Improving Web Search Relevance and Freshness with Content Previews

Siva Gurumurthy
Yahoo Inc.
Sunnyvale, CA, USA
shiiva@yahoo-inc.com

Hang Su
Yahoo Inc.
Sunnyvale, CA, USA
hangsu@yahoo-inc.com

Vasileios Kandylas
Yahoo Inc.
Sunnyvale, CA, USA
kandylas@yahoo-inc.com

Vidhyashankar Venkataraman
Yahoo Inc.
Sunnyvale, CA, USA
vidhyash@yahoo-inc.com

ABSTRACT

Traditional web search engines find it challenging to achieve good search quality for recency-sensitive queries, as they are prone to delays in discovering, indexing and ranking new web pages. In this paper we introduce PreGen, an adaptive preview generation system, which is run as part of a web search engine to improve search result quality for recency-sensitive queries. PreGen uses a machine learning algorithm to classify and select live web feeds, and generates"previews" of new web pages based on the link descriptions available in these feeds. The search engine can then index and present relevant page previews as part of its search results before the pages are fetched from the web, thereby reducing end-to-end delays. Our experiments show that PreGen improves the search relevance of a state-of-the-art search engine for recency-sensitive queries by 3% and reduces the average latencies of affected documents by 50%.

Categories and Subject Descriptors

H.1 [**Models and Principles**]: General; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Algorithms, Design, Measurement

1. INTRODUCTION

The recent success of social networking sites such as Twitter and Facebook has spawned a new interest in real-time search (RTS). Social networking sites provide a steady stream of user updates in real time. RTS primarily involves presenting these updates for time-sensitive user queries such as *"Haiti earthquake"* and *"BP oil spill"*. RTS suffers from relevance issues due to lack of sufficient content in the update streams. Alternatively, the freshness and relevance of web search engines can be enhanced to complement RTS by populating the search index with time-sensitive information from alternate sources such as web feeds.

Web feeds are a Web 2.0 application that provide information near real-time, but comparatively more descriptive than social networking updates. Really Simple Syndication (RSS) is a commonly used format of web feeds. Ever since its first version was created in 1999, RSS has been increasingly used in the World Wide Web to provide users with recent information and syndication about blog entries and news headlines. An RSS document consists of summaries or *abstracts* of recently published *items* along with their URLs and some meta-data, such as the publish time and the authors' names. With more web sites providing RSS feeds, the technology has truly become ubiquitous in the Internet today. However, it has to be noted that unlike real-time update streams, such as those from Twitter, RSS feeds need to be fetched regularly to receive fresh information, thereby making them less real-time than the former.

There has been a lot of work on RSS-based applications such as mining opinions from RSS corpus [13], and ranking articles from feeds for a news reader [17]. However, not much work has been done on how to effectively utilize RSS feeds to improve search engine quality. A search engine will potentially benefit from such an application in many ways.

First, many high-quality RSS feeds are updated frequently with fresh content from their web sites. In fact, they are updated almost simultaneously when a new web page is created. By effectively using these feeds, search engines discover fresh content in a timely manner and their coverage on recency-sensitive information is improved.

Second, there is significant room for improvement on the latency between discovering a new web page from an RSS feed and making it available to end-users in search results. Traditional search engines suffer from latency problems due to crawler scheduling algorithms and crawler's limited capacity. Many links which are considered lower priority by the crawler wait a significant amount of time for their turn to be fetched. The latency also stems from politeness constraints enforced by highly trafficked hosts. Besides, some web pages are forbidden by robots exclusion rules of their web sites and can never be fetched. With the help of abstracts and meta-data from RSS feeds, links from these feeds are indexed before they are actually fetched by creating synthetic bodies from the abstracts and meta-data, which we call "previews". This way the latency is reduced and the comprehensiveness of recency-sensitive information in the search engine index database is increased.

Finally, the presence of structured meta-data in the index also enables rich presentation of web search results. For example, authors and publish time of blog entries is shown to the end users in a clear and structured way for better user experience.

In this paper we present PreGen, a system that generates previews from RSS feeds. PreGen is run as a part of the search engine, which takes the content of a high-quality feed from crawler as input and creates previews for each and every link from the feed. These previews are then used to improve the search engine's coverage, freshness, relevance and presentation. Figure 1 illustrates how the generated previews are consumed and displayed in search engine for a recency sensitive query.

Using RSS feeds to improve the performance and quality of a search engine is not trivial. We have estimated the number of feeds present in the web to be in the order of billions and not all of them can be fetched and refreshed frequently due to crawler capacity constraints. Apart from magnitude, the quality of the feeds used for preview extraction is also a concern. For example, it has been shown that RSS feeds are often used for spamming and in some cases for search engine optimization [19], to artificially increase the rank of a page. Hence, we need a good selection algorithm that will only choose high-quality feeds. This algorithm should be able to predict the quality of an unknown RSS feed, filter out low-quality feeds, and promote feeds that are updated frequently and that contain relevant and descriptive content.

One simple way to select a feed with high quality is based on the goodness of its host or of its URL. However, this does not guarantee the feed to be a source of good previews. For example, page edit feeds from Wikipedia have a good host quality and are large in number, but they are practically useless for previews. Hence, ranking by host quality does not yield the desired preview source ordering. Page authority algorithms such as PageRank [3] also do not yield the correct ordering of preview sources. For instance, feeds of blogs and news pages are generally ranked low by these authority algorithms as they are not referred (linked) as much as their corresponding pages. Hence, we need a better algorithm to determine the quality of feeds. Under the guarantee of this algorithm, previews extracted from high-quality feeds will be trustworthy and a search engine can consume and display these previews in search results as a substitute for pages waiting to be crawled.

PreGen uses a machine learning (ML) based feed selection method to select feeds based on their *quality*. The notion of feed *quality* is defined later in detail, but in general the definition is based on feed use case, i.e. to extract previews. We built a Gradient Boosted Decision Tree (GBDT) [7] model for predicting the quality of the feed. This model is then used to select high-quality feeds. To the best of our knowledge, we do not know any existing work for feed selection method. Hence, we compared our ML-based feed selection with PageRank-based selection and editor picked feeds. The ML-based selection method performed on par with editorial selection, and outperformed the PageRank-based selection by 10% with respect to our evaluation metric, the click probability of the generated articles. The ML-based selection also generated more articles in the same period than editor picked feeds or PageRank-based feeds.

