Improving Recency Ranking Using Twitter Data

YI CHANG, ANLEI DONG, PRANAM KOLARI, RUIQIANG ZHANG, YOSHIYUKI INAGAKI, and FERNANODO DIAZ, Yahoo! Labs
HONGYUAN ZHA, Georgia Institute of Technology
YAN LIU, University of Southern California

In Web search and vertical search, *recency ranking* refers to retrieving and ranking documents by both relevance and freshness. As impoverished in-links and click information is the the biggest challenge for recency ranking, we advocate the use of Twitter data to address the challenge in this article. We propose a method to utilize Twitter TinyURL to detect fresh and high-quality documents, and leverage Twitter data to generate novel and effective features for ranking. The empirical experiments demonstrate that the proposed approach effectively improves a commercial search engine for both Web search ranking and tweet vertical ranking.

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

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Recency ranking, Twitter, tweet ranking

ACM Reference Format:
Chang, Y., Dong, A., Kolari, P., Zhang, R., Inagaki, Y., Diaz, F., Zha, H., and Liu, Y. 2013. Improving recency ranking using Twitter data. ACM Trans. Intell. Syst. Technol. 4, 1, Article 4 (January 2013), 24 pages.
DOI = 10.1145/2414425.2414429 http://doi.acm.org/10.1145/2414425.2414429

1. INTRODUCTION

Recency ranking refers to ranking documents by both freshness and relevance. As a large number of new Web pages are created every minute, information on those old Web pages is outdated quickly. If stale documents are presented to a user, the user's search experience would be seriously degraded. In the most recent years, temporal dimension of Web search has been studied from different perspectives, such as Web dynamics, temporal features, modeling, etc. Recency has been taken into account for some specific applications, such as research publication database, news article search, etc. However, large-scale comprehensive study on recency ranking is still very limited.

Recency-sensitive queries refer to those queries whose expected results are both fresh and relevant. For example, consider the occurrence of some natural disaster, such as an earthquake, a user who inputs related queries wants to find some documents both relevant (e.g., talking about earthquakes) and fresh (e.g., talking about the most recent earthquake event). Typical recency-sensitive queries include the topics

This submission is the extended version of the conference paper, which appears. In *Proceedings of the 19th International World Wide Web Conference (WWW10)*, 331–340.
Authors' addresses: Y. Chang (corresponding author), A. Dong, P. Kolari, R. Zhang, Y. Inagaki, F. Diaz, Yahoo! Labs; email: yichang@yahoo-inc.com; H. Zha, Georgia Institute of Technology; Y. Liu, University of Southern California.

DOI 10.1145/2414425.2414429 http://doi.acm.org/10.1145/2414425.2414429

of natural disasters, major sports events, celebrity gossips, political breaking stories, etc. Although recency-sensitive queries only account for a small percentage of the total query volume of a search engine (usually less than 10% of the traffic), these queries represent the most urgent information need, which is critical to user experience.

There are several challenging problems in recency ranking. Firstly, crawling and indexing fresh and high-quality content in real time from the Web is challenging. Secondly, detecting the actual age of a document is difficult, since a malicious user can easily change the time-stamps of numerous pages. As a result, precise temporal features cannot be extracted from Web pages, and only weak temporal evidence can be obtained. Lastly, given very limited in-links and click information, how to balance relevance and freshness is the most critical challenge for a learning-to-rank algorithm.

In this article we leverage Twitter, which is the most popular micro-blogging service, to improve search quality for recency-sensitive queries. Micro-blogging refers to a set of Web publishing systems where messages are strictly constrained in length. These constraints allow rapid publishing from a variety of interfaces (e.g., laptop, SMS), and encourage low-cost, real-time updates on developing topics. Considering the unique characteristic of Twitter, we utilize different perspectives of Twitter information, and successfully improve recency ranking, in particular, improve both Web search ranking and tweet vertical ranking. We enumerate our primary contributions as follows.

(1) Tweet information can be used to improve recency ranking.
(2) The social network of Twitter users can be leveraged to improve recency ranking.
(3) Retweet information, which represents the quality of a tweet, can be incorporated to improve recency ranking.

