIS Conference*, pp. 1-11.

Rodrigues, Maira Ribeiro et. al. (2003) "A System of Exchange Values to Support Social Interactions in Artificial Societies," *Proceedings of the Second Annual International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 81-88.

Technorati (2005) *Technorati: What's happening on the Web right now*. Retrieved 31 March 2005 from http://www.technorati.com.

APPENDIX A – SOFTWARE USED IN BLOGCRAWLER

Programming Libraries

Java SDK 1.5.0	(http://java.sun.com)
Apache Lucene API 1.4.3	(http://lucene.apache.org)

Web Server

Apache Jakarta Tomcat JSP Server 5.0	(http://jakarta.apache.org/tomcat)

APPENDIX B – BLOG LIST

The following tables are the blogs indexed by the web crawling module, using the specified sites as the initial blogs in the queue.

Table B-1: Top 100 Canadian Political Blogs

Root Sites: http://www.freethought.ca, http://calgarygrit.blogspot.com

	Site	Page Rank
1	Warren Kinsella	1.68E-05
2	CalgaryGrit	1.63E-05
3	Monte Solberg, Member of Parliament for Medicine Hat	1.60E-05
4	Peace, order and good government, eh?	1.48E-05
5	Blogs Canada : Canada's Blog Site	1.39E-05
6	weblog :: jordoncooper.com	7.68E-06
7	Revolutionary Moderation	7.61E-06
8	Monte Solberg, Member of Parliament for Medicine Hat	7.04E-06
9	Monte Solberg, Member of Parliament for Medicine Hat	7.04E-06
10	The Globe and Mail: Sparring officials to delay Kyoto proposals	6.76E-06
11	My Blahg	6.70E-06
12	sean incognito	6.53E-06
13	LIVING IN A SOCIETY	6.35E-06
14	TDH Strategies - Solutions For Everyone	6.14E-06
15	On the Fence (www.nestruck.com)	5.75E-06
16	BlueTory.ca	5.38E-06
17	Gauntlet.ca - the politics of .ca	5.27E-06
18	The Brock Press	5.02E-06
19	freethought.ca :: welcome	4.87E-06
20	The Reasonable Tory	4.68E-06
21	Capitalist Pig vs. Socialist Swine	4.51E-06
22	Highway 401 Blog	4.46E-06
23	Warren Kinsella	4.32E-06
24	Warren Kinsella	4.32E-06
25	Babbling Brooks	4.26E-06
26	lotusland - nervous politics from a laid-back city	4.26E-06
27	The Armchair Garbageman	4.23E-06
28	Just in From Cowtown	4.17E-06
29	PolSpy - Canadian Political Commentary and Satire	4.09E-06
30	The Heart of the Matter	4.06E-06
31	BlogsCanada: E-Group	4.04E-06
32	Crawl Across the Ocean	4.02E-06
33	Silly Conservatives.	3.90E-06
34	andrewcoyne.com	3.69E-06
35	John Murney's Blog	3.58E-06
36	daveberta	3.53E-06
37	CalgaryGrit: 06/13/2004 - 06/19/2004	3.52E-06

