Time is of the Essence: Improving Recency Ranking Using Twitter Data

Anlei Dong Ruiqiang Zhang Pranam Kolari Jing Bai
Fernando Diaz Yi Chang Zhaohui Zheng
Yahoo! Inc.
701 First Avenue, Sunnyvale, CA 94089
{anlei, ruiqiang, pranam, jingbai, diazf, yichang, zhaohui}@yahoo-inc.com

Hongyuan Zha
College of Computing
Georgia Institute of Technology
Atlanta, GA 30032
zha@cc.gatech.edu

ABSTRACT

Realtime web search refers to the retrieval of very fresh content which is in high demand. An effective *portal* web search engine must support a variety of search needs, including realtime web search. However, supporting realtime web search introduces two challenges not encountered in non-realtime web search: quickly crawling relevant content and ranking documents with impoverished link and click information. In this paper, we advocate the use of realtime micro-blogging data for addressing both of these problems. We propose a method to use the micro-blogging data stream to detect fresh URLs. We also use micro-blogging data to compute novel and effective features for ranking fresh URLs. We demonstrate these methods improve effective of the portal web search engine for realtime web search.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Twitter, recency ranking, recency modeling

1. INTRODUCTION

An effective portal web search engine must satisfy a wide range of queries. At a high level, these include, for example, navigational queries (e.g. 'yahoo') and transactional queries (e.g. 'red shoes') [5]. However, a portal's query population can be further segmented into more granular categories. Each query class may require a different ranking strategy and an effective system will support a broad set of query classes.

Recency sensitive queries refer to queries where the user expects documents which are both topically relevant as well as fresh. For example, consider the occurrence of some natural disaster such as an earthquake. A user interested in this topic probably wants to find documents which are both relevant (e.g. discuss aspects of the earthquake) and timely (e.g. discuss very recent information).

A web search engine must effectively handle time sensitive queries because failures can be more severe for this class than for other query classes. This is case for two reasons. First, time sensitive queries are more likely to suffer from the *zero recall problem*, the failure to index *any* relevant documents for a query. This results from the fact that these queries often refer to events for which documents have been lightly published. Zero recall queries are detrimental because no amount of effort—through reformulation or scanning—can expose the user to relevant content. Second, even when relevant content exists in the index, a user's need for relevant content is immediate. She may be less willing to invest the effort to search for relevant content through reformulation.

We hypothesize that information from a micro-blogging site can be exploited to improve search engine performance for recency sensitive queries. Micro-blogging refers to a web publishing system where posts are severely constrained in size. These constraints allow rapid publishing from a variety of interfaces (e.g. laptops, SMS) and encourage low-cost, realtime updates and links on developing topics. In our experiments, we use data from Twitter, a micro-blogging site where posts consist of no more than 140 characters. These posts are generated frequently and include a rich linking structure.[1]

In this paper, we provide empirical evidence supporting the following claims,

1. Twitter is likely to contain URLs of uncrawled documents likely to be relevant to recency sensitive queries.
2. The text of Twitter posts can be used to expand document text representations.
3. The social network of Twitter users can be used to improve ranking.

We test these claims using a real portal web search engine and evaluate performance using queries submitted to this engine.

[1]There are a variety of other sources which fall under our definition of micro-blogging. These include instant messaging status messages and social network status updates.

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

2. RELATED WORK

2.1 Ranking Twitter Content

Most of the prior work on Twitter and ranking deals with ranking individual tweets. Although Twitter maintains a specialized search engine (http://search.twitter.com/), there exist several vertical search engines which index content across realtime data (e.g. blogs, collaborative bookmarking sites)[2]. While these search engines are able to return very fresh documents, they can often suffer from coverage or ranking issues. For example, consider a breaking-news query. Ranked tweets consist of very brief comments on the news topic from random Twitter users. While such content may satisfy some searchers, other searchers may desire a more sophisticated ranking algorithm incorporating authority or integrating content outside of Twitter. As an alternative, a portal web search engine may decide to integrate Twitter content into general web search results. However, for these queries, Twitter content might obscure more relevant documents. Furthermore, this content also may hurt the user experience for those who are not familiar with Twitter; this is especially true for general web search users. Besides ranking individual tweets, Bing Twitter Search [3] also provides search results for the URLs referred to by Twitter users. The contents of this dedicated search website are all extracted from Twitter data. Our approach is to surface the URLs posted to Twitter on a general search results page. Even though we use the Twitter content in order to perform this blending, we never expose the user to this content. In this way, we can be more confident that the results are both high quality and comprehensive.

2.2 Studying Twitter

Twitter as a research topic has been investigated by researchers in social network anaslysis. Java *et al.* [18] study the topological and geographical properties of Twitter social network, and find that people use Twitter to talk about their daily activities and to seek or share information. More importantly, their analysis shows that users with similar intentions connect with each other. Huberman *et al.* [14] use Twitter data to confirm that users' attentions limit the number of people with whom they interact in a social network. Hughes *et al.* [15] examine Twitter usage as a result of an unexpected event. Compared to general Twitter behavior, they find that Twitter messages sent during unexpected events contain more information broadcasting. Jansen *et al.* [16] investigate Twitter as a form of sharing consumer opinions concerning brands, and found that discuss the implications for corporations using microblogging as part of their overall marketing strategy. Krishnamurthy *et al.* [21] identify distinct classes of Twitter users and their behaviors, geographic growth patterns and current size of the network, and compare crawl results obtained under rate limiting constraints. Shamma *et al.* [25] compare Twitter messages in the context of live media events. They find that analysis of Twitter usage patterns around this media event can yield significant insights into the semantic structure and content of the media object.

Twitter has also been studied in the context of education [3, 10], communication [27] and collaboration [13].

2.3 Recency Ranking

We use Twitter in order to address recency sensitive queries. Previous work has focused on detecting recency sensitive queries in the context of selectively displaying news articles [8, 20]. As we will see in Section 3, this approach has several shortcomings which would be addressed by incorporating recency into the general web search results. In this respect, our work builds on our prior results work in recency ranking [9]. We extend this work by using Twitter to quickly update our index and generate new features.

[2] http://collecta.com/
http://www.oneriot.com/
http://www.yourversion.com/
[3] http://bing.com/twitter

3. MOTIVATION

3.1 Challenges in Recency Ranking

Many existing search engines address recency sensitive queries by selectively integrating content from a specialized news index [8, 20]. The integration process usually comes in the form of a small display containing a small number of documents from the news index. Unfortunately, relying on this approach alone suffers from several problems,

- For some recency sensitive queries, relevant documents are not news documents but web pages.
- A relevant document may be published on a low-priority news site.
- Even when relevant news content exists, the web search results may remain stale.

