The Web Beyond Popularity

A Really Simple System for Web Scale RSS

Daniel Gruhl
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120.
druhl@almaden.ibm.com

Daniel N. Meredith
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120.
dnm@almaden.ibm.com

Jan H. Pieper
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120.
jhpieper@almaden.ibm.com

ABSTRACT

Popularity based search engines have served to stagnate information retrieval from the web. Developed to deal with the very real problem of degrading quality within keyword based search they have had the unintended side effect of creating "icebergs" around topics, where only a small minority of the information is above the popularity water-line. This problem is especially pronounced with emerging information–new sites are often hidden until they become popular enough to be considered above the water-line. In domains new to a user this is often helpful–they can focus on popular sites first. Unfortunately it is not the best tool for a professional seeking to keep up-to-date with a topic as it emerges and evolves.

We present a tool focused on this audience–a system that addresses the very large scale information gathering, filtering and routing, and presentation problems associated with creating a useful incremental stream of information from the web as a whole. Utilizing the WebFountain platform as the primary data engine and Really Simple Syndication (RSS) as the delivery mechanism, our "Daily Deltas" (Delta) application is able to provide an informative feed of relevant content directly to a user. Individuals receive a personalized, incremental feed of pages related to their topic allowing them to track their interests independent of the overall popularity of the topic.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval; H.5.4 [**Information Systems**]: Information Interfaces and Presentation—*Hypertext/Hypermedia*
; J.m [**Computer Applications**]: Miscellaneous

General Terms

document routing, crawler, rss, webfountain, internet

Keywords

Daily Delta, RSS, WebFountain

1. INTRODUCTION

As the web has expanded over the last dozen or so years, the means of finding timely and relevant information from it have evolved in an attempt to cope. The early web was not unlike many fringe cultures– the only way to find out about sites was from someone "in the know". Recommendations of sites of interest were passed by word-of-mouth, either literally, or through people posting URLs in other mediums. While this approach provided highly contextualized information, it also severely limited the scope of what an average person might be able to find.

Web scale search engines shattered the notion that no one system could index such a large, distributed collection of information. By providing keyword query access to web pages they freed users from having to rely on their social network to provide a relevant starting point for browsing. This worked well until a combination of hyper-growth in the volume of information, coupled with malicious commercial duplicity (i.e., spam) flooded these simple keyword based engines with noise to the point where their utility was seriously degraded.

In response to this problem we saw the development of global recommendation algorithms in the form of popularity based ranking in search engines (i.e., HITS[11], PageRank[6], Clever[26], etc.), which leveraged the nascent hyperlink structure of the web to provide a simple voting mechanism as to the relevance and quality of page content. While this did much to address the noise problem, the transition to popularity based ranking resulted in the loss of two of the more endearing features of the early web: the constant stream of *new* material regarding an issue of interest to you, and the notion that a page only has to be *popular* to you to be relevant.

In an attempt to recapture these aspects of the web user experience, we have looked to another recent phenomenon: blogs. Over the past few years there has been an explosion in personal publishing on the web. Blogs, bulletin boards and wikis provide simple mechanisms for individuals and collaborators to create content that is highly relevant to a given community. These sources comprise a large portion of the "incremental web" that are not *possibly* changing on a regular basis, but actually are intended to be updated frequently. The challenge of keeping a reader up to date with this changing information has led to the use of syndication systems, notably Really Simple Syndication (RSS)[31] and Atom[28]. These provide a mechanism to deliver content along with a notion of how frequently that content is changing directly to a user. A large class of applications have emerged that allow such syndicated streams to be opened, aggregated and presented to a user in a number of ways, on a variety of devices.

Is it possible to bring this sense of immediacy and relevance that are the hallmark of blogs to the web as a whole? We have created a system using the WebFountain[15, 24] platform to provide Web Scale RSS feeds. Built on top of this system we have explored several applications which utilize these feeds in a number of user scenarios. In addition, we address some of the challenges in distilling high quality content from the changing web.

The remainder of this paper will illustrate the problem and outline our approach to the system (Section 2), provide some detail on our implementation (Section 3), present performance and quality results on our system (Section 4), examine related work in the content acquisition, routing and presentation spaces (Section 5), and lastly close with some concluding thoughts and potential for future work (Section 6).

2. APPLICATION AND APPROACH

Alice and Bob are two individuals with particular information needs. Alice is a researcher in the field of text analytics, and is disappointed to find that there are no good discussion forums on the topic. She is looking for a way to stay up-to-date on this emerging field, and is looking for a few dozen articles to read on any given day. If she finds some important ones she would like to be able to save them, forward them, etc.

Bob is an up-and-coming bartender, and is always looking for quirky new drink recipes. He specifically wants to find ones that are not yet mainstream. He'd like to scan a few hundred of these recipes every day looking for inspiration. It would be great if he could do this task on the bus on the way to work.

Neither Alice nor Bob are going to be happy with the state of the art in modern alert systems as they tend to be somewhat limited. They either restrict their alerts to a limited set of documents (wire feeds, news, etc.) or they only consider pages that rank highly. Google Alerts[21], for example, requires a page to be one of the top 20 for the equivalent query within the Google search engine. While there are over 28,000 hits for the phrase "text analytics" on Google, there are only a few dozen new pages appearing on the topic every day. While the top results for an emerging topic may change while the topic initially develops, once a topic is stable and authorities have been established, the membership of the top result set also stabilizes. This produces a fairly static list that does not report new content that is still being generated on the topic.

One caveat is that the popularity of a page or site can be considered independent of a specific topic. This means that web pages which have a generally high rank will appear in the top result set for any topic on which they report. While this permits new pages to enter the top results for a given topic, only content from a limited, high popularity set of sites (Slashdot, CNN, etc.) can do so. Delta addresses the need to report on content being produced from sites outside this limited set.