PreGen adds several advantages over a state-of-the-art search engine :

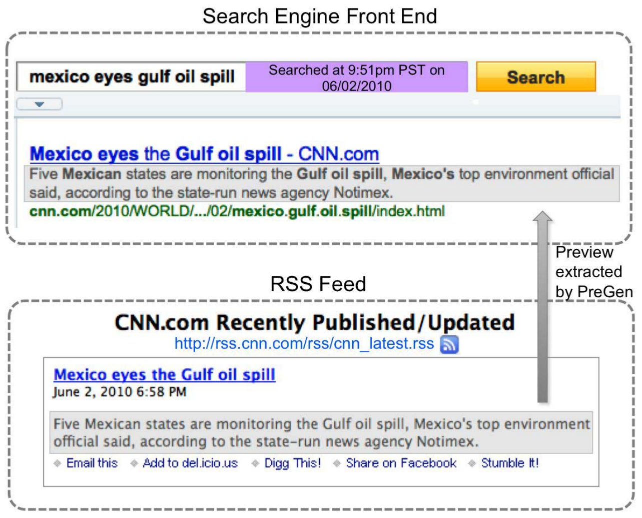

Search Engine Front End
mexico eyes gulf oil spill
Searched at 9:51pm PST on 06/02/2010
Search
Mexico eyes the Gulf oil spill - CNN.com
Five Mexican states are monitoring the Gulf oil spill, Mexico's top environment official said, according to the state-run news agency Notimex.
cnn.com/2010/WORLD/.../02/mexico.gulf.oil.spill/index.html
Preview extracted by PreGen
RSS Feed
CNN.com Recently Published/Updated
http://rss.cnn.com/rss/cnn_latest.rss
Mexico eyes the Gulf oil spill
June 2, 2010 6:58 PM
Five Mexican states are monitoring the Gulf oil spill, Mexico's top environment official said, according to the state-run news agency Notimex.
Email this
Add to del.icio.us
Digg This!
Share on Facebook
Stumble It!

Figure 1: An example of preview extracted by PreGen displayed in search result.

1. It can serve recency-sensitive queries, those queries for which user expects time-sensitive information [4].
2. It reduces the latency of affected documents.
3. It allows the search engine to present even robots-blocked pages in the search results.

Our experiments show that PreGen recalls at least one or more results for 90% of all the recency sensitive queries, which improves the search relevance metric Discounted Cumulative Gain (DCG [9]) by 3% over a state-of-the-art search engine. Furthermore, PreGen reduces the average latencies of affected documents by 50%. We also find that 10% of the documents from PreGen are previews of robots blocked pages, which would otherwise never be accessible to a search engine.

In summary, this paper makes the following contributions:

- We introduce PreGen, a novel preview generation system that improves the end-to-end latency and comprehensiveness of recency-sensitive information.
- We define the notion of feed quality with respect to preview generation.
- We present a new ML-based feed selection method to select feeds and compare it with other methods.
- We present experimental results demonstrating the advantages of content previews in terms of relevance gain and latency reduction over a state-of-the-art search engine.

This paper is organized as follows. The next section discusses related work. Section 3 describes the machine learning based feed selection method. Section 4 describes the PreGen system and Section 5 contains the experimental setup and results. Section 6 concludes the paper with directions to future work.

2. RELATED WORK

There has been extensive research on improving a search engine's core results properties, namely relevance, coverage and freshness. Examples include the use of social annotations and bookmarks [2, 8], click-through data [10], browsing information [20] and search history [18]. There has also been work to enrich the search results using non-regular web pages. Blogs have been commonly used in web search to increase the relevance. For example, Mishne [12] uses blog properties such as timestamps, number of comments and query-specific terms to improve ranking.

There has been some research involving the use of feed selection algorithms for various purposes. NectaRSS [17] and Cobra [15] select web feeds based on the user's past choices and preferences in the context of news readers. Elsas et al. [5] present probabilistic retrieval and feedback models for search and recommendation of blog feeds. RSSMicro uses a feed ranking algorithm for its feed search engine. It provides a free FeedRank calculation tool [16] that displays the rank of a web feed based on its quality measure and can successfully distinguish spam feeds from high-quality ones. However, there has been very little published work on using RSS feeds directly to improve the traditional web search results quality.

Our work comes close to that of Macdonald and Ounis [11], which ranks blogs based on individual rankings of their posts (similar to the ranking of feeds in this work) and experiments with indexing the XML feeds instead of the actual blog posts. However, the reason the authors do this is to reduce crawler and indexer load and not to improve recency or relevance of the search results.

3. ML FEED SELECTION

In this section, we define feed quality and model the feed selection problem as a machine learning problem. The developed ML models are used to classify and select feeds. These feeds are used to extract previews. The quality of previews depends on the quality of feeds, hence we need a good ML algorithm to predict the quality of feeds, which can be used for feed selection.

Feeds can be selected by using a heuristic method such as PageRank to order the feeds and select the top ones. Alternatively, feeds can be selected by an ML-based approach. Richardson et al. [14] have shown that a query-independent ML ranking algorithm can order web pages better than PageRank. Their work orders pages based on their usefulness in answering search queries. As feeds are not directly used for answering search queries, we do not use the same objective function. However, we follow a similar ML-based approach with a different objective function. We could model the feed selection problem as a ranking problem, but generating a preference order of feeds based on quality is difficult, as quality is query-independent and is subjective to the application. Hence, we chose to model the feed selection problem as a classification problem, where an ML algorithm is trained to predict the quality class of a feed. The feeds are then selected from the highest quality class. If the number of feeds from that class are more than what is required, then a subset of the feeds with the highest class probabilities are selected.

The quality of a feed is characterized by the application that uses it. For example, if the application is an RSS reader, then a high-quality feed can be characterized by how comprehensively it covers trustworthy and personally relevant information to the user. Past work [17] has addressed the feed ranking problem for RSS readers using this definition of quality. However, in our case, feeds are used as a preview source for search engine and the feed quality needs to be redefined to suit this application. We propose that the properties listed in Table 1 are desirable in a high-quality feed. These properties are generic in nature and can be used for other applications as well.

The listed properties determine the labels for modeling our ML algorithm. In order to measure whether a feed exhibits one or more of these properties, we define two quantifiers, *linked page quality* *(α)* and *item quality (β)* which we use to formally define the quality

1. The information from the feed is relevant to the search engine.
2. The feed items have rich descriptions about the links.
3. The feed produces high quality links about recent events.

Table 1: Desired properties of a good quality feed.