In order to provide for a consistently relevant experience for all users, we believe that addressing the recency ranking should focus on web results directly.

General web search algorithms incorporate a wide variety of signals when computing the rank of a document. These include query match signals (e.g. how often query words appear in different sections) as well as query-independent signals (e.g. PageRank, popularity, aggregated clicks). Unfortunately, many of these latter features may not be accurately represented in fresh documents. Therefore, even when relevant documents have been indexed, several issues might exist which prevent effective retrieval for time sensitive queries. For example,

- Fresh documents may have very few—if any—in-links, affecting link-based authority metrics.
- Fresh documents may have very few—if any—clicks, affecting click-based authority metrics.

Coping with poor link information can be problematic because it would require a second tier of crawling to monitor links to fresh content. Coping with poor click information requires the search to surface the fresh documents; this is precisely the problem we are addressing.

3.2 An Opportunity

In the context of recency ranking, micro-blogging data can address challenges in recency ranking. First, micro-blogging links include both news and non-news URLs. This allows us to gather information about relevant non-news documents and improve the web results. Second, micro-blogging links are posted according to users' diverse and dynamic browsing priorities, as opposed to a crawl policy attempting to predict that priority. Third, the social network unique to our micro-blogging data provides a method for computing authority for fresh documents. Furthermore, a micro-blogged URLs often include meta-data such as the messages containing this URL. This meta-data can be used to compute ranking features likely to be correlated with relevance.

Our system naturally breaks down into two parts: crawling and ranking. Crawling micro-blogged URLs requires addressing spam

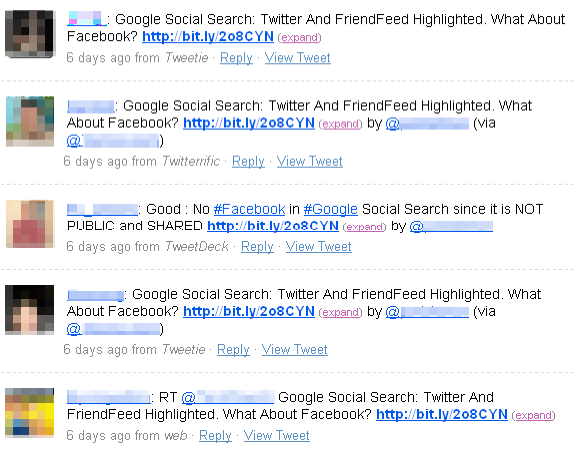

Figure 1: A typical tweet message accompanying a tiny URL. This specific tiny URL is tweeted by five unique users with their respective tweet messages. Such tweet messages are indicative of the content of the tiny URL. The users' photos and names are mosaicked to protect privacy.

and non-recency URLs (Section 4). Ranking micro-blogged URLs requires incorporating these documents into a larger web ranking system (Section 5). In the subsequent sections, we will address each of these tasks in the context of Twitter data.

4. CRAWLING TWITTER URLS

There are several disadvantages to naïvely crawling all Twitter URLs. The URLs posted by Twitter users include a significant amount of links to spam, adult and self-promotion pages, which probably should not be included in results for time sensitive queries. Furthermore, realtime crawling and indexing of *all* Twitter URLs would require considerable overhead.

We employ simple heuristics to filter out most of the undesired Twitter URLs. In order to address spam and self-promotion URLs, we discard URLs referred to by the same Twitter user more than two times. Furthermore, we discard URLs only referred to by one Twitter user. This removes spurious URLs from our crawl.

We did an experiment to study the effect of these filtering rules. During the period of 15:00∼20:00 (UTC) on September 9th, 2009, there are a total of 1,069,309 URLS that were posted in Twitter. After applying the first rule, 713,178 URLs remained, 66.7% of the original URLs. In other words, 33.3% of the URLs are filtered out as they are very likely to be spam, adult or self-promotion pages. After applying the second rule, 63,184 URLs remained, only 5.9% of the original URLs. We also manually inspected a sample of discarded URLs and found that most of them were undesirable.

5. RANKING TWITTER URLS

Machine-learned ranking, the underlying algorithm for many web search engines, refers to the automatic construction of a ranking function which optimizes retrieval performance metrics [6,7,11,19, 22,26,29]. Machine-learned rankers have demonstrated significant and consistent gains over many manually tuned rankers for a variety of data sets. Optimization usually is formulated as learning a ranking function from preference data in order to minimize a loss function (e.g., the number of incorrectly ordered documents pairs in the training data). Different algorithms view the preference learning problem from different perspectives. For example, RankSVM [19] uses support vector machines; RankBoost [11] applies the idea of boosting from weak learners; GBrank [28] uses gradient boosting with decision tree; RankNet [6] uses gradient boosting with neural network.

A typical learning-to-rank framework must be *trained* using some editorially labeled data. This is accomplished by sampling a set of query-url pairs for human judgment. Each of these query-url pairs is given a grade based on the degree of relevance (e.g. bad match, excellent match, etc). Each query-url pair is then represented by a feature vector consisting of variables indicating query term matches, link-structure-based features of the document (e.g. PageRank), and click-based features of the document. Using the features of each document as well as the editorially labeled training data, a machine learned ranking system can predict effective rankings for queries unseen in the training data.

One of the most important aspects of a learning-to-rank system is the feature set. We differentiate between the types of features we use in our work. *Content features* refer to those features which are functions of the content of the document (e.g. query term matches, proximity between query terms). We expect these features to behave normally for fresh URLs. *Aggregate features* refer to those features which represent a document's long term popularity and usage (e.g. in-link statistics, PageRank, clicks). We expect that these features will be poorly represented for fresh URLs. *Twitter features* refer to those features which are related to tweets containing a URL pointing to the document (e.g. tweet content, number of tweets containing the URL). We expect that these features will be poorly represented—and often non-existent—for those documents not crawled from the Twitter feed.

In the subsequent subsections, we will describe our Twitter features in detail. We will also describe a special training method required for recency sensitive queries. Content and aggregate features are thoroughly described in [1, 23, 28].

5.1 Twitter Features

Twitter is comprised of two parts: a publishing system and a subscription system. Data about the publishing system (i.e. what are users posting) allows us to detect new URLs. The publishing system also provides textual information associated with these new URLs. Data about the subscription system (i.e. which users are following which other users) allows us to measure the qualities of the new URLs. This is only possible because Twitter subscriptions are exposed to the public.