Additionally, no single presentation tool is going to meet the needs of Alice and Bob. Dozens to hundreds of alerts a day is going to be too much for an email delivery system, but Alice wants well developed document handling, while

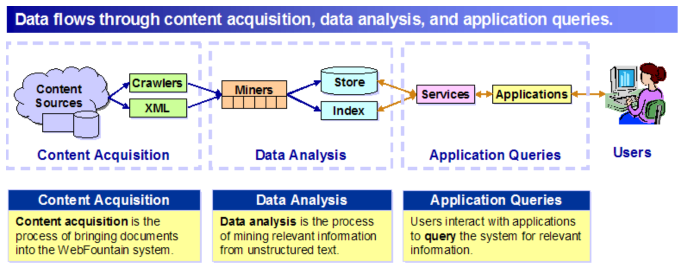

Data flows through content acquisition, data analysis, and application queries.
Content Sources
Crawlers
XML
Miners
Store
Index
Services
Applications
Content Acquisition
Data Analysis
Application Queries
Users
Content Acquisition
Content acquisition is the process of bringing documents into the WebFountain system.
Data Analysis
Data analysis is the process of mining relevant information from unstructured text.
Application Queries
Users interact with applications to query the system for relevant information.

Figure 1: Basic information flow in the system. The filtering discussed later can be implemented either as a service (for query based) or in the mining chain.

Bob wants a presentation that enables rapid scanning, with preference for a mobile environment.

Developing a system that can keep both Alice and Bob happy is conceptually quite simple, but an efficient implementation is an engineering challenge. The success of the system hinges on the ability to deliver alerts of sufficient quality to make the system useful, despite often high levels of noise in the source information feeds.

The overall system employed is fairly straightforward (see Figure 1). Content is gathered from a variety of sources, depending on the configuration of the system–it can be stored and indexed for later query, or filtered and routed in real time. This is not unlike many existing information aggregation services, consider the ticker tape and news wire services that have existed for decades[8], but we have extended the paradigm by allowing aggregation of arbitrary content on arbitrary topics. By allowing users to define the selection and filtering criteria, the content which is routed to the user does not have to be limited to sites which already provide a feed, and in theory the content does not have to be directly accessible on the web. Any content which is gathered by the system can be encapsulated in RSS and provided to the users.

The characteristics of a good document are easily defined with respect to web data. It must not be spam or adult content, the document must contain a reasonable amount of information which pertains to the topic in question, etc., but within the scope of our target user's goal, the bar is set much higher. Documents must be topical, unique, and present information which is new to the individual user. This personalization aspect creates a new type of problem that is not handled well by a typical search engine. Additionally not all users are created equal–what is good for one user with respect to a general topic may not be good for a different user within the same general topic. Thus we have the requirement to provide complete, personalized ranking and routing of documents to each individual user. Implementing such a system in a way that delivers sufficient accuracy in an efficient, scalable manner is the challenge we undertake and lay out in the following section.

3. IMPLEMENTATION

Delta explores the issues that arise from web scale feeds. Delta is built on top of the WebFountain platform, which queries over 5.6 billion pages, gathers millions of new documents a day and stores over a petabyte of data which has been annotated with dozens of classes of new metadata[15, 24]. Using this system, we have implemented the new selection, filtering, caching and presentation features that are needed to realize Web Scale RSS.

3.1 Content

Alice and Bob's information needs are different, but both require an information feed that is broad enough to locate emerging sources, and fresh enough to assure that the information they are provided is actually new.

Providing this source content employs the field of web crawling or harvesting, one that has been well explored by previous research efforts (see Section 5.1), so this section will focus on those requirements that differ from those of a general web search engine.

As with any content-based system, the quality of output can be directly correlated to the quality of the input data, and while there are various algorithms for filtering and removing low quality data, such as spam, adult content and duplicates from a corpus, a solid web crawling strategy strives to avoid ingesting this type of content in the first place. The largest differentiators in our crawl needs are *coverage* and *freshness*.

Providing this coverage and freshness means that we cannot rely on the link structure of the web to guide our crawl–we are trying to gather good content that is not yet heavily linked to. Nor can we use an individual focus crawl technique[12], as the resource cost of training a focus crawler for every possible feed is beyond reasonable capabilities of most systems, and introduces a serious scaling limitation if the goal is to support hundreds of thousands of feeds. The need for freshness is derived from the basic user expectation of Delta that content routed to them will be not just new to them, but also new to the web.

The next sections will outline our strategies to deal with each of these issues and subsequently gather the best content for our users.

3.1.1 Finding quality sources

The first strategy for content gathering is broad coverage of sources which are likely to contain content of interest to users. The ideal solution would be to crawl everything every few minutes and remove junk and duplicates, but this is obviously not feasible unless the system has infinite machines and bandwidth, and it doesn't mind degrading the performance of the greater Internet.

The opposite extreme would be to exclusively use a focused crawler[12] to gather documents which are relevant to known user topics of interest, and only follow links which are likely to lead to more relevant documents. However, this can be limiting as it becomes necessary to seed each new topic, and content which is not connected to the seeds may be missed. This skipping of "not relevant now" content is not acceptable as a good corpus of pages is required for the feed bootstrapping process to ensure that a user is presented with content immediately upon creating a new feed.

We use a hybrid solution of a broad crawl to examine links to all sources, but then prioritize each source based on heuristics such as its IP address, anchor text of links, and a sample of the source's content. This source-level classification is reliable at identifying sites which are likely to be adult content, spam, and other junk content, as well as identifying sources which may contain relevant content[19].

3.1.2 Finding fresh content

In addition to our need for broad coverage, Delta also requires a constant stream of content which is new to the system, but ideally also new to the web. We want to find fresh content as soon as it is published on the web, but again it is unrealistic to attempt to re-crawl every page constantly to look for changes.

This problem can be partially addressed by looking for link hubs–pages which are frequently the source of new links to new and relevant content. Site hub pages for a news site might include the front page, while hub sites for a discussion forum might be a list of the current discussions. Hub pages are measured by the frequency at which they provide new links. In addition, they are prioritized by the probability that the new links will have high quality content, as discussed in the next section.

Hubs are crawled using a dynamic re-crawl policy which crawls the more important and frequently changing hubs more often. The top hubs should be crawled daily, hourly, or whatever the maximum change rate is, at which the crawler expects to find new content[17]. New content from hubs is crawled immediately.

Our system is also able to exploit the increasing number of feeds presented by various content providers. When a feed is discovered, its content is crawled by a specialized ingestor, which improves timeliness and confidence in various metadata, such as date and author.

3.1.3 Query directed crawl strategy

The third strategy is to take hints from user queries to guide the crawling[30]. User preferences can be measured in three ways. First, any document which matches a user query is marked as potentially relevant. Second, users can be asked to manually mark any documents, which they find especially relevant or objectionable. Lastly, sites known to be rich in terms and entities of interest to users can be considered for inclusion in the high value set.