Y (Class)	α	β
A	1	1
B	1	0
C	0	1
D	0	0

Table 2: Feed quality classes.

properties of a feed. Let $G(q,u)$ be the relevance grade in scale 0-4 (*0-Bad, 1-Fair, 2-Good, 3-Excellent, 4-Perfect*) assigned by human editors for a query-URL pair (q,u), where u is the result returned by the search engine for query q. Let us define a feed r_i as a collection of item-link pairs as shown in figure 2. It can be expressed as $r_i = \bigcup_j \langle I_i^j, L_i^j \rangle$, where I_i^j is the j^{th} item and L_i^j is the j^{th} linked article page. The linked page quality α_i can be defined as

$$\alpha_i = \begin{cases} 1 & \exists j, \exists q \in Q \text{ s.t. } G(q, L_i^j) \geq 3 \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

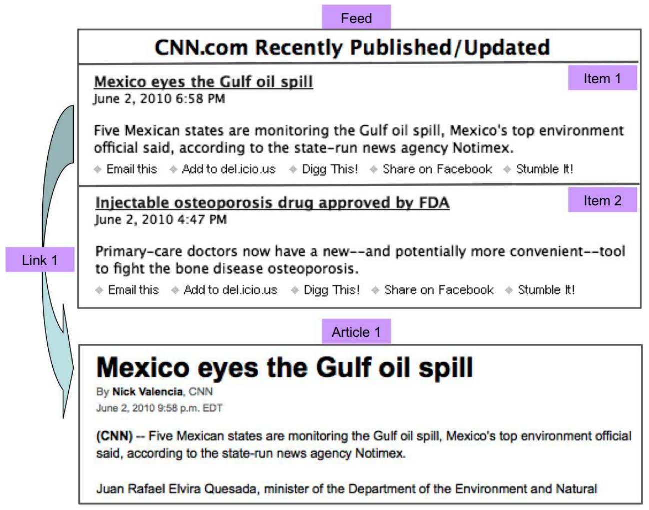
The item quality β_i can be defined as

$$\beta_i = \begin{cases} 1 & \exists j, \exists q \in Q \text{ s.t. } G(q, I_i^j) \geq 3 \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The query set Q was editorially selected to be "recency-sensitive" [4] from query logs of the search engine. These are queries for which the user expects relevant and fresh documents. $G(q, I_i^j)$ and $G(q, L_i^j)$ are assigned by editors when the item of the feed or the linked article is returned as a search result for a recency-sensitive query q.

We use α and β to determine the quality class of the feeds. The property α of a feed signifies whether it links to a web page that contains information relevant to at least one query from Q. The property β of a feed indicates if it contains an item (link description) which has sufficient information relevant to at least one query from Q. Therefore, the α and β quantifiers directly satisfy the first two properties listed in Table 1. In addition, as the query set Q is recency-sensitive, choosing feeds based on α and β will select those feeds that produce recency-sensitive articles, thus satisfying the third property. If Q is large, then the selected feeds are likely to cover a broad spectrum of topics as well. Hence, the quality classes described by the quantifiers α and β cover all the desirable feed properties, assuming a carefully selected query set Q. We have observed that feeds selected using this formulation satisfied all the listed properties and these observations are discussed elaborately in our experiments section 5.

The pair (α,β) can take four possible values as shown in Table 2, hence the feeds belong to one of four quality classes. The objective of the selection function is to predict a class for each feed and choose feeds that belong to class A, the highest quality class. This objective can be expressed as a general classification problem. Let $X = \{x_i\}$ be a collection of feature vectors of feeds and $Y = \{y_i\}$ be the associated class labels described by the feature vector. The



CNN.com Recently Published/Updated

Mexico eyes the Gulf oil spill
June 2, 2010 6:58 PM

Five Mexican states are monitoring the Gulf oil spill, Mexico's top environment official said, according to the state-run news agency Notimex.

Email this · Add to del.icio.us · Digg This! · Share on Facebook · Stumble It!

Injectable osteoporosis drug approved by FDA
June 2, 2010 4:47 PM

Primary-care doctors now have a new--and potentially more convenient--tool to fight the bone disease osteoporosis.

Email this · Add to del.icio.us · Digg This! · Share on Facebook · Stumble It!

Mexico eyes the Gulf oil spill
By Nick Valencia, CNN
June 2, 2010 9:58 p.m. EDT

(CNN) -- Five Mexican states are monitoring the Gulf oil spill, Mexico's top environment official said, according to the state-run news agency Notimex.

Juan Rafael Elvira Quesada, minister of the Department of the Environment and Natural

Figure 2: RSS definition.

classification problem is to find an underlying function $R(X)$ such that $Y = R(X)$. In general, the function $R(X)$ is learnt by minimizing a loss function $L(Y, R(X))$ such as squared loss or hinge loss. There are several methods in literature, such as Logistic Regression, Naive Bayes, SVM, Random Forest and decision trees, to find the fitting function.

An important point to note from Table 2 is that we cannot compare classes. For example, we cannot claim class B is better than class C or vice versa and this explains the reason why we cannot model the problem of identifying feed quality as a regression problem or a ranking problem.

We use Gradient Boosted Decision Trees (GBDTs) [7] to train the model. A boosted tree is an aggregate of regression trees. Each regression tree partitions the space of explanatory variable values into disjoint regions D_j, $j = 1, 2, \ldots J$ associated with the terminal nodes of the tree. Each region is assigned with a value ϕ_j which is the output of the regression tree for any input x in that region. The complete tree is then

$$T(x;\Theta) = \sum_{j=1}^{J} \phi_j I(x \in D_j) \tag{3}$$

where Θ is the set of all regions D_j and their values ϕ_j and I is the indicator function.

A boosted tree is an aggregate of M regression tees, which is an additive model of the form:

$$f_M(x) = \sum_{m=1}^{M} T(x;\Theta_m) \tag{4}$$

At each stage m, a new tree is added to the model iteratively. The parameters Θ_m of this new tree are estimated by fitting the residuals of the loss from stage $m-1$.

Most of the classification algorithms, including GBDT, usually solve only binary classification, but we have four classes as shown in Table 2. Hence we use a "one-versus-all" scheme to predict the class probabilities. For each feed, the highest probability decides the class of the feed. Depending on the application within PreGen (offline and online feed selection as described later in section 4), the selection function can choose the top feeds ordered by class probability or choose feeds based on a class probability threshold. We used four quality classes for classification, instead of two (i.e. class A versus the rest), because the other class feeds are useful for other search-related applications. For example, class B feeds can be used for discovery in the crawler of a search engine, and class C feeds can be used for rich presentation of search results. In section 5.1.4, we explain the classification modeling in greater detail.