38	CalgaryGrit: 03/20/2005 - 03/26/2005	3.52E-06
39	CalgaryGrit: 05/30/2004 - 06/05/2004	3.52E-06
40	CalgaryGrit: 02/06/2005 - 02/12/2005	3.52E-06
41	CalgaryGrit: 10/03/2004 - 10/09/2004	3.52E-06
42	CalgaryGrit: 10/31/2004 - 11/06/2004	3.52E-06
43	CalgaryGrit: 05/16/2004 - 05/22/2004	3.52E-06
44	CalgaryGrit: 01/30/2005 - 02/05/2005	3.52E-06
45	CalgaryGrit: 01/23/2005 - 01/29/2005	3.52E-06
46	CalgaryGrit: 06/27/2004 - 07/03/2004	3.52E-06
47	CalgaryGrit: 06/06/2004 - 06/12/2004	3.52E-06
48	CalgaryGrit: 06/20/2004 - 06/26/2004	3.52E-06
49	CalgaryGrit: 01/02/2005 - 01/08/2005	3.52E-06
50	CalgaryGrit: 01/09/2005 - 01/15/2005	3.52E-06
51	CalgaryGrit: 01/16/2005 - 01/22/2005	3.52E-06
52	CalgaryGrit: 02/13/2005 - 02/19/2005	3.52E-06
53	CalgaryGrit: 11/14/2004 - 11/20/2004	3.52E-06
54	CalgaryGrit: 02/20/2005 - 02/26/2005	3.52E-06
55	CalgaryGrit: 11/21/2004 - 11/27/2004	3.52E-06
56	CalgaryGrit: 05/23/2004 - 05/29/2004	3.52E-06
57	CalgaryGrit: 09/12/2004 - 09/18/2004	3.52E-06
58	Crawl Across the Ocean	3.48E-06
59	Sinister Thoughts	3.38E-06
60	Tilting at Windmills	3.17E-06
61	Paul Wolfowitz - Wikipedia, the free encyclopedia	3.12E-06
62	JimBobby Sez	3.00E-06
63	CathiefromCanada	2.84E-06
64	Voice in the Wilderness	2.78E-06
65	Peace, order and good government, eh?: Who are you and what have you done with Irwin Cotler?	2.62E-06
66	Stageleft:. Life on the left side	2.58E-06
67	Dean Rushes the Vote	2.58E-06
68	Stephen Taylor - Conservative Party of Canada Pundit	2.52E-06
69	Daimnation!	2.16E-06
70	Eschaton	2.09E-06
71	The Upper Canadian	2.00E-06
72	Instapundit.com	1.99E-06
73	No More Shall I Roam	1.99E-06
74	The Dominion Daily Weblog	1.96E-06
75	BlogsCanada : Canadian Politics Sites	1.93E-06
76	BlogsCanada : Canadian Politics Sites	1.93E-06
77	What it takes to win...	1.91E-06
78	BlogsCanada: E-Group	1.88E-06
79	Citizens' Assembly on Electoral Reform - IMPROVING DEMOCRACY IN B.C.	1.88E-06
80	Trudeaupia	1.87E-06
81	VanRamblings.com - reflections from vancouver, british columbia, canada	1.87E-06
82	OCCAM'S CARBUNCLE	1.85E-06
83	www.AndrewSullivan.com - Daily Dish	1.81E-06
84	Peace, order and good government, eh?: The people's medium	1.80E-06
85	Peace, order and good government, eh?: The people's medium	1.80E-06
86	Path of the Paddle	1.77E-06
87	Babbling Brooks	1.77E-06

88	Daily Kos	1.73E-06
89	Gen X at 40	1.72E-06
90	The Washington Monthly	1.68E-06
91	Ianism	1.63E-06
92	Gauntlet.ca - the politics of .ca: February 2005	1.59E-06
93	Gauntlet.ca - the politics of .ca: January 2005	1.59E-06
94	Gauntlet.ca - the politics of .ca: March 2005	1.59E-06
95	Sex, Drugs, and Rock and Roll	1.59E-06
96	Burkean Canuck	1.58E-06
97	Section 15	1.50E-06
98	The Armchair Garbageman	1.48E-06
99	sean incognito	1.47E-06
100	Footprints of a Gigantic Hound	1.46E-06

Table B-2: Top 100 Cathlolic Themed Blogs

Root Site: http://www.lovingit.co.uk

	Site	Page Rank
1	And Why Not?	1.40E-04
2	And Why Not?	1.31E-04
3	And Why Not?	1.31E-04
4	And Why Not?	1.31E-04
5	And Why Not?	1.31E-04
6	And Why Not?	3.53E-05
7	Catholic and Loving it!	3.09E-05
8	Zosh's online journal	2.76E-05
9	(Untitled)	2.68E-05
10	(Untitled)	2.09E-05
11	Zosh's online journal	1.84E-05
12	(Untitled)	1.75E-05
13	Zosh's online journal	1.60E-05
14	Zosh's online journal	1.60E-05
15	(Untitled)	1.49E-05
16	Zosh's online journal	1.40E-05
17	Zosh's online journal	1.18E-05
18	A Saintly Salmagundi	1.13E-05
19	The Curt Jester	9.09E-06
20	Zosh's online journal	7.77E-06
21	Musings of a Catholic Convert	7.12E-06
22	Dob-log	6.89E-06
23	Zosh's online journal	5.92E-06
24	Catholic and Loving it! : Archive : March 2005	5.72E-06
25	Gen X Revert	5.66E-06
26	Catholic and Loving it! : Entries by Ella	5.59E-06
27	Catholic and Loving it! : Archive : Book Reviews	5.59E-06
28	Catholic and Loving it! : Archive : Game Reviews	5.59E-06
29	Catholic and Loving it! : Archive : Juggling Stuff	5.59E-06
30	Catholic and Loving it! : Archive : Software Reviews	5.59E-06
31	Catholic and Loving it! : Archive : Religious Stuff	5.59E-06