Based on this input, both sources and hubs can be prioritized based on the past likelihood of containing good content. While this is not a true focused crawler, it does produce bias in the crawl frontier and therefore improves efficiency in resource utilization.

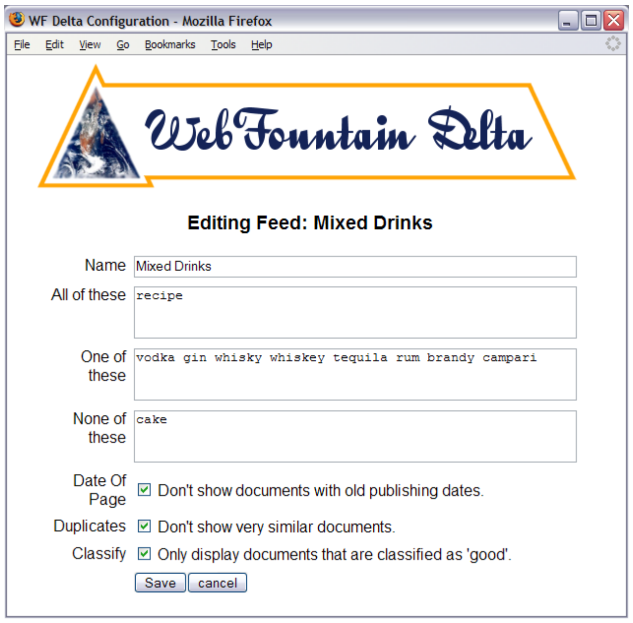
3.2 Document selection

Document selection is the process of determining which documents are relevant for a particular feed. While the goal of gathering is to assemble, with sufficiently high recall, all of the pages that *might* be relevant, selection tries to raise the precision of the documents shown to the user as high as possible. The web provides the traditional problem of a very low percentage of *good* documents compared to *noise* in any particular area. This means that false positives are a reality, and the system should present them in a way which allows the user to dismiss them as quickly as possible.

Given the large number of documents being considered (millions per day when running, potentially billions while defining a new feed) a tiered approach is employed. First, the initial feed selection criteria is defined. Then, given this smaller set of documents, traditional filtering and routing techniques are applied to increase the accuracy of a given feed.

3.2.1 Initial feed definition

Before Alice and Bob begin to receive articles from Delta, they need to start by defining their respective topics of interest. We've chosen to bootstrap this by providing them with a simple query interface. This has the advantage that


WF Delta Configuration - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
WebFountain Delta

Editing Feed: Mixed Drinks

Name	Mixed Drinks
All of these	recipe
One of these	vodka gin whisky whiskey tequila rum brandy campari
None of these	cake
Date Of Page	☑ Don't show documents with old publishing dates.
Duplicates	☑ Don't show very similar documents.
Classify	☑ Only display documents that are classified as 'good'.

Figure 2: A simple query interface is provided for initial feed selection.

users already know how to operate such a system and are thus intuitively able to define new feeds.

While this approach is fairly natural, it has the issue that queries tend to be overly general in the sense that the average query length is 2.4 words[34] and overly specific in the sense that users may not specify all possible terms that actually define the topic. To address this, while preserving the simple query approach, we employ a method of query expansion, although it would be better described as *document expansion*.

The documents are annotated with additional higher level metadata through a variety of sources. Included in this metadata is a taxonomy tree of entities which groups people, places, things, etc. into a hierarchy. We allow users to select these higher level concepts, and automatically capture all the documents that include any of the lower level concepts.

Executing these queries to generate candidates can be performed in one of two ways: an index query based approach and a mining based approach.

Index Query Based Approach

The WebFountain platform allows us to rank and sort query results by system ingestion date, thus creating a list of new pages that match a feed definition independent of popularity. This approach works well for prototyping new feeds, but has several drawbacks.

First, it is dependent on the index update rate. New pages only become visible after they have been added to the active index. WebFountain refreshes its full web indices once daily, which leaves us with a delay of up to 24 hours between ingestion of a relevant page and it being available through the index. This is similar to the slow feed approach seen in[10].

Secondly, updating feeds through index queries requires the system to periodically check if new content has come available. This can be done whenever the RSS file is polled by a presentation system, but since many of these systems check frequently, the result can be hundreds of queries per day per feed on the system. While caching can address some of this load, it is more efficient to design the system to examine the documents for feed membership as they enter the platform.

Mining Based Approach

Instead of relying on the index for updating existing feeds, we employ a feed miner that analyzes new documents as they are streamed into to the system by the crawler. Our *Delta Feed Miner* loads the feed definitions into a trie memory structure[18] that allows matching a large number of terms against a document with very little overhead. Documents which match a given query in the trie are considered part of the initial candidate set for said query. Candidate documents are then routed to a secondary analysis and filtering phase which determines the final set of documents. This approach is more scaleable and removes the delay in updates resulting from index update latency.

The mining based approach is truly forward looking and will only deliver newly ingested matching documents. Thus it is less than ideal for designing and populating new feeds. The user will not receive immediate feedback on the quality of the feed definition. Therefore, our system relies on a hybrid approach of index queries to populate new feeds with a sufficient number of initial pages. This enables the user to refine the feed until it captures the user's intent. Once defined, a feed is incrementally updated using the mining based approach. The user can revisit the feed definition and refine it at any point using the same hybrid method.

3.2.2 Filtering

The feed definitions are used to create a list of candidate documents. These sets tend to be much smaller than the full set of newly ingested documents, and are thus more suitable for traditional filtering and routing approaches. Our filtering phase is designed to address the following issues:

Duplicates

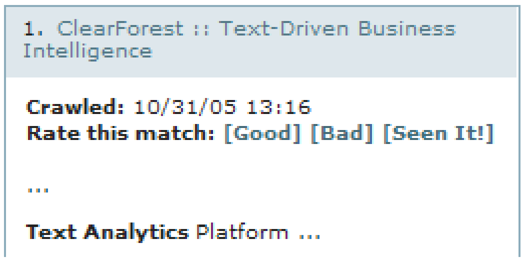
Without duplicate detection, the system re-suggests documents that have been modified, but not modified in a manner relevant to the feed definition. For example, if a new paragraph is added to an online newsletter, the page is considered updated by the system. But if the new paragraph is not relevant in a new way to a given feed it should not be re-suggested to a user.