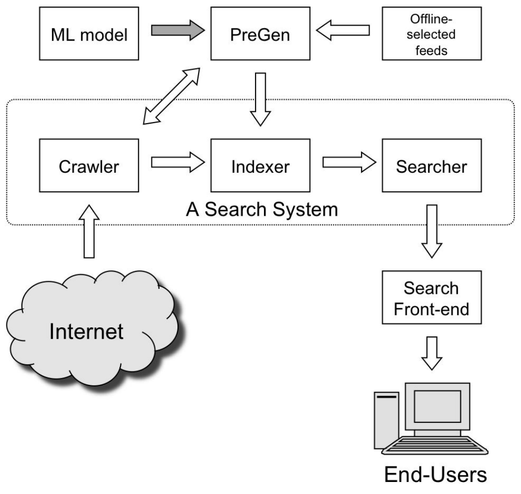


Figure 3: PreGen-enhanced search engine.

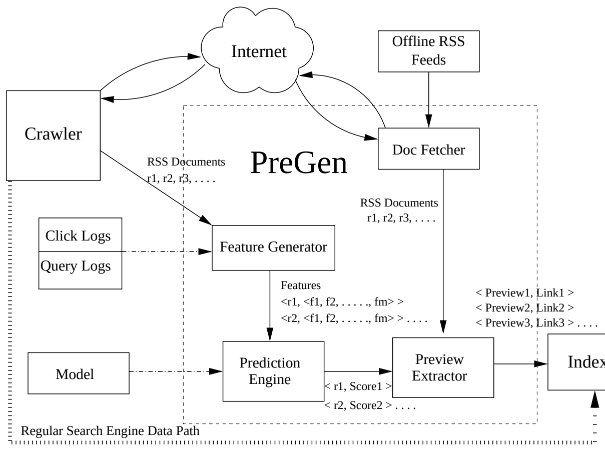

Internet
Offline RSS Feeds
Crawler
PreGen
Doc Fetcher
RSS Documents r1, r2, r3, ...
RSS Documents r1, r2, r3, ...
Click Logs
Query Logs
Feature Generator
Features <r1, <f1, f2, ..., fm> > <r2, <f1, f2, ..., fm> > . . . .
< Preview1, Link1 > < Preview2, Link2 > < Preview3, Link3 > . . . .
Model
Prediction Engine
Preview Extractor
Indexer
< r1, Score1 > < r2, Score2 > . . . .
Regular Search Engine Data Path

Figure 4: PreGen implementation architecture.

4. PREGEN: DESIGN AND IMPLEMENTATION

PreGen is a preview generation framework. Its central functionality is to use the trained model that we described in the previous section to classify and select "qualifying" feeds, extract their items and generate previews from the items to enhance search result quality.

4.1 Design

PreGen is tightly coupled with the operations of a search engine. A typical search engine consists of several key components that include the crawler, the indexer and the searcher [3]. Figure 3 shows these components and their relationship with PreGen. The crawler fetches web pages from the Internet. The indexer consumes their content and creates the search index database. The searcher uses this database for answering user queries. There is a search front-end that forwards search queries from the user to the searcher, converts the search results into the required format and shows them to the end-users.

As one of its inputs, PreGen receives a stream of RSS feeds that were just crawled from the crawler. It uses the ML model described in Section 3 to classify these feeds. Then, it generates previews of the items present in the qualifying feeds and sends these previews to the indexer. This way PreGen receives feeds from the crawler and applies the ML model to determine the quality of the input feeds dynamically. The quality of a feed as a score (class probability) is used for the *online feed selection* process. The online selection has access to only one feed at a time. Hence, the decision to select the feed is independent of other selections. Therefore, we fix a threshold based on capacity and select any feed that has a quality score greater than the threshold at any instant. If a selected feed falls below the quality threshold, it is no longer used for preview generation.

The feature sets used in the online ML models are limited in number and time-sensitive in nature, based on past crawl events of each page, such as page content features and linked page features, which change with every crawl event. Hence, we also complement the online feeds with *offline-selected feeds*. The offline feeds are generated by sending a large pool of known RSS feeds through a specialized ML model trained using offline features. The ML algorithm in this case can use sophisticated features such as PageRank, host quality and features from query logs, which may not be available online. This offline selection effectively creates a set of feeds that are pre-selected as appropriate for time-sensitive queries. In Section 5 we explain the effect of using different feature sets in terms of misclassification rate. By applying a capacity limit, a set of top-ranked feeds from the pool can be chosen as offline feeds. PreGen takes this set of feeds as input, preemptively fetches them and creates previews from them.

The previews generated from PreGen are sent to the indexer and stored in the database along with the other regular documents. When a relevant preview is returned from the searcher to the search front-end, the latter can use the preview and its associated metadata to enrich the search result presentation as shown in Figure 1.

4.2 Implementation

Figure 4 shows the internal workings of the PreGen implementation that we used in our experiments discussed later in Section 5. As the crawler fetches documents from the web, it sends only RSS documents through an additional pipeline of PreGen modules. The crawled feeds are streamed into a *Feature Generator* that creates or extracts a set of features from each feed. While some features such as *word count* and *number of items* can be obtained directly from the feed, others such as *view count* and *click count* have to be derived from the search engine's query and click logs.

The generated feature sets are then sent to the *Prediction Engine*, which executes our ML-based model, and calculates a quality score (class A probability) for each feed. The *Prediction Engine* is a C++ implementation of GBDT model applier. It reads a model file which contains a series of regression decision trees that uses feed features as its input variable. The models are built during the training process using Treenet software. The modeling procedure is explained in Section 5.

The feeds and their scores computed by the *Prediction Engine* are then streamed into the *Preview Extractor* which produces previews of items only for qualifying feeds. The *Preview Extractor* may also receive feeds from a *Doc Fetcher* that regularly fetches RSS documents that were preselected offline. The role of the *Preview Extractor* is to parse an RSS feed and extract the links and meta-data, such as link description, date, author name and create a synthetic document which we call a preview. Hence for each link present in the RSS feed, the text present in the preview is used as a substitute for the content of the page while indexing against the link.

5. EXPERIMENTS

We implemented PreGen and integrated it as part of an end-to-end baseline search engine, which shows relevant previews in the search result for appropriate queries. In this section, we present the evaluation of the ML feed selection used in PreGen. We also present the search quality performance of the PreGen-enhanced baseline system and compare it with the baseline system in different experiments.

5.1 Feed Selection Evaluation

The objective of a feed selection algorithm is to select a desired number of feeds, based on their likelihood to improve the search quality. We perceive search quality by two measures, namely relevance and user engagement in terms of number of clicks.

5.1.1 Measures

The evaluation of the feed selection was split in two phases. First, we evaluated the performance of the ML algorithm in classifying feeds into quality classes. Second, we evaluated the selection function that utilized these classes to select the desired number of feeds.

In order to evaluate the multi-class learning algorithm, we used misclassification rate. If y_i' is the predicted class of feed r_i and y_i is the actual class, the misclassification rate (γ) can be expressed as:

$$\gamma = \frac{1}{N}\sum_{i=1}^{N} I(y_i' \neq y_i) \qquad (5)$$

where I is the indicator function, and N is the total number of instances.