5.1.1 Textual Features

A URL posted to twitter can be associated with the text surrounding it. Figure 1 depicts a set of tweets from users about a common tiny URL. The text in tweets accompanying the tiny URL can provide useful additional information.

Assume we have m tweets and w URLs. Let $\mathbf{M}$ be the $m \times w$ binary matrix represent the occurrence of a URL in a tweet. Assume we have observed v words in all tweets. We define the $m \times v$ matrix $\mathbf{D}$ so that D_{ij} represents the number of times a tweet i contains a term j. In practice, we remove stop words from our vocabulary. We can construct a term vector for a URL, j, as,

$$\mathbf{u}_j^{\mathsf{T}} = \sum_i M_{ij} D_{i\cdot} \tag{1}$$

where $D_{i\cdot}$ represents row i of $\mathbf{D}$. This represents a URL by the combination of tweet contents. A query can also be represented as the $v \times 1$ vector, $\mathbf{q}$, of term occurrences. These representations allow us to use text similarity features in order to predict URL relevance. For example, we can use the cosine similarity between

the URL term vector (Equation 1) and $\mathbf{q}$ in order to determine the similarity between a URL and a query. For a URL, j, the *cosine similarity* feature is defined as,

$$\phi^j_{\text{cosine}} = \frac{\mathbf{u}_j^\mathsf{T}\mathbf{q}}{\|\mathbf{u}_j\|_2\|\mathbf{q}\|_2} \tag{2}$$

By design, tweets are very short pieces of text and therefore are susceptible to problems when applying classic text ranking methods [24]. For example, unmatched terms should be more severely penalized than they are in cosine similarity. For this reason, we also inspect the term overlap as another textual feature. Let $\tilde{\mathbf{D}}$ be the binary version of $\mathbf{D}$ (i.e. $\tilde{D}_{ij} = 1$ if $D_{ij} > 0$; $\tilde{D} = 0$ otherwise). Define $\tilde{\mathbf{q}}$ similarly. The term overlap between a query and the text of a tweet can be represented as,

$$\begin{aligned} \omega_{iq} &= (\tilde{D}_{i\cdot})^\mathsf{T}\tilde{\mathbf{q}} && \text{overlapping terms} \\ \epsilon_{iq} &= \|\tilde{D}_{i\cdot}\|_1 - \omega_{iq} && \text{extra terms} \\ \mu_{iq} &= \|\mathbf{q}\|_1 - \omega_{iq} && \text{missing terms} \end{aligned}$$

were $\|\mathbf{x}\|_1$ is the ℓ_1 norm of $\mathbf{x}$. For a candidate URL j, the *unit match* feature is defined as,

$$\phi^j_{\text{unit}} = \frac{1}{\|\mathbf{q}\|_1}\sum_{i=1}^{m}\epsilon_{iq}^{\alpha}\mu_{iq}^{\beta}\omega_{iq}M_{ij} \tag{3}$$

The parameters α and β control the importance of extra and missing terms. In our experiments, parameters α and β are set to be 0.5 and 0.65 respectively, based on earlier experience with this parameter in a popular search engine.

Finally, we also include a simple *exact match* feature. The feature counts the number of tweet messages in which all query tokens appear contiguously, and in the same order,

$$\phi^j_{\text{exact}} = \frac{1}{\|M_{\cdot j}\|_1}\sum_{i=1}^{m}\text{phraseMatch}(q,i)M_{ij} \tag{4}$$

where $M_{\cdot j}$ returns column j of $\mathbf{M}$ and phraseMatch(q,i) return one if there exact phrase q occurs in tweet i.

5.1.2 Social Network Features

We adopt the convention of representing user data as a social network where vertices represent twitter users and edges represent the follower relationship between them. Mathematically, we represent this graph as a $u \times u$ adjacency matrix, $\mathbf{W}$, where $W_{ij} = 1$ if user i follows user j. In practice, we normalize $\mathbf{W}$ so that $\sum_j W_{ij} = 1$. Given this matrix and an eigensystem, $\mathbf{W}\pi = \lambda\pi$, the eigenvector, π, associated with the largest eigenvalue, λ, provides a natural measure of the centrality of the user [2]. The analog in web search is the PageRank of a document [4]. This eigenvector, π, can be computed using power iteration,

$$\pi_{t+1} = (\lambda\mathbf{W} + (1-\lambda)\mathbf{U})\pi_t \tag{5}$$

where $\mathbf{U}$ is a matrix whose entries are all $\frac{1}{m}$. The interpolation of $\mathbf{W}$ with $\mathbf{U}$ ensures that the stationary solution, π, exists. The interpolation parameter, λ, is set to 0.85. In our experiments, we perform fifteen iterations (i.e. $\tilde{\pi} = \pi_{15}$). If we assume that a user i posted URL j, we define the authority feature of URL j as

$$\phi^j_{\text{authority}} = \pi_i \tag{6}$$

We can also use the authority of the user in the computation of our unit match score (Equation 3). In particular, we define the authority-weighted unit match score as

$$\phi^j_{\text{unit-}\pi} = \frac{1}{\|\mathbf{q}\|_1}\sum_{i=1}^{m}\epsilon_{iq}^{\alpha}\mu_{iq}^{\beta}\omega_{iq}M_{ij}\phi^i_{\text{authority}} \tag{7}$$

$\phi_{\text{other-1}}$	average number of followers for the users who issued the tiny URL
$\phi_{\text{other-2}}$	average post number for the users who issued the tiny URL
$\phi_{\text{other-3}}$	average number of users who retweeted the tweets containing the tiny URL
$\phi_{\text{other-4}}$	average number of users who replied those users that issued the tiny URL
$\phi_{\text{other-5}}$	average number of followings for the users who issued the tiny URL
$\phi_{\text{other-6}}$	average Twitter score of all the users who issued the tiny URL
$\phi_{\text{other-7}}$	number of followers for the user who first issued the tiny URL
$\phi_{\text{other-8}}$	number of posts by the user who first issued the tiny URL
$\phi_{\text{other-9}}$	number of users who retweeted the user who first issued the tiny URL
$\phi_{\text{other-10}}$	number of users who replied the user who first issued the tiny URL
$\phi_{\text{other-11}}$	number of followings for the user who first issued the tiny URL
$\phi_{\text{other-12}}$	Twitter score of the users who first issued the tiny URL
$\phi_{\text{other-13}}$	number of followers for the user who issued the tiny URL with the highest Twitter score
$\phi_{\text{other-14}}$	number of posts by the user who issued the tiny URL with the highest Twitter score
$\phi_{\text{other-15}}$	number of users who retweeted the user who issued the tiny URL and has the highest Twitter score
$\phi_{\text{other-16}}$	number of users who replied the user who issued the tiny URL and has the highest Twitter score
$\phi_{\text{other-17}}$	number of followings for the user who has the highest Twitter score among the users that issued the tiny URL
$\phi_{\text{other-18}}$	Twitter score of the users who issued the tiny URL and who is the highest Twitter score
$\phi_{\text{other-19}}$	number of different users who sent the tiny URL.