32	Catholic and Loving it! : Archive : Film Reviews	5.59E-06
33	Catholic and Loving it! : Archive : April 2003	5.59E-06
34	Catholic and Loving it! : Archive : September 2003	5.59E-06
35	Catholic and Loving it! : Archive : May 2004	5.59E-06
36	Catholic and Loving it! : Archive : February 2004	5.59E-06
37	Zosh's online journal	4.78E-06
38	Zosh's online journal	3.90E-06
39	Bettnet - Musings from Domenico Bettinelli, Jr.	3.71E-06
40	Zosh's online journal	3.21E-06
41	JIMMY AKIN.ORG	2.75E-06
42	Zosh's online journal	2.68E-06
43	Meet Joe Convert - JoeConvert.com - Meet Joe Convert	2.34E-06
44	Why Catholic?	2.34E-06
45	Catholic and Loving it! : Archive : February 2005	2.30E-06
46	Catholic and Loving it! : Archive : February 2005	2.30E-06
47	Catholic and Loving it! : Archive : February 2005	2.30E-06
48	Catholic Light	1.96E-06
49	Danger! Falling Brainwaves	1.85E-06
50	Musings of a Catholic Convert	1.85E-06
51	Musings of a Catholic Convert	1.85E-06
52	Musings of a Catholic Convert	1.85E-06
53	Conversion of St. Paul: Today celebrates the co...	1.85E-06
54	Musings of a Catholic Convert	1.85E-06
55	Catholic Ragemonkey	1.55E-06
56	Disputations	1.44E-06
57	laodicea	1.39E-06
58	EveTushnet.com	1.36E-06
59	Dob-log	1.32E-06
60	Moleskine On A Bus	1.30E-06
61	De Fidei Oboedientia	1.26E-06
62	My Chcken Rantings	1.17E-06
63	Carpe Biem - Seize the Beer	1.17E-06
64	Oh Happy Day!	1.17E-06
65	Rosie's Blog	1.17E-06
66	Dob-log	1.17E-06
67	Dob-log	1.17E-06
68	Dob-log	1.17E-06
69	Dob-log	1.17E-06
70	Dob-log	1.17E-06
71	Dob-log	1.17E-06
72	Dob-log	1.17E-06
73	Dob-log	1.17E-06
74	Dob-log	1.17E-06
75	Dob-log	1.17E-06
76	Dob-log	1.17E-06
77	Dob-log	1.17E-06
78	Dob-log	1.17E-06
79	Dob-log	1.17E-06
80	Dob-log	1.17E-06
81	Dob-log	1.17E-06

82	Dappled Things	1.12E-06
83	Irish Elk	1.11E-06
84	Fructus Ventris	1.10E-06
85	Thrown Back	1.09E-06
86	E-Pression	1.08E-06
87	Summa Mamas	1.05E-06
88	man with black hat	1.03E-06
89	Heart, Mind & Strength - Blog Admin Panel	1.01E-06
90	Veritas	9.89E-07
91	moleskinerie	9.80E-07
92	Insight Scoop 2004	9.77E-07
93	A brushpen, a bat, and other cool things.	9.55E-07
94	Musings of a Catholic Convert	9.45E-07
95	Sed Contra	9.31E-07
96	The Mighty Barrister - Catholic Commentary Online	8.81E-07
97	Catholic and Loving it! : Archive : January 2005	8.75E-07
98	Dyspeptic Mutterings	8.45E-07
99	Shrine of the Holy Whapping	8.28E-07
100	The Blog from the Core - America's Small-Town Weblog	8.21E-07

APPENDIX C – DEPLOYMENT INSTRUCTIONS

Requirements

- Java Runtime Environment, version 1.5
- Apache Jakarta Tomcat Web Server, version 5.0

Installation Instructions

1. Copy the entire contents of the **/blogcrawler/tomcat** directory on the application CD to the root directory of your Tomcat installation.
2. Copy the contents of the **/blogcrawler/bin** directory to the hard drive.
3. Edit the file "settings.jsp" and change the value of the LUCENE_DIR variable to that the directory where you want the blog index to be located.

Usage

First, you must crawl the web to obtain a set of blogs to search from. To do this:

1. Change the directory to where you copied the files from /**blogcrawler/bin**
2. On the command line, start the crawler by typing the following:

   > crawl [DATA_DIR] [MAX_DEPTH] [THREADS] [URL1] [URL2] ...

DATA_DIR:	Directory to store the blog index
MAX_DEPTH:	Maximum depth to crawl as an integer
THREADS:	Number of threads to instantiate as an integer
URL:	The initial list of sites to crawl

   i.e. crawl c:\crawler\data 10 25 http://www.blogsite.com http://www.site.com

After the crawler finishes, generate the page ranks for the index by typing the following:

> pagerank [DATA_DIR] [URL1] [URL2]

Where URL1, URL2, etc. are the list of blogs to bias the rankings towards.

Finally, at any time, you may update the index with new content by typing the following:

> update [DATA_DIR]