To evaluate the selection function, we used the clicks generated on articles from selected feeds. Let S be the selection function that takes a set of feeds R as input and produces the top n feeds. It can be expressed as $S(R) = (r_1, r_2, \dots r_n)$. Let us assume that each feed r_i generates k articles in one month, $(a_1, a_2, \dots, a_k)$. For each generated article a_j, we aggregate all the clicks $c(a_j)$ and impressions $v(a_j)$ for all the search queries in a month, independent of its rank in the search results. We define the *simple click probability* of an article a_j as $\frac{c(a_j)}{v(a_j)}$. We compute the *aggregated click probability* $M(r_i)$ for each feed r_i by averaging the click probabilities of all the articles generated from the feeds:

$$M(r_i) = \frac{1}{k}\sum_{j}\frac{c(a_{ij})}{v(a_{ij})} \qquad (6)$$

This aggregate click probability $M(r_i)$ is used to evaluate the selection functions.

5.1.2 Dataset

There are several types of feeds in the web, such as news feeds (about news articles), blog feeds (about blog articles), live feeds (updates), and product feeds. The training examples should be representative of these types. Hence, we sampled feeds from the following sources for modeling:

1. A news consortium consisting of more than 800 newspapers.

2. A personalization site (my.yahoo).

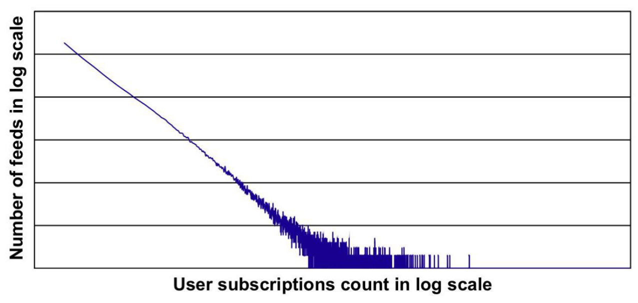


Figure 5: Distribution of feed user subscriptions.

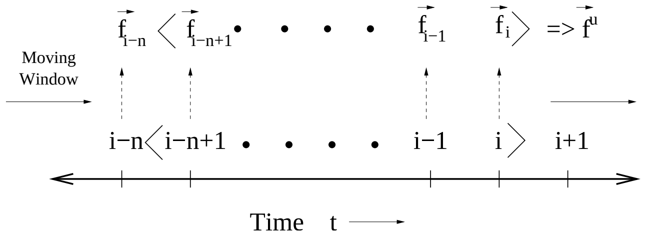
3. A bookmarking site (delicious).
4. A pool of feeds selected by human editors.
5. A random sample set of feeds from web.

As some of the sources such as book marking sites tend to have high number of low quality feeds, a random sampling for training would skew the distribution of training set towards low quality feeds. Hence, for these sources that have statistics which can be used as surrogates for quality, we used those statistics and generated stratified feed samples for modeling. For example, the personalization site has a statistic called *user subscription count*, the number of users subscribing to a feed. This measure gives a simple way to order feeds from this source approximating their quality. As shown in Figure 5, the user subscriptions are distributed by a power law. In order to select feeds for modeling, we performed stratified sampling based on the user subscription distribution. A similar procedure was adopted for the bookmarking site using the number of bookmarked tags as the quality measure. For the feeds from the web, we performed stratified sampling based on the number of feeds in a host. The feeds from sources which did not have such a statistic were included directly in the final sample set for modeling. This included the feeds recommended by the news consortium and editorially picked feeds. The final feed set contained a total of 12,000 feeds.

In order to get the label for each feed, we obtained the quality class of a feed with the help of the quality indicators, namely linked article quality (α), and link description quality (β) as discussed in Section 3. These indicators were obtained by the following experiment. The generated 12000 feed samples were fed into the PreGen-enabled search engine which was crawled every 15 minutes and the previews created from these feeds were indexed continuously. Every 24 hours for 1 week, human editors picked queries from a recency-sensitive query dictionary and issued them to the PreGen-enabled search engine. The relevant previews and articles returned by the search engine were graded by editors, and these relevance grades were used to quantify the two feed quality indicators α and β as described in equations 1 and 2. We determined the labels of all the feeds by mapping their respective quality indicators to the quality class as shown in table 2. The queries used in this experiment were editorially short-listed from a dictionary of recency-sensitive queries, which was updated every hour by running a recency-sensitive query classifier [4] on the query logs of the search engine.

5.1.3 Feature extraction

We compute our features using the content of the feed, the content of the linked page, the link structure, host characteristics, and refresh and update characteristics. Since the feeds change frequently,



$\vec{f}_{i-n}$ $\langle$ $\vec{f}_{i-n+1}$ • • • • $\vec{f}_{i-1}$ $\vec{f}_{i}$ $\rangle$ => $\vec{f}^u$
Moving Window
i−n $\langle$ i−n+1 • • • • i−1 i $\rangle$ i+1
Time t

Figure 6: Time sensitive feature vector computation

their quality also changes and the feature vector describing the quality of a feed needs to account for the variation. Every crawl event i produces a snapshot of the feed feature vector $\vec{f_i}$, such as linked page content features and feed content features. These snapshots track the feed quality characteristics over time. At the time of crawl event i, we take the past n snapshots of this feature vector $\langle \vec{f}_{i-n+1}, \dots \vec{f}_{i-1}, \vec{f_i} \rangle$ and apply aggregating functions (such as average, min, or max) to get a set of *time-sensitive* feature vectors $\vec{f^u}$ (Figure 6). An example of these features is average out-link counts over the past n snapshots. We use the aggregating function to smooth the variation that is inherent with the content-level features of a frequently changing feed. There is also another type of features $\vec{f^v}$ which is insensitive to time such as host, and URL characteristics. The final feature set for a particular feed contains these two types of features: $F = \vec{f^u} \cup \vec{f^v}$.

Some of the features we used are based on query-independent ranking [14] and clickrank [10], which explain features used for general web page ranking. The features we used were based on:

- *Feed content*: the number of items in the feed, latest update date, average number of words in the title and abstract, total words, number of tags per item, and whether the feed linked to image and video.
- *Linked page content*: cosine similarity in feed item content and the linked page content, similarity in titles, word frequencies, and word popularity.
- *Link characteristics*: out-link counts, in-link counts, average in-link count of the linked pages, PageRank of the feed, average PageRank of the linked pages, and anchor text lengths.
- *Host and URL characteristics*: average PageRank at the host and domain level, and average link counts at the host level.