Table 1: Twitter features.

5.1.3 Other Features

In addition to the features described in the preceding sections, we can compute simple aggregate metrics of the URL over a period of time (the details on the period selection are in Section 6.1.1). We present these features in Table 1. Some of the features are designed to improve relevance ranking by incorporating Twitter specific features: user's authority estimation from twitter rank (5), which we also call *Twitter user score* or *Twitter score*. For example, the feature $\phi_{\text{other-6}}$ is the average Twitter score of all the users who issued the tiny URL. Over a period of time, there could be many users who issued, replied, or retweeted the tiny URL. This feature is to calculate the average twitter score of all the users. The feature $\phi_{\text{other-12}}$ is the Twitter score of the users who first issued the tiny URL.

The features in Table 1 can be grouped into three set. Features $\phi_{\text{other-1}} - \phi_{\text{other-6}}$ are the average statistics of the users who issued the tiny URL. Using average statistics can improve feature's robustness and discount any bias on a single user. Features $\phi_{\text{other-7}} - \phi_{\text{other-12}}$ are the features related to the user who is the first that issued the tiny URL. We assume the authority of the first issuer may affect the URL importance. Features $\phi_{\text{other-13}} - \phi_{\text{other-18}}$ are the features related to the user who issued the tiny URL but has the highest Twitter score. The user with the highest Twitter score means he/she is the one with the most authority among the users.

In each set, we consider the number of followers, tweets, users being retweeted and replied, and user's Twitter score. Those features estimate the tiny URL's popularity from different aspects. The last feature, $\phi_{\text{other-19}}$, is the number of different users who issued the

tiny URL. The higher the number, the more popular the tiny URL is.

Algorithm 1 Ranking functions used in the ranking system, including ranking functions for documents represented using content and aggregated features ($\mathcal{M}_{\text{regular}}$), only content features ($\mathcal{M}_{\text{content}}$), and twitter and content features ($\mathcal{M}_{\text{twitter}}$). $\mathbf{D}$ represents data set including query-URL pairs with labeled relevance grades. $\mathbf{F}$ represents feature set. TRAIN-MLR$(\mathbf{D}, \mathbf{F})$ is the ranking function learning algorithm, which is based on the training data set $\mathbf{D}$ using feature set $\mathbf{F}$. PREDICT$(\mathbf{D}, \mathcal{M})$ scores the data set, $\mathbf{D}$ using model $\mathcal{M}$.

TRAIN-MODELS$(\mathbf{D}_{\text{regular}}, \mathbf{D}_{\text{Twitter}})$

$\mathbf{D}_{\text{regular}}$: training data set from regular data
$\mathbf{D}_{\text{Twitter}}$: training data set from Twitter data
1 $\mathcal{M}_{\text{regular}} \leftarrow$ TRAIN-MLR$(\mathbf{D}_{\text{regular}}, \{\mathbf{F}_{\text{content}}, \mathbf{F}_{\text{aggregate}}\})$
2 $\mathcal{M}_{\text{Twitter}} \leftarrow$ TRAIN-MLR$(\mathbf{D}_{\text{Twitter}}, \{\mathbf{F}_{\text{content}}, \mathbf{F}_{\text{Twitter}}\})$
3 $\mathcal{M}_{\text{content}} \leftarrow$ TRAIN-MLR$(\mathbf{D}_{\text{regular}}, \mathbf{F}_{\text{content}})$
4 $\mathbf{y}_{\text{Twitter}} \leftarrow$ PREDICT$(\mathbf{D}_{\text{Twitter}}, \mathcal{M}_{\text{content}})$
5 $\mathcal{M}_{\text{composite}} \leftarrow$ TRAIN-MLR$(\mathbf{D}_{\text{Twitter}}, \{\mathbf{y}_{\text{Twitter}}, \mathbf{F}_{\text{Twitter}}\})$

5.2 Ranking

5.2.1 Relevance Models

The most straightforward method to train a ranking function for Twitter documents is to follow the standard procedure prescribed above: sample query-URL pairs (including both regular URLs and Twitter URLs) and label them, train a ranking function, and apply this function on future queries. Unfortunately, there are far more regular URLs than Twitter URLs. Twitter feature values will be missing from the majority amount of regular URLs. As a result, the machine-learned ranking system will likely ignore these features.

We employ a divide-and-conquer strategy, which fully exploits the available ranking features for regular URLs and Twitter URLs respectively. As shown in Algorithm 1, for regular URLs, we learn a regular ranking function $\mathcal{M}_{\text{regular}}$ based on content features and aggregate features; for Twitter URLs, we learn a Twitter ranking function $\mathcal{M}_{\text{twitter}}$ based on content features and Twitter features. In addition to these two ranking functions, we also learn a ranking function $\mathcal{M}_{\text{content}}$ only based on content features. We train this model for comparison in our experiments.

We use the Gradient Boosted Decision Tree (GBDT) algorithm [12] to learn a ranking function for TRAIN-MLR in Algorithm 1. GBDT is an additive regression algorithm consisting of an ensemble of trees, fitted to current residuals, gradients of the loss function, in a forward step-wise manner. It iteratively fits an additive model as

$$f_t(x) = T_t(x; \Theta) + \lambda \sum_{t=1}^{T} \beta_t T_t(x; \Theta_t)$$

such that certain loss function $L(y_i, f_T(x+i))$ is minimized, where $T_t(x; \Theta_t)$ is a tree at iteration t, weighted by parameter β_t, with a finite number of parameters, Θ_t and λ is the learning rate. At iteration t, tree $T_t(x; \beta)$ is induced to fit the negative gradient by least squares. That is

$$\hat{\Theta} := \text{argmin}_\beta \sum_{i}^{N} (-G_{it} - \beta_t T_t(x_i); \Theta)^2$$

where G_{it} is the gradient over current prediction function

$$G_{it} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{t-1}}$$

The optimal weights of trees β_t are determined by

$$\beta_t = \text{argmin}_\beta \sum_{i}^{N} L(y_i, f_{t-1}(x_i) + \beta T(x_i, \theta))$$