To address this issue the system keeps track of all pages that have been selected for a given feed. A fingerprint[7] of the relevant content for each selected document is associated with the feed. All future candidate documents must provide a unique fingerprint in order to be selected for the feed.

Old content

The crawler may encounter a previously uncrawled page that contains dated content. Sometimes this is acceptable–such content may be new to the user. However, for news-like feeds it presents a problem.

WebFountain provides a *DateOfPage Miner* which attempts to determine the publishing date of a document by analyzing the document content. For example by spotting a date in


1. ClearForest :: Text-Driven Business Intelligence

Crawled: 10/31/05 13:16
Rate this match: [Good] [Bad] [Seen It!]

...

Text Analytics Platform ...

Figure 3: In feed feedback mechanism to train classifiers.

the headline of a news article or relying on the *last modified* date as reported by the web server. We use the metadata created by the *DateOfPage Miner* to remove old content where possible. However, there is a large set of pages that do not provide any hint of their publishing date and thus remain on the candidate list.

Templates
It is also possible for a site template (per page replicated content) to match a feed definition. These matches are of little or no value to the user and must be excluded from selection. We employ a template identification algorithm to restrict feed matching to the core text of a document. The algorithm divides a page into a series of logical blocks and then uses a variety of heuristics on each block to determine if the block resides within a page template.

Classification
The previous three filters are targeted at removing spurious content from the candidate document set. Another issue is that the feed definition mechanism is fairly rudimentary and does not allow precise topic specification. We provide further refinement of a feed by applying a classifier. The classifier is trained by the user via a simple feedback mechanism. We selected Naive Bayes[1] classifier, as it is known to perform well with relatively little training data. Obvious future work is to explore other more complex filtering approaches.

Using these tools Alice and Bob can select which post-filters are appropriate (see Figure 2), and then begin providing feedback by in-feed links (see Figure 3) to increase the accuracy of the results. The total document selection process is summarized in Figure 4. We may apply filters in a slightly different order depending on cost and effectiveness.

3.3 Feed presentation

Alice and Bob are planning on using their Delta feeds in very different ways. Alice is looking for a way to work through dozens of fairly targeted and personally important web pages. Bob is more interested in scanning hundreds of pages a day looking for inspiration. Supporting both of these led us to a display independent method of presenting information–in our case RSS.

The choice of RSS as a delivery mechanism both greatly simplifies and greatly complicates the task of creating the best presentation of the results. Traditional feed viewers presuppose a certain level of quality of the feeds–while people do write useless or irrelevant content in their blogs it is assumed if this is the rule for a source the user will simply unsubscribe from the feed.

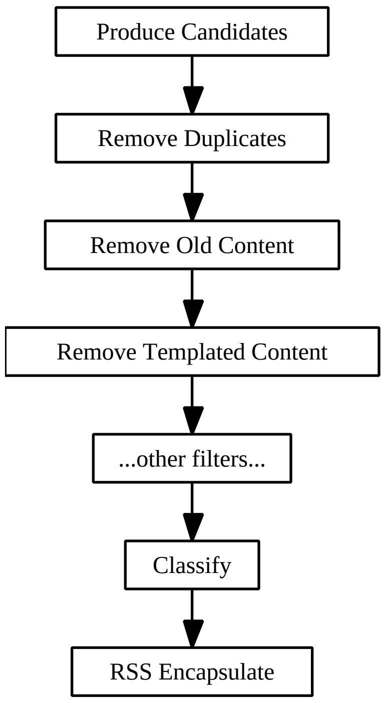


Figure 4: Document Selection

The presentation of Delta information from the web is an inherently noisy process. A large fraction of the matches it produces may be irrelevant. In addition, we need to enable our feeds with a way to let users share their relevance concerns with the system, providing the feedback needed for the traditional filtering and routing problem.

3.3.1 *User Interface Considerations*

Designing an interface which allows for personalized document ranking and routing when one doesn't control the reader is a challenge. How do you create lightweight feedback mechanisms within the system that will be supported generically by an arbitrary information conduit?

One of the most important features of RSS is that a user can choose from a number of different interaction paradigms when working with feeds. From simple online readers like Google Reader[23] or Bloglines[3], to in browser plugins such as Sage[32], there is a wide range of existing readers that present users with options as to how a feed will be presented. There are also tools which mimic the email and older NNTP style of working with the feed[37, 38, 41]. And with the recent release of Flock[36] there appears to be a new generation of personalized web clients, which merge browsing, editing and publishing content. While this is a boon to users and creates an extremely flexible delivery mechanism for the system, it also constrains the facilities available for user feedback to the system.

In designing the user experience, some assumptions must be made as to what basic capabilities of the feed readers must first be established before the mechanism of feedback can be considered. The basic function of a reader is to parse the XML of a feed, present a user with the content of the feed and allow them to browse from the feed to the original content or links contained in the feed. From this, it can be assumed that HTTP hyper-link support is present in the majority of reader clients. This assumption is reinforced by the RSS specification[31], which explicitly states that entity encoded HTML is allowed with the description element of a feed item.

Beyond simple HTML rendering and link support, more complex feature of the modern web page such as CSS or JavaScript cannot be assumed to be under the control of the feed producer and therefore should not be relied upon when designing our feedback mechanism.

3.3.2 Feedback Mechanism

Given the technological constraints of our reader capability assumptions, all feedback in the system is handled by inserting HTML links and some minor layout code into the description field of a feed item (see Figure 3). The action of a user clicking a feedback link is registered by the system and accounted for internally. We recognize that there could be privacy concerns stemming from a user feedback system of this type. This is partially mitigated by sending all information back to the system as plain text in the link itself so that a user can easily tell what information is being recorded, and the information itself is sent via an SSL HTTP connection to ensure that third-parties are not able to harvest the information in transit.

3.3.3 Feedback Properties

The first element of feedback is a good or bad determination. The *Good* link signifies that user liked the document and would like to see others like it. The *Bad* link indicates that the document was not what the user was looking for from this feed. While this lacks granularity, it is simple enough for most users utilize effectively and provides the necessary feedback to train the classifier.