The rest of this article is organized as follows. We first introduce the background and characteristics of Twitter in Section 2, followed by data analysis of Twitter data in Section 3. In Section 4 we discuss recency ranking models which fully leverage Twitter information. We then introduce Twitter new features in Section 5. We next introduce experimental results for Twitter TinyURL ranking in Section 6, and present empirical experiments for tweet vertical ranking in Section 7. We list the related works in Section 8, and we draw our conclusions in Section 9.

We mainly enrich this article on the basis of our previous conference paper [Dong et al. 2010b] in the following parts: Twitter data analysis is added in Section 3; two sets of new features are introduced: retweet user features in Section 5.4, retweet language model features in Section 5.5; more promising experimental results are added in Section 6 with more features; tweet vertical ranking is included in Section 7.

2. TWITTER: AN OPPORTUNITY FOR RECENCY RANKING

Micro-blogging is one of the most popular online communication paradigms nowadays, and short informal messages are shared by users to their social networks. Twitter, being the most popular and successful micro-blogging service, attracts hundreds of millions of users worldwide. The text messages on Twitter are called *tweets*, which are 140 characters or less. The receivers of these messages are termed as *followers*. Sample of tweets are shown in Figure 1. The 140-character limit spurs the usage of URL shortening services, such as bit.ly.[1] Twitter users often post shortened URL links, which are called *Twitter TinyURLs* in this article. According to Alonso et al. [2010], the quality of tweets with TinyURLs is higher than those tweets without TinyURLs. If multiple tweets refer to the same TinyURL, as an example shown in Figure 2, we

[1]http://bit.ly.

Fig. 1. Typical tweets on Twitter, with retweet and TinyURL examples.

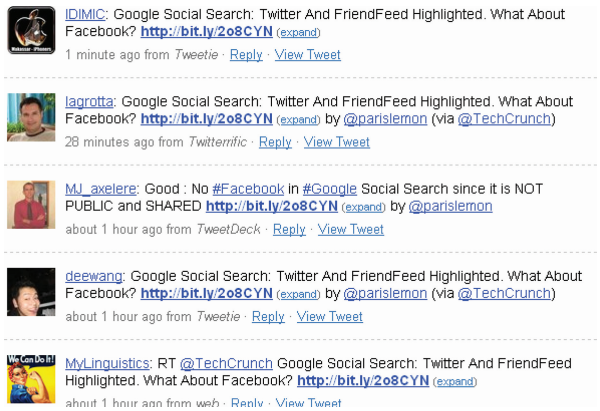

Fig. 2. Typical tweets on Twitter, and the TinyURL is referred by 4 different tweets.

can consider these tweets providing some supplementary information which is similar to the information provided by anchor texts. A tweet can be rebroadcasted by a user to its followers, and such rebroadcasting behavior is termed as *retweet behavior*, and each rebroadcasted tweet is also called a *retweet*.

A commercial search engine could leverage Twitter content to enrich its index and improve its recency ranking. First, Twitter TinyURLs include both news and non-news URLs, which allows a search engine to gather fresh content to improve the Web results. Second, Twitter TinyURLs are posted according to users' diverse and dynamic browsing priorities, as opposed to a crawler policy which attempts to predict such priorities. Third, the Twitter social network provides a method to compute authority of fresh documents. Lastly, a Twitter TinyURL also relates to meta-data, such as tweets, which can be used to generate new ranking features.

3. TWITTER DATA ANALYSIS

Tweets posted by Twitter users are mixed with spam, nonsense, and self-promotion information, therefore, how to distinguish high-quality data from noisy data is the foremost challenge to utilize Twitter data.