$(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{regular}})$	Use $\mathcal{M}_{\text{regular}}$ on regular and Twitter URLs.
$(\mathcal{M}_{\text{content}}, \mathcal{M}_{\text{content}})$	Use $\mathcal{M}_{\text{content}}$ on regular and Twitter URLs.
$(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{content}})$	Use $\mathcal{M}_{\text{regular}}$ on regular URLs and $\mathcal{M}_{\text{content}}$ on Twitter URLs.
$(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{twitter}})$	Use $\mathcal{M}_{\text{regular}}$ on regular URLs and $\mathcal{M}_{\text{twitter}}$ on Twitter URLs.
$(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{composite}})$	Use $\mathcal{M}_{\text{regular}}$ on regular URLs and $\mathcal{M}_{\text{composite}}$ on Twitter URLs.

Table 2: Runs used in our experiments

5.2.2 Ranking Models

To rank the URLs (both regular URLs and Twitter URLs) with the given query, we apply our relevance models to regular URLs and Twitter URLs. We then rank the URLs by sorting their ranking scores. Because we always model a relevance grade, the predicted grades are calibrated and comparable. As a result, we can directly blend regular URLs and Twitter URLs according to their ranking scores.

We study the five ranking approaches listed in Table 2. We adopt the convention of representing algorithms as model tuples where $(\mathcal{M}_x, \mathcal{M}_y)$ means 'apply $\mathcal{M}_x$ to regular URLs and $\mathcal{M}_y$ to Twitter URLs'.

Our two baseline approaches apply Twitter-unaware models to all URLs being considered. The $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{regular}})$ baseline can be interpreted as applying a general ranking algorithm to all URLs. In the cases where URLs lack valid aggregate features, we set their aggregate feature values as default value zeros. The $(\mathcal{M}_{\text{content}}, \mathcal{M}_{\text{content}})$ baseline indirectly promotes the Twitter URLs focusing the ranking on features shared between both regular and Twitter URLs.

We also consider approaches which apply different models to different URLs. The $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{content}})$ run preserves the production ranking for regular URLs but applies a content-only model to Twitter URLs. We expect this model to leverage the content features learned across the pooled data to rank Twitter URLs. This behavior may not be present in a model which was trained using both content and aggregate features. The $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{twitter}})$ run explores the benefit of combining features specific to Twitter with content features. One drawback of the $\mathcal{M}_{\text{twitter}}$ model is the relatively small training pool: we expect far fewer example documents with both Twitter and content features defined compared to those

with only content features defined. As a result, we also consider a final run, $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{composite}})$, which uses the content model score as a feature for Twitter URLs.

6. METHODS AND MATERIALS

6.1 Data

6.1.1 Documents and Queries

Our data set of queries and tweets was collected over a few different days. We use a web index from a large-scale commercial search engine. Our Twitter stream consists of all tweets from the Twitter Firehose.[4] On each day of the study, we collected queries issued to the search engine between 23:00∼23:59 UTC. Recall that we are interested using Twitter only for time-sensitive queries, those queries which expect fresh URLs. Therefore, we only consider queries which are classified as "time-sensitive" queries using an automatic classifier [9].

We constructed two sets of URLs for each day,

- *regular URLs*: in the search engine index during 23:00∼23:59,
- *Twitter URLs*: posted by Twitter users during the 9-hour period before the query time (i.e., 14:00∼22:59).

The 9-hour period is heuristically determined only for experimental purposes. This period corresponds to the hours during which Twitter volume is highest. For each query, we apply simple text-matching rules on Twitter URLs in order to remove non-relevant URLs. For example, we remove URLs from consideration if there are no query term matches in the body or title.

For the regular URLs, we consider the top ten URLs as decided by the production ranking algorithm of the search engine. We also consider all Twitter URLs. Recall that we train the ranking function $\mathcal{M}_{\text{content}}$ in Section 5.2, which is only based on content ranking features (from document title and body). For each query, we apply $\mathcal{M}_{\text{content}}$ to the Twitter URLs and heuristically determine a ranking score threshold: if a Twitter URL has higher ranking score than this threshold, we keep this Twitter URL for the query; otherwise, we discard this Twitter URL. Therefore, we obtain Twitter URLs with reasonable relevance to the query.

6.1.2 Labels

Given these queries, we are interested in labeling the relevance of documents in both sets. We ask human editors to label each tuple $\langle \text{query}, \text{URL}, t_{\text{query}} \rangle$ with a relevance grade. We apply five judgement grades on query-URL pairs: perfect, excellent, good, fair and bad. For editors to judge the tuple, we ask them to first grade it by non-temporal relevance, such as intent, usefulness, content, user interface design, and domain authority.

Because we are interested in time-sensitive queries, we categorize documents according to their temporal properties. We present the classes we consider in Table 3. We would like to promote 'very fresh' documents and demote 'outdated' documents. Those documents which are 'temporally insensitive' or 'somewhat fresh' are unlikely to affect the recency of a ranking so we leave those documents in the original order. We can combine these temporal categories with the relevance judgments using *recency demotion* [9],

- shallow demotion (1-grade demotion): if the result is 'somewhat outdated', it should be demoted by one grade (e.g., from excellent to good);
- deep demotion (2-grade demotion): if the result is 'totally outdated', it should be demoted by two grades (e.g., from excellent to bad).

[4] http://apiwiki.twitter.com/

document class	example document
time insensitive	wikipedia entry
time sensitive	
very fresh	very recent news article
somewhat fresh	a day-old news article
somewhat outdated	old news article
totally outdated	very old news article

Table 3: Document classes for time-sensitive queries. The "very fresh" documents are those which were published on the same day as the query.

6.1.3 Testing Data

We collect testing data from the search engine and Twitter stream on the day of October 14th, 2009. The time-windows and procedures are described in Section 6.1.1. The testing data consists of 3781 regular query-URL pairs and 769 Twitter query-URL pairs, in which there are unique 392 queries.

For regular query-URL pairs, content features and aggregate features are extracted. For Twitter query-URL pairs, content features and Twitter features are extracted. In the experiment, there are 66 content features, 454 aggregate features and 23 Twitter features.

6.1.4 Training Data

There are two training data sets. One set is used to train ranking function for regular URLs, another set is to train ranking function for Twitter URLs.