The second element of desired feedback is whether or not a user has seen a routed document before. This is accomplished by the *Seen it!* link in the feed item. This serves two purposes: 1) feedback on false negatives within duplicate detection and 2) producing a list of sites that a user looks to for information about a given topic. The system does not have *a priori* knowledge of every document the user has read during their web browsing lifetime, and therefore can only build the required knowledge base via user feedback. Implicit feedback can be harvest by monitoring which documents in the feed a user views, but the *Seen it!* mechanism allows a user to indicate that a document is a prior view without having to browse away from the feed list.

3.4 Deployment

One of the advantages of RSS is the simplicity of deployment. Once the RSS file is created it is merely placed on a static web server and fetched as needed. This means that, with the exception of user feedback, there can be a strong decoupling of the system from the user load. A single feed server machine can support tens of thousands of users, since the feeds can be generated by the back-end system at opportunistic times.

	Pre-filter		Post-filter	
	Count	Precision	Count	Precision
Text Analytics	69	.81	47	.89
Mixed Drinks	5906	.41	760	.82

Table 1: Document Count and Precision before and after filtering in for the two examples.

4. EVALUATION

There were four concerns we had going into the Delta project:

- Would there be enough information to make up a feed on a daily basis? Our experience with various alert systems had been that we received very few alerts per day.
- Once we got the information, could the feed be selected and transformed so that it had a high enough relevance. Our rule of thumb was that three out of every four articles should be relevant (75% precision).
- Could we present feed information effectively in existing viewers, given the constrained format and generally lower data quality?
- Could the feeds be created in a way that we could support tens to hundreds of thousands of users?

4.1 Feed Characteristics

We found that the characteristics of feeds can vary greatly. Figure 5 shows the number of documents in the system by week for the time period of September 2005 through early November 2005 for each of our example feeds. We also provide a breakdown by day of the first week in November.

Alice's "Text Analytics" feed is a good example of a fairly targeted one. The selection criteria is unambiguous and expressive, and the feed results support this. The Text Analytics feed turns out to be a low-volume feed, with under a hundred postings a week and tens a day.

Bob's mixed drink feed had to be defined a little more carefully, as "mixed drink" as a query does not work as well as might be hoped. We quickly discovered that the term "recipe" with one or more common drink ingredients and a negation of "cake" did a good job of generating a reasonably on-topic feed (see Figure 2).

This mixed drink feed represents the other extreme–first, the volume is significantly higher. With thousands of postings weekly and hundreds every day (see Figure 5) it is unlikely that it will be possible to read all of them. In fact, it is fairly clear that this feed is overly broad, unless you are interested in looking at trends (e.g., the large peak in posts at the beginning of the college school year in the weekly chart. . .). For Bob this is fine as he is merely looking for something to scan, but for more serious applications there is an obvious information overload problem which needs to be addressed here.

4.2 Precision

Both of these initial feeds have a certain amount of noise, and we were interested in how precise a cleaned up stream could be produced. We performed an evaluation of the precision for the best results we could get through tuning the

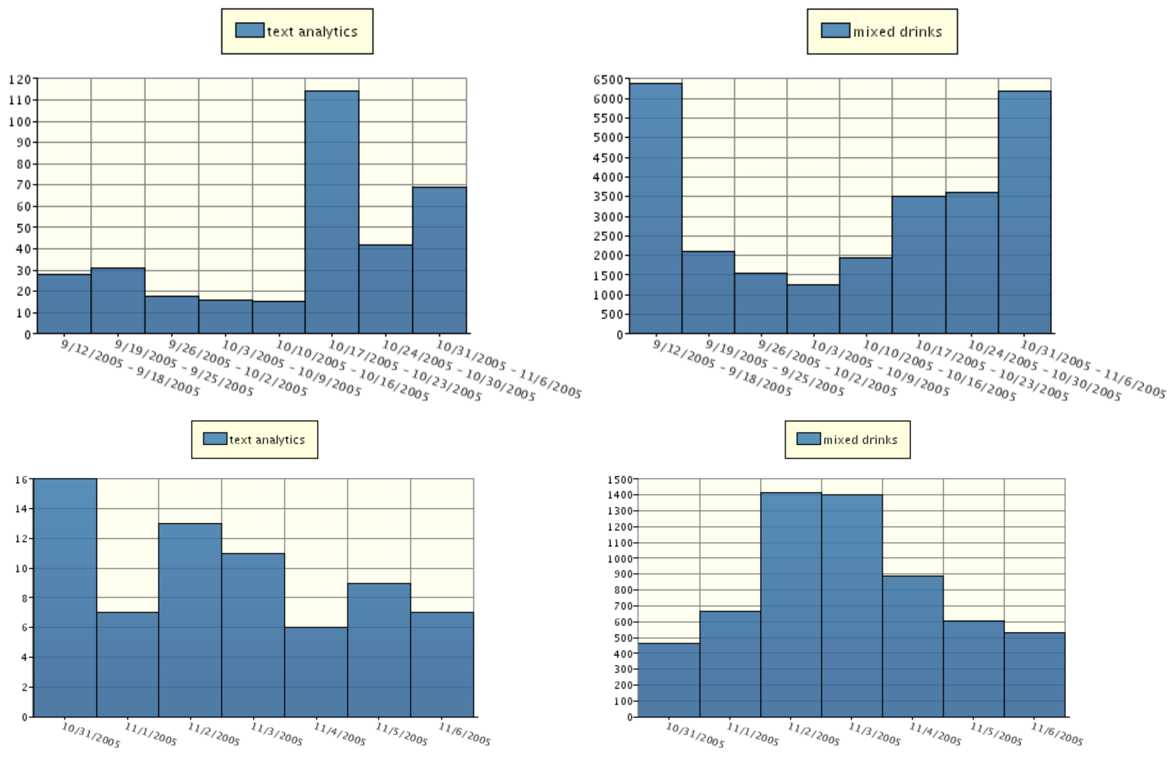


Figure 5: Pages per week and per day on the Text Analytics and Mixed Drinks feed.

boolean selection criteria for the two sample feeds discussed in this paper, and then for the post-filtered feeds using a Naive Bayes classifier with a 40 document training set.

For the week of October 31st, from an initial new document set for this experiment of 160,594,704 documents, query selection produced 69 documents for Alice's "Text Analytics" feed. As can be seen in Table 1 the precision before cleanup was a respectable 81%. Not surprising since the phrase "text analytics" is fairly selective. After post filtering we saw the feed dropped to 47 documents, but the precision is boosted to 89%.