Table I. Fraction of Overall Tweets Getting Retweeted on Twitter

Numer of Retweets	%volume
$\geq$ #1	1.48%
$\geq$ #2	0.31%
$\geq$ #4	0.13%

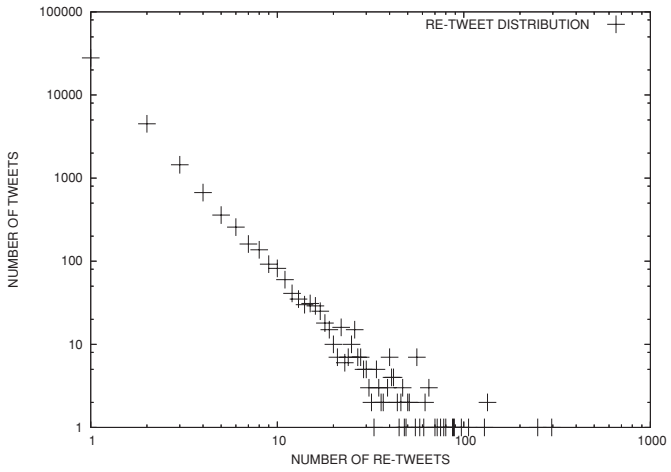

Fig. 3. The number of retweets received by tweets follows a power law with slope -1.6, and a small percentage of tweets receive most of the retweets.

An effective, bottom-up collective behavior, which can be utilized to identify high-quality tweets, is the retweet behavior. Similar to in-links on a popular Web page, the number of retweeted times somewhat reflects its popularity and attractiveness of a tweet. Therefore, analyzing retweet behavior would help us to understand the overall quality of tweets.

Table I shows the overall fraction of tweets that get retweeted on Twitter, based on the sampled data in September 2009. Overall, only 1.48% of total tweets are retweeted, and the number of tweets with high retweeted rate ($\geq$ 4) is only 0.13%. A plot of the distribution of retweets is shown in Figure 3, on a logarithmic scale. The x-axis plots the number of retweets, and the y-axis plots their distribution over all tweets. The distribution of retweets follows the power law, which is similar to the distribution of in-links on the Web [Broder et al. 2000]: the slope of the retweet power law is -1.6, which is higher than the slope of Web in-links.

In general, retweet characteristics can be briefly summarized as follows: first, the volume of retweets is fairly light as a fraction of all tweets; secondly, retweets follow a typical power law, with a tiny portion of tweets gaining widespread popularity.

4. RANKING MODEL WITH TWITTER DATA

4.1. Learning-to-Rank Base Model

Machine-learned ranking, the underlying algorithm for many Web search engines, refers to the automatic construction of a ranking function which optimizes retrieval relevance metrics [Burges et al. 2005; Cao et al. 2007; Freund et al. 1998; Joachims 2002; Zheng et al. 2007]. Optimization usually is formulated as learning a ranking function from preference data in order to minimize a loss function, for example, the number of incorrectly ordered document pairs in training data. Different algorithms

view the preference learning problem from different perspectives. For example, RankSVM [Joachims 2002] uses support vector machines; RankBoost [Freund et al. 1998] applies the idea of boosting from weak learners; GBrank [Zheng et al. 2007] uses gradient boosting with decision trees; RankNet [Burges et al. 2005] uses gradient boosting with neural networks.

A typical learning-to-rank framework must be trained using some editorially labeled data. This is accomplished by sampling a set of query-document pairs for human judgment, and each query-document pair is given a grade based on the degree of relevance, for example, bad match, excellent match, etc. Then, each query-document pair is represented by a feature vector consisting of variables, such as query term matching, link-structure-based features of the document, click-based features of the document, etc. Using the features of query-document pairs as well as the corresponding editorial labels, a machine-learned ranking model could be trained with training data. In the scenario of recency ranking with Twitter data, a document could refer to either a Twitter TinyURL, or an individual tweet.

One of the most important aspect of a learning-to-rank system is the feature set, and we differentiate the types of features in this work. *Content features* refer to those features which are functions of the content of the document, for example, query term matches, proximity between query terms, etc. *Aggregate features* refer to those features representing a document's long-term popularity or usage (e.g., in-link statistics, PageRank, clicks), which are poorly represented for those fresh documents. Content and aggregate features are thoroughly described and explained in Agichtein et al. [2006] and Chapelle and Chang [2011]. *Twitter features* refer to those features which are related to tweets or TinyURLs, for example, tweet content, number of tweets containing the TinyURL, etc.