5.1.4 Model evaluation

We generated 12,000 samples of feeds, which contained 4 types of labels, and features extracted as described in the previous section. We split the samples into 90% for training and validation, and 10% for testing. We trained different models using different ML algorithms such as gradient boosted decision trees, support vector machines (SVM), random forests, Adaboost, J48 decision trees, and logistic regression. We used 10-fold cross-validation on the training set to tune parameters and measure the classification errors.

We used the Treenet software [1] to train GBDT models with the following parameters: 300 trees, 15 nodes and 0.15 as shrinkage factor. We used the Weka library [6] to test the rest of the algorithms. Some algorithms allow multi-class classification by design; for those that do not, we used a 1-vs-all scheme and classified the instances to the highest scoring class. In Table 3, we report the misclassification rate and its standard deviation for the 10-fold cross-validation. We also report the performance of the models on

Model	10-fold validation	Test set
GBDT	20.4% ± 0.7%	19.4%
SVM	20.5% ± 0.7%	19.5%
Random forest	20.8% ± 1.2%	20.8%
Adaboost M1 with trees	24.0% ± 0.7%	22.8%
Logistic regression	24.1% ± 0.6%	23.0%
J48 decision tree	26.1% ± 0.9%	25.9%

Table 3: Misclassification rate of ML methods.

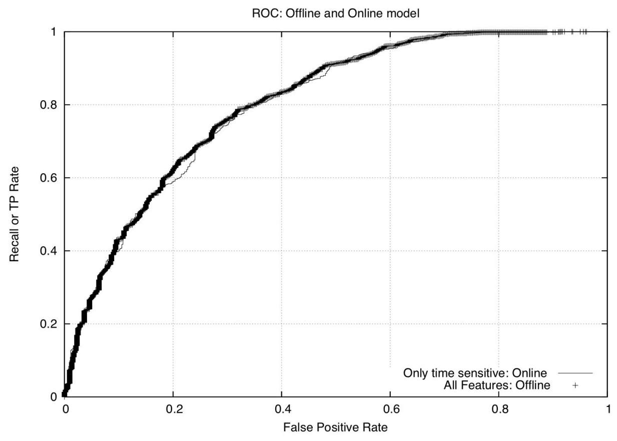


Figure 7: Class A ROC curves for online and offline model.

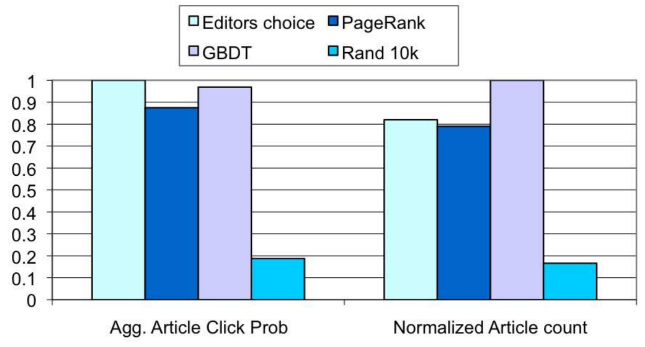


Figure 8: Comparing selection functions using the aggregate article click probability $M(r_i)$ and the count of generated articles. GBDT outperforms PageRank and comes close to human-selected feeds with respect to click probability. GBDT also generates the highest number of articles.

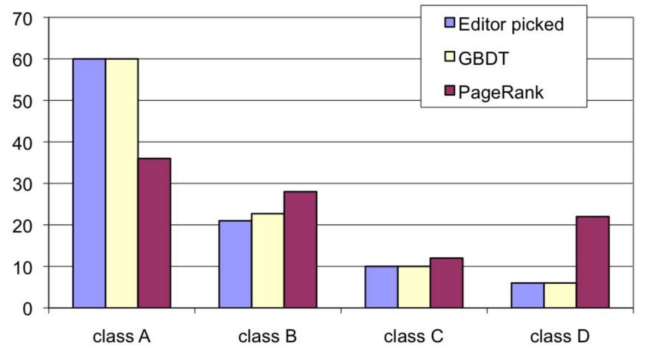


Figure 9: Comparing selection functions by quality class ownership. The quality classes were defined in section 3. GBDT selects feeds whose class distribution resembles that of the editorially-picked dataset. PageRank misses a lot of feeds from the highest quality class A.

the test set. The GBDT model had the lowest misclassification rate of 19.4% on the test set. The SVM model trained with the radial basis function $exp(-\gamma * |u-v|^2)$ as kernel ($\gamma = 50$, cost $c = 1$) had comparable performance with GBDT. Even though the difference is not statistically significant, we chose GBDT for its better average performance and its faster classification due to its feature reduction capabilities.

As discussed in the Section 4, ML models are used for both online and offline feed selection. These selection functions use different feature sets, but their goal is similar, to select high quality feeds for recency-sensitive queries. The offline models can comprehensively include all the host-level, page-level plus the time-sensitive features, whereas the online models can only use the time-sensitive features. The model performance in both cases did not differ much in overall misclassification rate. The time-sensitive online model had around 19.7% misclassification rate, whereas the offline model had 19.6%. The ROC curves as shown in Figure 7 for the class A classification for online and offline models are very similar and they both have similar area under the curve. This shows that time-sensitive features alone are strong enough indicators to match the performance of the offline selection feature set.

5.1.5 Evaluating selection function with heuristic methods

Our selection function uses the GBDT classifier to classify feeds and selects the top K feeds of class A feeds, where K is the capacity of the PreGen system. The order of feeds within a class is decided by the class ownership probabilities output by the classifiers. In order to evaluate the selection function, we used a test feed pool of $100,000$ feeds and fixed K to be $10,000$. We used four sets of selected feeds for our experiment:

1. Top $10,000$ feeds ordered by PageRank from the test set
2. $10,000$ random feeds from the test set
3. $10,000$ news consortium feeds picked by human editors, and
4. $10,000$ feeds selected from the test set using GBDT.

Figure 8 shows the comparison with respect to the aggregated click probability (Equation 6) of articles normalized to the maximum scale. For example, the article click probability for editorially picked feeds was scaled to 1.0 since they had the highest $M(r_i)$ score as described in Equation 6. GBDT-based feeds also generate very high click probability, close to editor picked feeds. They also have the highest article generation rate (as shown in Figure 8); this can be attributed to the fact that our model is adaptive and picks up feeds that cover many articles.

In order to examine the quality of 10,000 short-list feeds with respect to our notion of quality introduced in Section 3, we sampled 1,000 feeds from each set containing 10,000 feeds (excluding the random set) and assigned quality classes through manual labeling.