The mixed drinks feed started from the same initial feed. After boolean selection this dropped to 5,906 documents with a precision of only 41%. This could probably be raised slightly with more complex subqueries, but such complexity quickly goes beyond our assumed skill set for users such as Bob. Post filtering reduced the document set to 760 but boosted precision to 82%. It seems that there are a few general classes of bad documents, which when removed quickly bring the quality up to a more acceptable level.

We did not explore this much further as the field of document filtering is well established [5, 9, 10] and more meaningful numbers will require developing larger, scored representative corpora. This is an important future work item for us, if only to vet the various existing filtering techniques for use in this domain.

4.3 Presentation

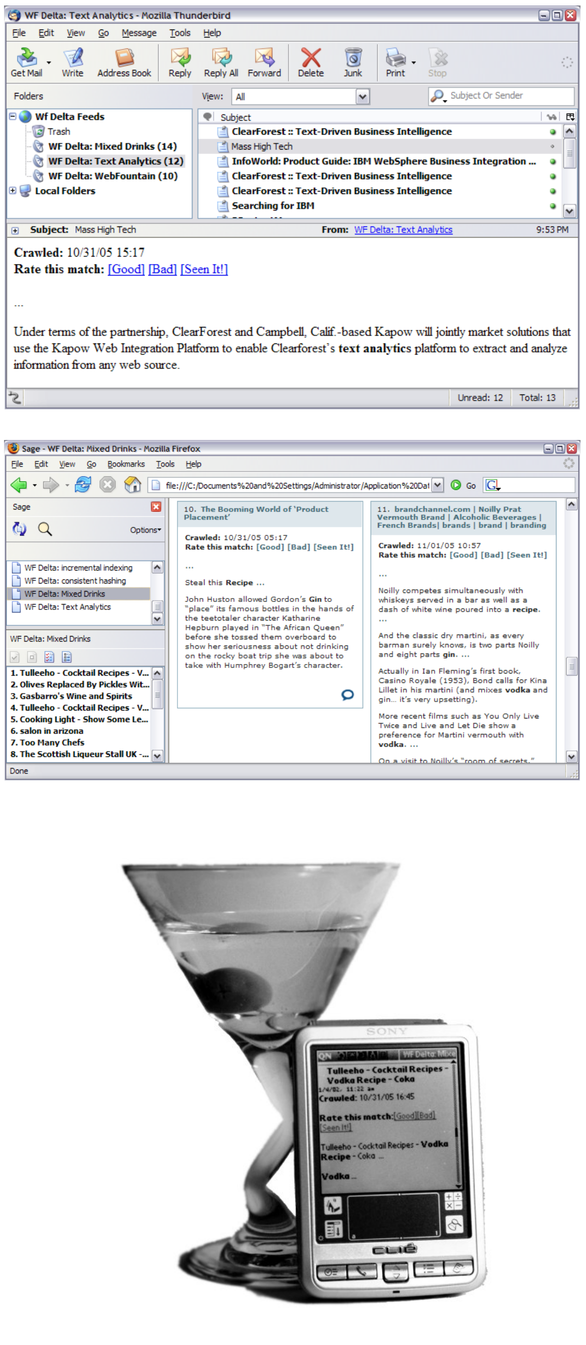
While precision is an important aspect of usability, the right presentation and user interface often make or break an application. Alice and Bob's very different needs illustrate that one size most certainly does not fit all for the presentation. Fortunately there are many RSS readers available.

	Index based	Mining based
Initial set	1.4-16.4s/query	25-30ms/item
Filtering	15-19ms/item	
Formatting	78-127ms/item	
Serving	tens of ms	
Total (20 items)	≈ 19s	≈ 6s

Table 2: Where the time is spent on providing these feeds.

Alice needs to work in detail with the results of the "Text Analytics" feed. Feeds of this nature are often best thought of as mid-traffic "net-news" like data sources–you want to read all of the traffic on them, and may want to save the pages and/or forward them to colleagues for further consideration. This speaks to a newsreader like tool such as the one included in Thunderbird (see Figure 6). On such a small feed enabling duplicate removal is not always appropriate–it is actually useful to know that the ClearForest announcement is getting broad coverage.

As noted above, Bob's mixed drink feed is broad. In terms of display, the Thunderbird-style reader might not be the best bet. Fortunately, other options exist, such as "in browser" readers like Sage (see Figure 6) that let you quickly scan the results (note the use of sentence snippeting to allow rapid browsing). Of course, some of these feeds are more useful when you are not sitting in front of a computer. For these occasions PDA based readers[35] that allow you to take these feeds with you for easy reference (see Figure 6) might be a more appropriate choice.


WF Delta: Text Analytics - Mozilla Thunderbird

Subject
ClearForest :: Text-Driven Business Intelligence
Mass High Tech
InfoWorld: Product Guide: IBM WebSphere Business Integration ...
ClearForest :: Text-Driven Business Intelligence
ClearForest :: Text-Driven Business Intelligence
Searching for IBM

Subject: Mass High Tech From: WF Delta: Text Analytics 9:53 PM

Crawled: 10/31/05 15:17
Rate this match: [Good] [Bad] [Seen It!]

...

Under terms of the partnership, ClearForest and Campbell, Calif.-based Kapow will jointly market solutions that use the Kapow Web Integration Platform to enable Clearforest's **text analytics** platform to extract and analyze information from any web source.

Sage - WF Delta: Mixed Drinks - Mozilla Firefox

1. Tulleeho - Cocktail Recipes - V...
2. Olives Replaced By Pickles Wit...
3. Gasbarro's Wine and Spirits
4. Tulleeho - Cocktail Recipes - V...
5. Cooking Light - Show Some Le...
6. salon in arizona
7. Too Many Chefs
8. The Scottish Liqueur Stall UK -...

10. The Booming World of 'Product Placement'

Crawled: 10/31/05 05:17
Rate this match: [Good] [Bad] [Seen It!]

...

Steal this **Recipe** ...

John Huston allowed Gordon's **Gin** to "place" its famous bottles in the hands of the teetotaler character Katharine Hepburn played in "The African Queen" before she tossed them overboard to show her seriousness about not drinking on the rocky boat trip she was about to take with Humphrey Bogart's character.

11. brandchannel.com | Noilly Prat Vermouth Brand | Alcoholic Beverages | French Brands| brands | brand | branding

Crawled: 11/01/05 10:57
Rate this match: [Good] [Bad] [Seen It!]