In this article, we use the Gradient Boosted Decision Tree (GBDT) algorithm [Friedman 2001] to learn the basic ranking functions. 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

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

such that a 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

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

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

$$G_{it} = \left[\frac{\partial L(y_i,\ f(x_i))}{\partial f(x_i)}\right]_{f=f_{t-1}}. \tag{3}$$

The optimal weights of trees β_t are determined by

$$\beta_t = \text{argmin}_\beta \sum_{i}^{N} L(y_i,\ f_{t-1}(x_i) + \beta T(x_i,\theta)). \tag{4}$$

Table II. Ranking Model Combination Used in Our Experiments

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

4.2. Ranking Model for Twitter TinyURL

The most straightforward method to train a ranking function for those documents redirected via Twitter TinyURLs is to follow the standard procedure prescribed before: sample query-URL pairs (including both regular URLs and Twitter TinyURLs) and label them, and train a ranking function, then apply this function on future queries. Unfortunately, there are far more regular URLs than Twitter TinyURLs, and the machine-learned ranking model would likely ignore those Twitter features.

We employ a divide-and-conquer strategy, which fully exploits the available ranking features for regular URLs and Twitter TinyURLs 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 TinyURLs, 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. In this algorithm, function Train-MLR uses GBDT which is introduced in the previous section. In the empirical experiments part, we train multiple models for comparison.

ALGORITHM 1: Ranking functions used in the system, including ranking functions for documents using content and aggregated features ($\mathcal{M}_{\text{regular}}$), only content features ($\mathcal{M}_{\text{content}}$), and twitter features ($\mathcal{M}_{\text{twitter}}$). $\mathbf{D}$ represents the data set including query-URL pairs with labeled relevance grades. $\mathbf{F}$ represents the 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, where $\mathbf{D}$ uses 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($\mathbf{D}_{\text{Twitter}}, \{\mathbf{y}_{\text{Twitter}}, \mathbf{F}_{\text{Twitter}}\}$)

4.3. Ranking Model Combination for Twitter TinyURL

To rank the URLs with the given query, we apply our relevance models to regular URLs and Twitter TinyURLs. Since the predicted scores are calibrated and comparable, we can directly blend regular URLs and Twitter TinyURLs according to their ranking scores. We study 5 different ranking approaches listed in Table II, where $(\mathcal{M}_x, \mathcal{M}_y)$ means applying $\mathcal{M}_x$ to regular URLs and $\mathcal{M}_y$ to Twitter TinyURLs.

Our two baseline approaches apply Twitter-unaware models to all URLs being considered. ($\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{regular}}$) 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 0. ($\mathcal{M}_{\text{content}}, \mathcal{M}_{\text{content}}$) indirectly promotes the Twitter TinyURLs via focusing the ranking on features shared by both regular URLs and Twitter TinyURLs.

We also consider approaches which apply different models to different types of documents. ($\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{content}}$) preserves the production ranking for regular URLs but applies a content-only model to Twitter TinyURLs. We expect this model to leverage the content features learned across the pooled data to rank Twitter TinyURLs. ($\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{twitter}}$) explores the benefit of combining features specific to Twitter with content features, while one drawback of the $\mathcal{M}_{\text{twitter}}$ model is the relatively small training pool. We also consider another combination, ($\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{composite}}$), which uses the content model score as a feature to rank Twitter TinyURLs.

5. TWITTER FEATURES FOR RECENCY RANKING

In this section, we propose to use a few different categories of features which are generated from Twitter data or Twitter users, including Twitter user social network features (Section 5.1), tweet textual features (Section 5.2), Twitter TinyURL attribute features (Section 5.3) and retweet-based features (Section 5.4 and 5.5).

5.1. Twitter User 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 relationships 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 users [Bonacich 1972]. The analog in Web search is the PageRank [Brin and Page 1998]. The eigenvector π can be computed as

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

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 15 iterations (i.e., $\tilde{\pi} = \pi_{15}$).