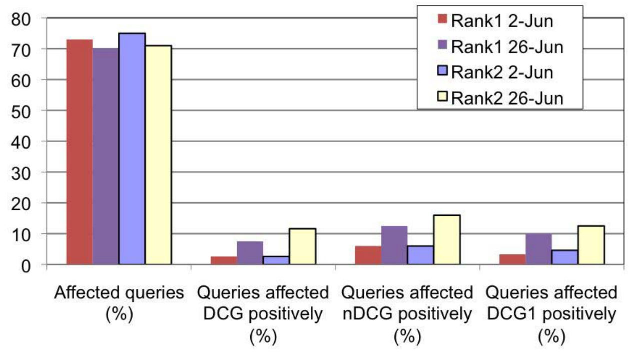


Figure 10: Percentage of queries for which relevance is improved using PreGen versus the baseline web index. Experiments were conducted at two different times and with two ranking functions to verify that the improvement was due to the preview index.

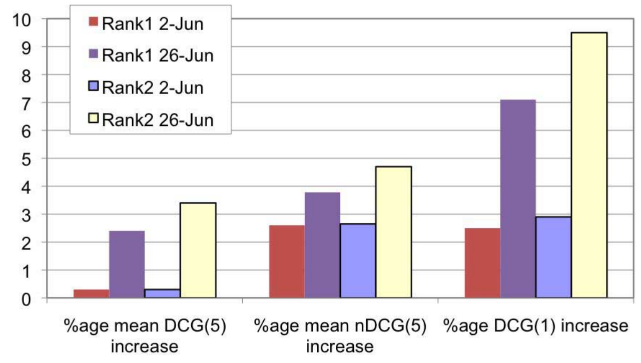


Figure 11: Percentage of DCG improvement. PreGen provides a minor to moderate improvement to DCG metrics for recency-sensitive queries.

Figure 9 shows the distribution of the feeds by class ownership. GBDT-based feeds have high quality (class A) feeds in the same proportion as editorially picked feeds. PageRank feeds do not perform as well as others. This is due to the fact that the GBDT model was trained to predict these classes. The fact that the majority of the GBDT selected feeds fall in class A is desirable as this is the class of feeds with the highest item and linked page quality. These metrics show the advantage of our GBDT-based selection function. In the next section, we study the effect on search result quality due to the previews created from the feeds selected via this function.

5.2 Previews in Search Relevance and Freshness Improvement

As illustrated in section 4, PreGen generates previews for answering relevant recency sensitive queries. In this section, we will study the effect of PreGen on a state-of-the-art search engine. We assigned a maximum capacity of $K = 100,000$ feeds for PreGen. PreGen updates the feed set periodically based on the ML selection function. As the feeds selected online change frequently, we had to use only offline feeds in our experiment for the sake of consistency. The offline feeds were updated once every month to study the effect of feed set change on evaluation metrics before and after an update.

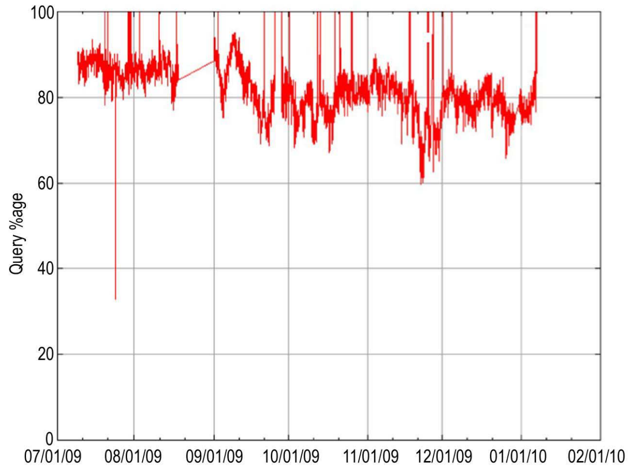


Figure 12: Percentage of queries that are covered only by previews over a period time.

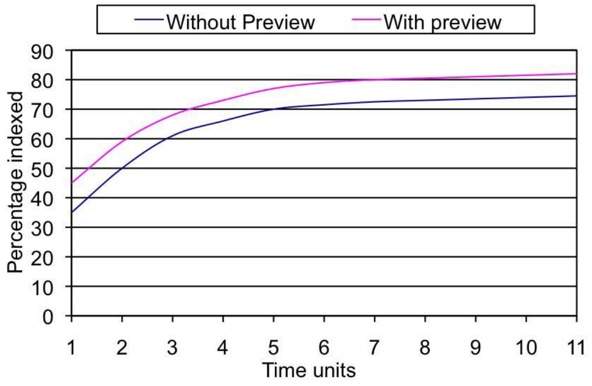


Figure 13: Latency Improvement. Whereas the baseline search engine takes 5 time units to index 70% of the affected documents, the use of previews reduces that time to 3.3 time units.

The effect of adjusting the update period or using online feeds will be studied as part of future work.

At any time instant, there were 250,000 previews generated and stored in the search index. This included previews of robots-blocked pages, network-constrained pages and regular web pages. Most of the pages from high-bandwidth popular hosts are instantly fetched by the crawler and hence the need for previews is limited. The real advantage of previews surfaces when the previews are of robots-blocked article pages, from network-constrained popular hosts or from non-popular hosts. These previews reside in the search index to answer recency-sensitive queries. We have observed relevance improvements due to the presence of the previews.

5.2.1 Relevance Improvement

We sampled around 1,200 recency-sensitive queries and conducted the relevance experiment at two different time points, before and after one feed set update to understand the effect of time. Also, in order to study the effect of PreGen independently from the underlying relevance ranking function, we used two experimental ranking functions ($Rank_1$ and $Rank_2$ [4]) against the same set of queries. The experiment included querying the baseline web search system that indexes all web pages and the PreGen system that indexes only previews. We used the Discounted Cumulative Gain (*DCG*) and the normalized Discounted Cumulative Gain (*nDCG*) metrics to measure the advantage of PreGen. Given a query q and a ranked list of L documents, the *DCG* for query q can be expressed as:

$$DCG(L) = \sum_{i=1}^{L} \frac{2^{g(i)} - 1}{\log(1+i)}$$

where $g(i)$ is the relevance grade (assigned by human editors) in scale 0 to 4 of the i^{th} document in the ranked result set. The *nDCG* of L documents is defined as $\frac{DCG(L)}{iDCG(L)}$, where $iDCG(L)$ is the ideal *DCG* computed by applying the *DCG* function on the optimal relevance order of L documents. In this case the optimal relevance order is obtained by sorting the relevance grade $g(i)$ of a document. For each query, we compared the search results from the preview index and the baseline web index using the metrics $DCG(5)$, $DCG(1)$ and $nDCG(5)$. Figure 10 shows the percentage of recency sensitive queries for which the relevance metrics were higher for results from the preview index than those from the baseline web index. It can be observed that variations caused by using different ranking functions are small. However, the time of study affects the relevance due to the use of previews. 8-11% of all recency-sensitive queries affected $DCG(5)$ positively due to previews on June 26th 2009, and 2-3% of queries on June 2nd 2009. The variation in time needs to be studied carefully in our future work as it could be due to varying feed set, changing preview documents, or changing nature of recency sensitive queries. For the case of queries where the *DCG* is lower than the baseline, the relevance will not be prominently affected when PreGen is integrated with the baseline. This is because, for those queries, although PreGen generates irrelevant previews, a good ranking function will not consider these previews for inclusion in the final search results. Hence, subject to the choice of a good ranking function, PreGen can only have a neutral or positive effect on the baseline search engine.