For regular training data set, we collect a large amount of 206,249 query-URL pairs. Content features and aggregate features are extracted from this training set.

For Twitter training data set, we collect the Twitter data from two days: October 12th, 2009 and October 19th, 2009. The time-windows and procedures are described in Section 6.1.1. The data from these two days are combined together, and there are totally 8025 query-URL pairs.

To make it fair for our experiment evaluations, we remove the queries from this training set that are similar to or same as the queries in the testing set. After removing these similar (same) queries, the Twitter training data set consists of 5006 query-URL pairs and there are 1800 associated unique queries. Content features and Twitter features are extracted.

6.2 Evaluation

We desire an evaluation metric which supports graded judgments and penalizes errors near the beginning of the ranked list. In this work, we use *discounted cumulative gain* (DCG) [17],

$$\text{DCG}_n = \sum_{i=1}^{n} \frac{G_i}{\log_2(i+1)} \tag{8}$$

where i is the position in the document list, G_i is the function of relevance grade. Because the range of DCG values is not consistent across queries, we adopt the *normalized discounted cumulative gain* (NDCG) as our primary ranking metric,

$$\text{NDCG}_n = Z_n \sum_{i=1}^{n} \frac{G_i}{\log_2(i+1)} \tag{9}$$

where Z_n is a normalization factor, which is used to make the NDCG of ideal list be 1. We use NDCG_1 and NDCG_5 to evaluate the ranking results.

Our recency demotion guidelines conflate relevance and recency. In order to evaluate freshness in isolation, we also include a freshness metric based on DCG, *discounted cumulative freshness* (DCF),

$$\text{DCF}_n = \sum_{i=1}^{n} \frac{F_i}{\log_2(i+1)} \tag{10}$$

where i is the position in the document list, F_i is the freshness label (1 or 0). A query may have multiple very fresh documents, for example when multiple news sources simultaneously publish updates to some ongoing news story. Note that DCF is a recency measurement that is independent of overall relevance. Therefore, when we evaluate a ranking, we should first consider demoted NDCG which represents the overall relevance, then inspect the value of the DCF. We define *normalized discounted cumulative freshness* (NDCF) as in Equation 9.

In our experiments, we use the following freshness criterion: if the main content of a document is created on the same day as query time, this document is labeled as a very fresh document. Using this criterion, editors can easily and quickly evaluate documents. For a very small portion of breaking-news queries, it is possible that a document becomes stale only a few hours after its creation because more related documents are created with significantly newer contents. However, the current criterion, in general, appropriately reflects the fresh document distribution for most breaking-news queries.

7. RESULTS

7.1 Data analysis

Before presenting results demonstrating our ranking improvements, we offer some descriptive statistics of our collected data.

Recall that we used an automatic classifier to extract our candidate queries. We were interested in validating the accuracy of this pool of queries. For the queries in the testing set, we randomly selected 242 queries and asked editors to judge whether these queries were breaking news queries or not. Our criterion for breaking-news query was stricter than those used to train the automatic classifier [9] we use in Section 6.1.1. Specifically, we asked editors to label a query as a breaking news query only if there is at least one new document created within the last 24 hours that is relevant to the query. Our editorial experiment confirmed that 212 (91.7%) of the queries were breaking new queries.

We can also measure the quality of Twitter URLs in aggregate by inspect the freshness and relevance grades of our Twitter and regular URLs. We present the distribution of grades broken down by source in Table 5. We observe that the quality of Twitter URLs is better than Regular URLs in sense of both relevance and recency. Of the Twitter URLs, 53.8% are very fresh documents while, for Regular URLs, this fraction is only 19.4%. Furthermore, the relevance grade distribution does not change after recency demotion, which means there are no stale documents in Twitter URLs. This confirms our assumption that the URLs extracted from Twitter data are generally very fresh. At same time, the overall relevance quality of Twitter URLs is also higher than of regular URLs. The percentages of perfect and excellent Twitter URLs are higher than those of Regular URLs, while the percentages of fair and bad Twitter URLs are lower than those of regular URLs. This means Twitter URLs are potentially useful to improve ranking *for time sensitive queries*.

Finally, while we use our user authority feature, $\tilde{\pi}$ as a ranking feature, it is also worth noting that it can be used in isolation to qualitatively inspect a set of users. We computed $\tilde{\pi}$ for ten million users and present the top users associated with high values of $\tilde{\pi}_i$ in Table 5. Though the top users are largely dominated by celebrities, many popular bloggers, and news sources are also surfaced as highly authoritative. Therefore, we expect that this feature will be valuable when used in conjunction with our other features.

(a) relevance grade (demoted)

	Perfect	Excellent	Good	Fair	Bad
Regular	0.7%	17.0%	44.9%	26.6%	10.9%
Twitter	13.0%	33.4%	41.0%	20.7%	3.6%

(b) relevance grade (non-demoted)

	Perfect	Excellent	Good	Fair	Bad
Regular	0.9%	23.0%	61.0%	36.1%	14.8%
Twitter	13.0%	33.4%	41.0%	20.7%	3.6%

(c) recency label

	Fresh	Non-fresh
Regular	19.4%	80.6%
Twitter	53.8%	46.2%

Table 4: Data distribution in sense of relevance grade and recency label.

7.2 Ranking results

As shown in Table 6, our proposed approach which blends Twitter content into the standard ranked list significantly improves ranking in sense of both relevance and recency. We notice this improvement across all of our metrics.

The baseline approach, $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{regular}})$, uses content and aggregate features for both regular and Twitter URLs. This prevents Twitter URLs from being promoted because Twitter URLs suffer from feature impoverishment. We expect this behavior given our discussion of the role aggregate features in Section 5.

The content-only approach, $(\mathcal{M}_{\text{content}}, \mathcal{M}_{\text{content}})$, underperforms the baseline approach, $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{regular}})$, because it does not use aggregate features. Nevertheless, as a result, Twitter URLs have no disadvantage when they compete with regular URLs. The NDCF values are improved which means more fresh documents (i.e., Twitter documents) are promoted to the top ranking results. However, in sense of relevance represented by NDCG values, there is no improvement because the absence of aggregate features hurts the ranking of regular URLs.

When we consider models which leverage the representational strength of each URL class, performance improves across metrics. For example, using the content and aggregate features for regular URLs and content features for Twitter URLs, $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{content}})$, improves both relevance and recency metrics. If we enrich the representation of the Twitter URLs, $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{twitter}})$, we get the best performance across all metrics. This means that we were able to successfully incorporate realtime web content without hurting relevance. In fact, we improve relevance.