If we assume that a user i posts a TinyURL j, we define the authority feature of TinyURL 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 (Eq. (10)). In particular, we define the authority-weighted unit match score as

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

As Twitter Firehose doesn't include follower relationship data, we use a large-scale crawling of the Twitter network to capture this follower relationship. We compute $\tilde{\pi}$ for about 10 million users, collected by crawling the Twitter social network in 2009. To compute eigenvector with such a big matrix is time consuming. As the topology of the follower relationship won't change dramatically in a short period, we can update this type of feature periodically, such as once a month. We show the top users associated with high values of $\tilde{\pi}_i$ in Table III. Although the top users are largely dominated

Table III. Result of Top Twitter Users from the Follower Graph

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

The first half shows the top 10 users, which are dominated by celebrities; the second half shows a selective subset from the top hundred users, including news media sites, popular bloggers.

by celebrities, many popular bloggers or news sources are also surfaced as highly authoritative.

5.2. Tweet-TinyURL Textual Features

A TinyURL posted to Twitter can be associated with the text surrounding it. Figure 2 depicts a set of tweets from different users but containing the same tiny URL. The text in tweets accompanying the TinyURL can provide useful additional information.

Assume we have m tweets and w TinyURLs. Let $\mathbf{M}$ be the $m \times w$ binary matrix representing the occurrence of a TinyURL 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 TinyURL j as

$$\mathbf{u}_j^\mathsf{T} = \sum_i M_{ij} D_{i\cdot}, \tag{8}$$

where $D_{i\cdot}$ represents row i of $\mathbf{D}$. This represents a TinyURL by the combination of tweet contents. A query $\mathbf{q}$ can also be represented as the $v \times 1$ vector, and the vector is composed of occurrences of each term. These representations allow us to use text similarity features in order to predict TinyURL relevance. For example, in order to determine the similarity between a TinyURL and a query, we can use the cosine similarity between the TinyURL term vector (Eq. (8)) and the query $\mathbf{q}$. For a TinyURL j, the *cosine similarity* feature is defined as

$$\phi^j_{\text{cosine}} = \frac{\mathbf{u}_j^\mathsf{T}\mathbf{q}}{\|\mathbf{u}_j\|_2 \|\mathbf{q}\|_2}. \tag{9}$$

However, tweets are too short to apply classic text ranking methods [Metzler et al. 2007]. For example, unmatched terms should be more severely penalized 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), and define $\tilde{\mathbf{q}}$ similarly. The term overlap between a query and a tweet can be represented as

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

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

$$\phi^j_{\text{unit}} = \frac{1}{\|\mathbf{q}\|_1} \sum_{i=1}^{m} \epsilon^{\alpha}_{iq} \mu^{\beta}_{iq} \omega_{iq} M_{ij}, \tag{10}$$

where 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, which is based on previous studies [Bai et al. 2008]. In addition, hashtags (prefixed by #) in tweets are also treated as plain texts in feature generation.

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

$$\phi^j_{\text{exact}} = \frac{1}{\|M_{\cdot j}\|_1} \sum_{i=1}^{m} \text{phraseMatch}(q, i) M_{ij}, \tag{11}$$

where $M_{\cdot j}$ returns column j of $\mathbf{M}$, and phraseMatch(q, i) returns 1 if exact phrase q occurs in tweet i.

5.3. Twitter TinyURL Attribute Features

In addition to the features described in the previous subsections, we also generate a set of attribute features for each Twitter TinyURL over a period of time. We present these features in Table IV. Some of the features are designed to improve relevance ranking by incorporating Twitter user social network features (Eq. (5)), which is also called *Twitter user score*. For example, feature $\phi_{\text{attr-6}}$ is the average Twitter user score of all the users who issued this particular TinyURL, since there could be many users who issued, replied, or retweeted this TinyURL over time.