Now, we study the effect of integrating PreGen with the baseline search system. The overall relevance of the PreGen enhanced baseline improved as shown in Figure 11. For instance, the $DCG(5)$ improvement is between 0.2% and 3.2% over the baseline $DCG(5)$ for $Rank_2$. Another important result to observe is the presence of previews in the first position. $DCG(1)$ increases by 3% in the worst case and 9.5% in the best case. These results indicate the usefulness of previews in improving the search relevance. Previews also cover a broad spectrum of recency-sensitive queries. Figure 12 shows that 90% of the queries are covered by the previews over different periods in time.

All these results support the fact that feeds selected using the ML algorithm satisfy our notion of quality as described by the three properties listed in Table 1. For example, Figure 8 indicates that the selected feeds generated high quality articles which had as high click probability as articles from editorial picked feeds. Furthermore, Figures 10 and 11 show that the previews generated from feeds contained enough description and were relevant to the search engine. Thus, these results validate the first two properties. As shown in Figure 12, the selected feeds generated articles that covered a broad spectrum (90%) of recency-sensitive queries, therefore the articles must be about the recent events, which validates the third property.

5.2.2 Latency Improvement

The preview documents reduce the end-to-end latency of the search engine. Previews temporarily stay in the index as synthetic documents and answer relevant queries before actual contents of the web pages are fetched and indexed. Figure 13 shows the distribution of end-to-end delay of the documents. Only the affected documents in PreGen are considered, as PreGen does not affect the delays of regular documents. The graph shows 70% of the affected documents in the baseline have delays of 5 time-units or less, while documents in the PreGen system has 3.3 units or less. This means that for 70% of the documents, PreGen saves 1.7 time-units in end-to-end delay. For all the affected documents which were indexed within 5 time units by the baseline since their discovery, the average reduction in the latency was 50%. If we consider any document that was indexed later than 5 time units, the improvement is even greater. The graph also shows that there are 6-10% additional documents in the PreGen-enhanced baseline system, which are the previews of robots restricted pages. The baseline system will never be able to present the links to these restricted pages for relevant queries without PreGen.

6. FUTURE WORK AND CONCLUSION

In this paper, we presented PreGen, a system which runs as an enhancer to traditional web search engines to generate previews. The previews are generated from a selected sample of RSS feeds. We developed an ML-based feed selection algorithm that classifies feeds into four quality classes and uses the feeds from the highest quality class for selection. We evaluated this algorithm against other methods, such as PageRank-based selection, and found that the ML algorithm performed better according to our notion of feed quality. The feed selection problem can also be solved by generating a ranked list of feeds. In our future work we aim to study learning the feed rank based on the preference order of any pair of feeds and compare it with the classification version.

This paper also presented the effect of previews on improving the search result quality. The $DCG(5)$ relevance metric increased up to 3%. The generated previews also covered a broad spectrum (90%) of all the recency-sensitive queries. We also showed that a PreGen-assisted search system reduces the end-to-end latency of the affected documents by 50% on average.

7. REFERENCES

[1] Salford systems. treenet, cart, random forests manual.

[2] Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei, and Zhong Su. Optimizing web search using social annotations. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 501–510, New York, NY, USA, 2007. ACM.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[4] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Karolina Buchner, Ruiqiang Zhang, Ciya Liao, and Fernando Diaz. Towards Recency Ranking in Web Search. *Third ACM International Conference on Web Search and Data Mining*, 2010.

[5] Jonathan L. Elsas, Jaime Arguello, Jamie Callan, and Jaime G. Carbonell. Retrieval and feedback models for blog feed search. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 347–354, New York, NY, USA, 2008. ACM.

[6] E. Frank, M.A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I.H. Witten. Weka: A machine learning workbench for data mining. *Data Mining and Knowledge Discovery*

Handbook: A Complete Guide for Practitioners and Researchers, pages 1305–1314, 2005.

[7] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[8] Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina. Can social bookmarking improve web search? In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 195–206, New York, NY, USA, 2008. ACM.

[9] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, July 2000.

[10] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM.

[11] Craig Macdonald and Iadh Ounis. Key blog distillation: ranking aggregates. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1043–1052, New York, NY, USA, 2008. ACM.

[12] G. Mishne. Using blog properties to improve retrieval. *Proceedings of ICWSM*, 2007, 2007.

[13] Rudy Prabowo and Mike Thelwall. A comparison of feature selection methods for an evolving rss feed corpus. *Inf. Process. Manage.*, 42(6):1491–1512, 2006.

[14] Matthew Richardson, Amit Prakash, and Eric Brill. Beyond pagerank: machine learning for static ranking. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 707–715, New York, NY, USA, 2006. ACM.

[15] I. Rose, R. Murty, P. Pietzuch, J. Ledlie, M. Roussopoulos, and M. Welsh. Cobra: Content-based filtering and aggregation of blogs and RSS feeds. In *Proceedings of the 4th USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2007)*, Cambridge, MA, 2007.

[16] RSSMicro. Feedrank: Rssmicro search. [Online; accessed 10-January-2010].

[17] J.J. Samper, P.A. Castillo, L. Araujo, JJ Merelo, Ó. Cordón, and F. Tricas. NectaRSS, an intelligent RSS feed reader. *Journal of Network and Computer Applications*, 31(4):793–806, 2008.

[18] Bin Tan, Xuehua Shen, and ChengXiang Zhai. Mining long-term search history to improve search accuracy. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 718–723, New York, NY, USA, 2006. ACM.

[19] D. Wall. An insider's secret to seriously high rankings with Yahoo. *http://www.seochat.com/c/a/Yahoo-Optimization-Help/An-Insiders-Secret-To-Seriously-High-Rankings-With-Yahoo*, 2005.

[20] Guangyu Zhu and Gilad Mishne. Mining rich session context to improve web search. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1037–1046, New York, NY, USA, 2009. ACM.