...

Noilly competes simultaneously with whiskeys served in a bar as well as a dash of white wine poured into a **recipe**. ...

And the classic dry martini, as every barman surely knows, is two parts Noilly and eight parts **gin**. ...

Actually in Ian Fleming's first book, Casino Royale (1953), Bond calls for Kina Lillet in his martini (and mixes **vodka** and gin... it's very upsetting).

More recent films such as You Only Live Twice and Live and Let Die show a preference for Martini vermouth with **vodka**. ...

On a visit to Noilly's "room of secrets,"

WF Delta: incremental indexing
WF Delta: consistent hashing
WF Delta: Mixed Drinks
WF Delta: Text Analytics
Tulleeho - Cocktail Recipes - Vodka Recipe - Coka
Crawled: 10/31/05 16:45
Rate this match: [Good] [Bad] [Seen It!]
SONY

Figure 6: The text analytics feed in the Thunderbird reader. The Mixed Drinks feed in the sage viewer, and a PDA based reader. Sometimes the right presentation format makes all the difference.

4.4 Performance

We observed that total feed time ran to a worst case of around 19 seconds for the query based approach, or six seconds for mining based feeds (see Table 2 for breakdown). A bigger differentiator is that the query based approach loads the entire cluster, as opposed to the mining based approach which can be implemented with scale-out parallelization. Our delta feed miner checks a page against all existent feeds at the same time, whereas the query based approach is also an inherently sequential process.

This results in a striking difference in the number of feeds which can be supported. The query based approach is lower bounded at about 4500 feeds. Employing the cluster to do the mining during page ingest, the final filtering can be done in an off-cluster scale-out architecture. To give an idea of what can be achieved, at 3 seconds per feed per processor per day, a 14 dual processor machine Blade Center is able to maintain roughly 800,000 feeds with daily updates. Obviously tuning and performance implementations of the filters could raise this number substantially.

Once the feed files are created the task of serving the RSS feeds is a trivial static page serve problem, and we have established that a single Apache[2] server can support tens of thousands of users a day.

In short, this approach to producing Daily Deltas is amenable to fairly broad deployment, even in the face of a high degree of personalization in the content provided.

5. RELATED WORK

The concept of providing an alert-style feed is not new–clipping services provided essentially identical functionality for over a hundred years. What is new with Delta is the scope of the data set considered, and the ability to personalize at this scale. Accomplishing this required us to focus on three main technical areas; the crawl, the document selection process, and the presentation aspects.

5.1 Crawling

We considered several well known strategies to improve the quality and breadth of the documents gathered by our crawler: Focus crawling [12], a broad web crawl[17] and feedback from the query system to the crawler as e.g. described in [30]. Site level analysis is well covered by [19], and self similarity is explored in [16]. Identifying no-longer fresh sites is explored in [4], and identifying leading blogs is in[25].

5.2 Document Filtering and Routing

Document filtering distinguishes itself from information retrieval in that it usually employs statistical methods, which generally lead to better precision and recall compared to unranked boolean systems. Moreover, a filtering algorithm must be designed to make routing decisions without knowing what documents will be seen in the near future.

A lot of research has been done in the area of document filtering and routing[5, 9, 10, 42, 13, 27, 14]. The focus is on high speed, stream based filtering and better selection of relevant documents. Comparing and improving filtering algorithms is at the core of this discipline.

In this paper, we present a framework for a web-scale document filtering and routing system. We demonstrate the ability to use filtering algorithms with a simple Naive Bayes classifier[1], which could be swapped with a number of more sophisticated algorithms. Our query driven system (see Figure 1) is considered "slow filtering", whereas the in-line filter based system is more similar to traditional document filtering and routing systems

Duplicate removal is by shingle[7], and template identification follows from[20].

5.3 Topical News Feeds

Information and media monitoring services are not a product of the Internet. Firms such as Burrelles*Luce*[8] have been providing media monitoring to clients for over a century. In an extension of news wire services, companies such as Topix.net[39] have created web deployed news aggregation and feeds. While Topix.net does offer over 300,000 individual feeds to its users, it specializes in news content which is targeted to users based on geography and a high level topic. Topix.net does not offer feeds on arbitrary user topics, nor does it provide mechanisms for feed personalization to the user (e.g., we could find no Text Analytics, nor Mixed Drink recipe feed).

5.4 Presentation

Providing news on-line has gone through a number of presentation approaches, from the "picture of the front page"[29] approach to Yahoo's headline lists[40] to Google's news portal page[22], each approach has it's strengths and weakness. Having tried many of them we've found that the overall low quality of the web feeds compared to traditional wire service feeds requires a format that is easier to scan. The San Francisco Chronicle site SFGate[33] provides an approach that facilitates this rapid scanning (and is similar to the Sage[32] newsreader). We are working on a similar display format for Delta, but for now are contenting ourselves with existing RSS readers such as those found in Firefox and Thunderbird[37], as well as various Palm RSS readers such as the Quick News reader from Stand Alone[35].

5.5 Query based RSS feeds

There are a number of email and RSS based alert systems available on the Internet. To our knowledge none of these systems provide truly personalized results on web scale. They are either based on one of the popular search engines, including Google's own "Google alerts"[21] or specialize on a specific segment of the web, e.g. Topix.net as discussed in Section 5.3.

Search engine based feeds usually forward only pages that are ranked in the top N results. For example, in Google Alerts documents have to be in the top 10 for news, the top 20 for the web or the top 50 for newsgroups to be forwarded. In contrast, Delta will consider the relevance of any newly crawled document with regard to each of the user defined feeds.

6. CONCLUSIONS AND FUTURE WORK

In attempting to turn the Web into a Daily Delta feed we have encountered a number of challenges, from scope and scale through quality and specificity. We found that a multi-level approach to document selection, coupled with an integrated user feedback system and delivery through standards based feed mechanisms, such as RSS, allows the creation of useful feeds in a large number of areas. It should be noted that the exclusion of popularity based ranking does make the task of selecting only documents above a certain quality level vastly more difficult. While this is not a new problem, it must be combatted for systems such as Delta to meet user expectations.

The combination of RSS and mining based document selection provides a low cost mechanism to support a large number of feeds. Users may employ the interface best suited for their particular needs.