The features in Table IV can be grouped into 3 sets. Features $\phi_{\text{attr-1}} - \phi_{\text{attr-6}}$ are average numbers of the users who issued the TinyURL. Using average number can improve a feature's robustness and discount any bias over a single user. Features $\phi_{\text{attr-7}} - \phi_{\text{attr-12}}$ are the features related to the user who originally issued the TinyURL. We assume the authority of the original issuer may strongly reflect the TinyURL's importance. Features $\phi_{\text{attr-13}} - \phi_{\text{attr-18}}$ are the features related to the user issuing the TinyURL and with the highest Twitter user score. The user with the highest Twitter score means he/she is the one with the most authority or popularity among the users. In each set, we consider the number of followers, tweets, users being retweeted or replied, and user's Twitter score. Those features estimate the popularity of the TinyURL from different aspects. In addition, feature $\phi_{\text{attr-19}}$ is the number of different users who are issuing the TinyURL. Intuitively, the higher the number, the more popular the TinyURL is.

Table IV. Twitter TinyURL Attribute Features

$\phi_{\text{attr-1}}$	average number of Followers of the users who issued the TinyURL
$\phi_{\text{attr-2}}$	average post number of the users who issued the TinyURL
$\phi_{\text{attr-3}}$	average number of users who retweeted the Tweets containing the TinyURL
$\phi_{\text{attr-4}}$	average number of users who replied those users issuing the TinyURL
$\phi_{\text{attr-5}}$	average number of followings of the users who issued the TinyURL
$\phi_{\text{attr-6}}$	average Twitter user score of all the users who issued the TinyURL
$\phi_{\text{attr-7}}$	number of Followers of the users who originally issued the TinyURL
$\phi_{\text{attr-8}}$	number of posts of the users who originally issued the TinyURL
$\phi_{\text{attr-9}}$	number of users who retweeted the users originally issuing the TinyURL
$\phi_{\text{attr-10}}$	number of users who replied the users originally issuing the TinyURL
$\phi_{\text{attr-11}}$	number of followings of the users who originally issued the TinyURL
$\phi_{\text{attr-12}}$	Twitter user score of the users who originally issued the TinyURL
$\phi_{\text{attr-13}}$	number of Followers of the users who issued the TinyURL with the highest Twitter score
$\phi_{\text{attr-14}}$	number of posts of the users who issued the TinyURL with the highest Twitter score
$\phi_{\text{attr-15}}$	number of users who retweeted the user issuing the TinyURL and with the highest Twitter score
$\phi_{\text{attr-16}}$	number of users who replied the user issuing the TinyURL and with the highest Twitter score
$\phi_{\text{attr-17}}$	number of followings of the user who has the highest Twitter user score among the users issuing the TinyURL
$\phi_{\text{attr-18}}$	Twitter user score of the user who issued the TinyURL and with the highest Twitter score
$\phi_{\text{attr-19}}$	number of different users who sent the TinyURL.

Table V. Retweet User Features that Capture Users whose Tweets Are Highly Retweeted

1	ϕ_{mean_rt}	mean of Retweets of all Tweets by the user
2	ϕ_{sd_rt}	standard deviation of Retweets of all Tweets by the user
3	ϕ_{rt}	total Retweets of all Tweets by the user
4	ϕ_{tweet}	total Tweets by the user

5.4. Retweet User Feature

As we discussed in Section 3, to distinguish high-quality tweets from the low-quality ones is an important task, if we want to make good use of Twitter data. Generally speaking, retweet-related features could represent the likelihood of a tweet to be retweeted, which indirectly indicates the quality of the tweet. We extract a number of features based on the past tweet behavior of the user, as well as retweets on these historic tweets. The intuition is that tweets from those users who got retweeted in the past are more likely to attract retweets in the future.

The set of features extracted from a user based on retweet behavior are summarized in Table V. One of the discriminative features, ϕ_{mean_rt}, is the mean of retweet counts of all tweets by a user. A user who consistently shares high-quality tweets is expected to have a high ϕ_{mean_rt}. However, to directly account for the consistency among these top users, we also capture the standard deviation ϕ_{sd_rt} as an additional feature. In addition to these two attributes, we incorporate total number of tweets ϕ_{tweet}, and total number of retweets from a user as ϕ_{rt}.

Candidate users surfaced as having high ϕ_{mean_rt} are shown in Table VI. Interestingly, the top users in Table III and Table VI are totally different, which implies that Twitter user social network features and retweet user features are complementary.