Our experiments did not confirm that $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{composite}})$ leveraged the additional training data from regular URLs for content features. Our results show that the performance of this algorithm is very similar to using $\mathcal{M}_{\text{twitter}}$, a model built with much less training data.

Table 7 qualitatively illustrates the behavior of our algorithms. Compared with the baseline result, $(\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{regular}})$, our Twitter-

Table 6: Ranking results comparison. All the improvements are statistically significant (p-value < 0.01). Bold entries indicate the top performance for that metric.

	Top 1						Top 5					
	$NDCG_{demote,1}$		$NDCG_{nodemote,1}$		$NDCF_1$		$NDCG_{demote,5}$		$NDCG_{nodemote,5}$		$NDCF_5$	
$(\mathcal{M}_{regular}, \mathcal{M}_{regular})$	0.588		0.611		0.474		0.666		0.681		0.518	
$(\mathcal{M}_{content}, \mathcal{M}_{content})$	0.570	-3.2%	0.610	-0.2%	0.513	$+7.5\%$	0.652	-2.1%	0.682	$+0.3\%$	0.587	$+11.7\%$
$(\mathcal{M}_{regular}, \mathcal{M}_{content})$	0.600	$+1.8\%$	0.618	$+1.2\%$	0.520	$+8.8\%$	0.680	$+2.1\%$	0.690	$+1.3\%$	0.569	$+8.9\%$
$(\mathcal{M}_{regular}, \mathcal{M}_{twitter})$	**0.720**	$+18.4\%$	**0.708**	$+13.7\%$	**0.717**	$+33.8\%$	**0.739**	$+9.9\%$	**0.729**	$+6.5\%$	**0.736**	$+29.6\%$
$(\mathcal{M}_{regular}, \mathcal{M}_{composite})$	0.715	$+17.9\%$	0.702	$+13.0\%$	0.747	$+36.5\%$	0.735	$+9.4\%$	0.723	$+5.8\%$	0.756	$+31.4\%$

Table 7: An example of ranking recency improvement. The query is *wwe captain lou albano*, and the query issue time is during 23:00∼23:59 UTC on October 14th, 2009.

(a) Ranking result by baseline approach $(\mathcal{M}_{regular}, \mathcal{M}_{regular})$.

rank (rank in (b))	URL	grade	fresh	from Twitter
1 (2)	http://en.wikipedia.org/wiki/Captain_Lou_Albano	good	no	no
2 (3)	http://www.wwe.com/superstars/halloffame/inductees/captainloualbano/	excellent	no	no
3 (6)	http://www.wrestlingmuseum.com/pages/bios/halloffame/albanobio.html	good	no	no
4 (4)	http://www.wwe.com/superstars/halloffame/inductees/captainloualbano/photos/	good	no	yes
5 (7)	http://wjz.com/entertainment/captain.lou.albano.2.1248290.html	excellent	yes	no

(b) Ranking result by new approach $(\mathcal{M}_{regular}, \mathcal{M}_{Twitter})$.

rank (rank in (a))	URL	grade	fresh	from Twitter
1 (8)	http://edition.cnn.com/2009/SHOWBIZ/TV/10/14/obit.albano/	excellent	yes	yes
2 (1)	http://en.wikipedia.org/wiki/Captain_Lou_Albano	good	no	no
3 (2)	http://www.wwe.com/superstars/halloffame/inductees/captainloualbano/	excellent	no	no
4 (4)	http://www.wwe.com/superstars/halloffame/inductees/captainloualbano/photos/	good	no	yes
5 (12)	http://www.wrestling-edge.com/wwe-news/wwe-hall-of-famer-capt-lou-albano-passes-away.html	fair	yes	yes

userID	User/Type
twitter	Twitter Official
kimkardashian	Kim Kardashian
aplusk	Ashton Kutcher
denise_richards	Denise Richards
ddlovato	Demetria Lovato
katyperry	Katy Perry
khloekardashian	Khloe Kardashian
johncmayer	John Mayer
astro_mike	Mike Massimino
robdyrdek	Rob Dyrdek
...	...
nasa	NASA Space Program
mcuban	Mark Cuban
wired	Wired Magazine
problogger	Darren Rowse
chrispirillo	Chris Pirillo
cbsnews	CBS News
jkottke	Jason Kottke

Table 5: Result of Markov chain analysis on the twitter follower graph. The top half shows the top ten users, most, if not all dominated by celebrities. However, a select sub-set from the top hundred users features many news media sites, popular bloggers, and technology authorities.

Table 8: Twitter feature importance list. The Twitter feature definitions can be found in Table 1.

Twitter feature	importance rank	importance score
ϕ_{unit}	9	31.1
$\phi_{other\text{-}17}$	10	27.1
$\phi_{other\text{-}15}$	11	26.6
$\phi_{other\text{-}3}$	13	22.8
$\phi_{other\text{-}1}$	18	16.7

based algorithm significantly promotes relevant and recent content to the top of the ranked list. Note that in this example, none of the displayed URLs is stale; thus, the recency demotion grades and non-demotion grades are always equal.

7.3 Feature importance

We have demonstrated that Twitter features can significantly boost the performance of a recency sensitive ranker. It is worth investigating which Twitter features in particular were highly valued by our model. As presented in Algorithm 1, the Twitter ranking function, $\mathcal{M}_{Twitter}$, uses both content features and Twitter features. We can compute the importance of each feature by the method proposed in [12]. We rank features by the descending order of the importance and show the top five Twitter features in Table 8. The feature importance score is on a scale of $[0, 100]$.

The most important Twitter feature is ϕ_{unit}, which is the unit match feature between query and tweet text as defined in (3). This means the text similarity between a query and a tweet in general highly correlates with the relevance between the query and the

Twitter URL posted in the tweet. Furthermore, this text-proximity feature is highly complementary to the content ranking features (e.g., text-proximity features based on document title and body) and can be see as a proto-'anchor text' for new URLs.

The other important Twitter features include $\phi_{\text{other-17}}$, $\phi_{\text{other-15}}$, $\phi_{\text{other-3}}$ and $\phi_{\text{other-1}}$, which are described in Table 1. These features represent the authority and activity of the users that are related to the Twitter URLs from different aspects. For example, $\phi_{\text{other-17}}$ is the number of the followings for the user who has the highest twitter user score among the users that issued the Twitter URL.