Future work includes exploring how other filtering approaches apply in this domain. We expect that many of them will need to be tuned to the lower quality, higher volume conditions found on the web. We have seen strengths and weaknesses in existing RSS readers with this class of content, and are looking to provide alternate interfaces that may be more appropriate.

We have focused on personal feeds, but there is an obvious extension to include community collaboration in the definition of a feed. Websites such as Slashdot enable a community of users with similar interest to submit recently published articles for distribution to the larger group. A simple permission structure consisting of group read, group write and world read, world write could easily support this type of collaboration.

We are excited about the way this approach helps to transform the web for a user back from a popularity contest into a source of timely, relevant information that a user looks forward to every day.

7. ADDITIONAL AUTHORS

Additional authors: Alex Cozzi (IBM Almaden Research Center, email: `cozzi@almaden.ibm.com`) and Stephen Dill (IBM Almaden Research Center, email: `dill@us.ibm.com`)

8. REFERENCES

[1] R. Agrawal, R. Bayardo, and R. Srikant. Athena: Mining-based Interactive Management of Text Databases. In *Proc. of the Seventh Int'l Conference on Extending Database Technology (EDBT)*, Konstanz, Germany, March 2000.

[2] Apache Software Foundation. The apache http server project. `http://httpd.apache.org`, 2005.

[3] Ask Jeeves. Bloglines. `http://www.bloglines.com/`, 2005.

[4] Z. Bar-Yossef, A. Broder, R. Kumar, and A. Tomkins. Sic transit gloria telae: Towards an understanding of the web's decay. In *Proceedings of the Thirteenth International World Wide Web Conference*, New York, New York, 2004.

[5] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, 35(12):29–38, 1992.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW7: Proceedings of the Seventh International Conference on World Wide Web 7*, pages 107–117. Elsevier Science Publishers B. V., 1998.

[7] A. Z. Broder, S. C. Glassman, and M. S. Manasse. Syntactic clustering of the web. In *Proc. of 6th International World Wide Web Conference (WWW6)*, April 1997.

[8] I. BurrellesLuce. `http://www.burrellesluce.com/`.

[9] J. P. Callan. Document filtering with inference networks. In *SIGIR*, pages 262–269, 1996.

[10] J. P. Callan. Learning while filtering documents. In *SIGIR*, pages 224–231. ACM, 1998.

[11] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*, volume 30 of *Computer Networks and ISDN Systems*, pages 65–74, Brisbane, April 1997. Elsevier.

[12] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *Proceedings of the 8th International World Wide Web Conference (WWW8)*, pages 1623–1640, Toronto, Canada, 1999.

[13] K. Collins-Thompson, P. Ogilvie, Y. Zhang, and J. Callan. Information filtering, novelty detection, and named-page finding. In *TREC*, 2002.

[14] W. B. Croft, J. P. Callan, and J. Broglio. Trec-2 routing and ad-hoc retrieval evaluation using the inquery system. In *TREC*, pages 75–84, 1993.

[15] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *WWW*, pages 178–186, 2003.

[16] S. Dill, R. Kumar, K. McCurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-similarity in the web. In *IEEE International Conference on Very Large Databases (VLDB)*, Rome, Italy, September 2001.

[17] J. Edwards, K. S. McCurley, and J. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, pages 106–113, Hong Kong, China, 2001.

[18] E. Fredkin. Trie memory. *Commun. ACM*, 3(9):490–499, 1960.

[19] D. Gibson. Surfing the web by site. In *WWW Alt. '04: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pages 496–497, New York, NY, USA, 2004. ACM Press.

[20] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *Proceedings of the Fourteenth International World Wide Web Conference*, 2005.

[21] Google. Google alerts. `http://www.google.com/alerts`.

[22] Google. Google news. `http://news.google.com`.

[23] Google. Google reader. `http://reader.google.com`.

[24] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. How to build a webfountain: An architecture for very large-scale text analytics. *IBM Systems Journal*, 43(1):64–77, 2004.

[25] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *Proceedings of the Thirteenth International World Wide Web Conference*, New York, New York, 2004.

[26] IBM. Clever. `http://www.almaden.ibm.com/cs/k53/clever.html`.

[27] R. Jin, L. Si, C. Zhai, and J. Callan. Collaborative filtering with decoupled models for preferences and ratings. In *CIKM '03: Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 309–316, New York, NY, USA, 2003. ACM Press.

[28] Network Working Group. The atom syndication format. `http://ietfreport.isoc.org/all-ids/draft-ietf-atompub-format-11.txt`, August 2005.

[29] New York Times. Front page scan. `http://www.nytimes.com/pages/pageone/scan/`, 2005.

[30] G. Pant, S. Bradshaw, and F. Menczer. Search engine-crawler symbiosis: Adapting to community interests. In T. Koch and I. Sølvberg, editors, *ECDL*, volume 2769 of *Lecture Notes in Computer Science*, pages 221–232. Springer, 2003.

[31] RSS Advisory Board. RSS 2.0 Specification. `http://blogs.law.harvard.edu/tech/rss`, January 2005.

[32] Sage Development Team. Sage: a feed reader for Firefox. `http://sage.mozdev.org/`, 2005.

[33] San Franscisco Chronicle. SF Gate: News and Information for the San Francisco Bay Area. `http://sfgate.com`, 2005.

[34] C. Silverstein, M. R. Henzinger, H. Marais, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[35] Stand Alone, Inc. Quick news. `http://standalone.com/palmos/quick_news`, 2005.

[36] The Flock Development Team. Flock is a free, open source web browser. `http://www.flock.com/`, 2005.

[37] Thunderbird Development Team. Thunderbird - Reclaim Your Inbox. `http://www.mozilla.org/products/thunderbird/`, October 2005.

[38] Toomas Toots, Marcus Hettlage, Karsten Hoffrath. Feedreader is a lightweight open-source aggregator that supports rss and atom formats. `http://www.feedreader.com`, 2005.

[39] Topix.net, Inc. `http://topix.net`, 2005.

[40] Yahoo. The top news headlines on current events from Yahoo! News. `http://news.yahoo.com`, 2005.

[41] Ykoon B.V. Rssreader - free rss reader is able to display any rss and atom news feed (xml). `http://www.rssreader.com`, 2005.

[42] Y. Zhang and J. P. Callan. The bias problem and language models in adaptive filtering. In *TREC*, 2001.