8. DISCUSSION

Our experiments demonstrate that there is a clear advantage to using Twitter both as a source of URLs as well as a source of evidence for fresh URLs. These results complement existing Twitter ranking results insofar as they demonstrate the efficacy of a blending approach as opposed to a vertical selection approach [8]. On the other hand, our Twitter features can be further developed with a knowledge from social studies of Twitter (Section 2.2).

There is also an opportunity for more sophisticated spam detection in our work. The Twitter URLs we used in our study have undergone multiple filtering steps during both crawling (Section 4) and ranking (Section 6.1.1). As shown in Table 5, the Twitter URLs in our experiments are both highly relevant and fresh. We can potentially increase the size of candidate Twitter URLs by relaxing our filtering rules. However, as a result, the quality of the candidate Twitter URLs will almost certainly become degrade. Our model, then, would need to incorporate spam filtering. For example, it would need to learn that certain features (e.g. number of users posting the URL) are even more indicative of a low quality (or spam) document. Fortunately, our training procedure supports such an approach. As a search system begins to index Twitter URLs in near real time, spam detection will become increasingly important.

We have also assumed that the set of Twitter URLs is disjoint from the set of regular URLs. This models a retrieval system as being composed of two parts: a long term index and a realtime index. The long term index contains content whose freshness is limited by the effectiveness of the crawler finding and incorporating new data. The realtime index consists of very fresh content with impoverished representation. In many cases an overlap will exist. In our experiments, using a commercial search engine, roughly 10% of the Twitter URLs were already indexed. To simplify experimentation, we treated these URLs as Twitter URLs. However, as the long term index begins to accumulate fresher and fresher content—for example, through more effective/adaptive crawl policies or superior indexing architectures, the overlap will increase. As a result, developing models which support Twitter, content, *and* aggregate feature will be important.

Finally, we have only touched the surface of blended ranking. In our experiments we combined scores based on relevance alone. This suffers from a few problems inherent to ranking in general. For example, this creates a problem for multiple intent queries (e.g. 'election results' could refer to one of several regional elections) or queries which deserve summarization (e.g. 'candidate speeches' may be satisfied by a summary including documents which together discuss several candidates' recent speeches). As a result, ranking diversity will be an important area of research. Traditional diversification approaches focus on content-based similarity of documents from the same index. It is unclear how these approaches can be extended to rankings which combine content from several indexes.

9. CONCLUSION

We have presented preliminary evidence supporting the claim that micro-blogging data can be exploited to improve web ranking for recency sensitive queries. Our approach is based on preserving the quality of data presented to the general web searcher by only using micro-blog data as evidence for discovering and ranking URLs. For recency queries, we demonstrated that both relevance-based and freshness-based metrics can be improved with our approach.

More generally, our results demonstrate the power of leveraging widespread user behavior for recency sensitive queries. Although other sources of user behavior information exist (e.g. click logs, toolbar data), Twitter is one of the only sources which is both public and widely adopted. This makes Twitter a valuable source of realtime user behavior for institutions lacking access to more sensitive log data.

In the future, we are interested in improving spam detection, enriching the features we extract from Twitter using regular URL information and results from Section 2.2, and incorporating diversity. Furthermore, we are interested in synthesizing signals from Twitter streams with other sources of realtime evidence into a cohesive recency ranking module. Finally, if demographic information about Twitter users can be extracted or predicted, then this resource can also be used for conducting personalization experiments.

10. ACKNOWLEDGES

We thank Alex Smola, Tamas Sarlos, Deepayan Chakrabarti, Ciya Liao and Jyh-Herng Chow for their helpful discussions.

11. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of 29th ACM SIGIR*, 2006.

[2] P. Bonacich. Factoring and weighting approaches to clique identification. *Journal of Mathematical Sociology*, 2:113–120, 1972.

[3] K. Borau, C. Ullrich, J. Feng, and R. Shen. Microblogging for language learning: Using twitter to train communicative and cultural competence. In *International Conference on Web Based Learning (ICWL) 2009*, 2009.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Proceedings of International Conference on World Wide Web*, 1998.

[5] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *Proc. of Intl. Conf. on Machine Learning*, 2005.

[7] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise. *Proceedings of ICML conference*, 2007.

[8] F. Diaz. Integration of news content into web results. *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM)*, pages 182–191, 2009.

[9] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 11–20, New York, NY, USA, 2010. ACM.

[10] J. C. Dunlap and P. R. Lowenthal. Tweeting the night away: Using twitter to enhance social presence. In *Journal of Information Systems Education Special Issue, Impacts of Web 2.0 and Virtual World Technologies on IS Education*, 2009.

[11] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Proceedings of International Conference on Machine Learning*, 1998.

[12] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[13] C. Honeycutt and S. C. Herring. Beyond microblogging: Conversation and collaboration via twitter. In *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*, pages 1–10, 2009.

[14] B. A. Huberman, D. M. Romero, and F. Wu. Social networks that matter: Twitter under the microscope. Dec 2008.

[15] A. L. Hughes and L. Palen. Twitter adoption and use in mass convergence and emergency events. In *Proceedings of the 6th International Conference on Information Systems for Crisis Response and Management*, 2009.

[16] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Twitter power: Tweets as electronic word of mouth. *Journal of the American Society for Information Science and Technology*, pages 1–20, 2009.

[17] K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20:422–446, 2002.

[18] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65, New York, NY, USA, 2007. ACM.

[19] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[20] A. C. König, M. Gamon, and Q. Wu. Click-through prediction for news queries. In *SIGIR 2009*, 2009.

[21] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about twitter. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 19–24, New York, NY, USA, 2008. ACM.

[22] T. Y. Liu. Learning to rank for information retrieval. *Tutorial on WWW conference*, 2009.

[23] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[24] D. Metzler, S. T. Dumais, and C. Meek. Similarity measures for short segments of text. In *ECIR*, pages 16–27, 2007.

[25] D. Shamma, L. Kennedy, and E. Churchill. Tweet the debates: Understanding community annotation of uncollected sources. In *Proceedings of the ACM International Conference on Multimedia*. ACM, 2009.

[26] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *Proceedings of the 30th ACM SIGIR*, 2007.

[27] D. Zhao and M. B. Rosson. How and why people twitter: the role that micro-blogging plays in informal communication at work. In *GROUP '09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 243–252, New York, NY, USA, 2009. ACM.

[28] Z. Zheng, H. Zha, K. Chen, and G. Sun. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th ACM SIGIR conference*, 2007.

[29] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. *NIPS*, 2007.