Table VI. Candidate Top Users Featuring High ϕ_{mean_rt}

userID	User/Type
shitmydadsays	Pop Culture
barackobama	Politics
revrunwisdom	Spiritual
pink	Music
tfln	Texts from Last Night
thecharlieday	Charlie Day
themime	Entertainment
theonion	News
wordpress	Product
iphone_dev	Product
tinybuddha	Spiritual

These users tend to have political and spiritual themes or are news breakers and celebrities.

Table VII. The Size of Language Model Trained from Historical Retweets, and Tweets without Retweets

ngram	retweeted LM	non-retweeted LM
unigram	108,891	643,795
bigram	300,298	2,272,674
trigram	16,085	187,178

5.5. Retweet Language Model Features

Using Language Models (LM) is based on the intuition that the text styles of those tweets being retweeted differ significantly from those that are not retweeted, and such a difference can be observed with a traditional LM. Here, the text style refers to word distribution, writing style, grammar, etc. Users are more likely to use formal words to compose higher-quality tweets, while they may use short and informal vocabulary to post less interesting tweets.

Language models for general information retrieval have been well studied [Lafferty et al. 2001; Ponte and Croft 1998; Zhai and Lafferty 2004]. Traditional usage of a language model is to measure the degree of relevance between a document and a query. For this purpose, both a document language model and a query language model are built. Lafferty et al. [2001] present a method to combine document models and query models using a probabilistic ranking function based on Bayesian theory. Zhai and Lafferty [2004] experiment with language model approaches using a range of data smoothing techniques, including Good-Turing estimate, curve-fitting functions, and model combinations.

In this work we use LMs to distinguish retweeted tweets from nonretweeted tweets. To build the LMs, we randomly select 3 million retweeted tweets and 3 million non-retweeted tweets in September 2009, to train a retweeted LM and a nonretweeted LM. We use the SRILM software package [Stolcke 2002] to train a trigram LM for each, and its smoothing method is the Good-Turing approach. The LM is a back off type which means if a higher-order n-gram is unseen in the training data, it is approximated by a lower-order n-gram. The size of the retweet LM and the nonretweet LM are shown in Table VII.

To share some basic characteristics of the developed LMs, we use the log-likelihood test [Dunning 1993] which is quite common in the language modeling literature. The log-likelihood test has been successfully used to compare two language models to

Table VIII. Key Indicator Terms in Tweets that Are Not Retweeted

i	my	so
im	me	lol
was	just	:)
but	it	u
:d	that	going
am	watching	yeah
got	haha	oh
:(	work	(:
had	then	its
hey	good	like
been	sleep	go
back	bored	#mobsterworld
hope	gonna	bed
ok	cant	home
wait	homework	school
class	tired	night

Table IX. Key Indicator Terms in Tweets that Are Retweeted

#iranelection	#tcot	social
#quote	#ffnew	
your	#thugs	marketing
our	blog	obama
#p2	check	tea
#tlot	success	iphone
article	follow	up
#followfriday	free	get
win	top	#jesus
#sex	retweet	business
#teaparty	socialist	white
communist	socialism	health
facebook	#truth	list

quantify surprise, and representative terms in blog profiles [Mishne 2007]. Prominent terms in the nonretweeted LM are shown in Table VIII, whereas terms appearing in the retweeted LM are shown in Table IX. Clearly, the nonretweeted LM features terms that are informal and highly subjective. The retweeted LM feature terms are more formal, yet also include many hashtags, which are prefixed by #. Both tables indicate the potential of content-based approaches for identifying high-quality tweets of broader interests.

The calculation of LM score, in the developed language models (unigram, bigram, and trigram), is formalized as follows. Given a tweet consisting of a word sequence, $w_0, w_1, \cdots, w_N$, the language model score, in the case of trigram, is defined as

$$P(w_0 w_1 \cdots w_N) = P(w_0)P(w_1|P(w_0)\prod_{i=2}^{N} P(w_i|w_{i-1}w_{i-2}). \tag{12}$$

We generate 4 LM features, as detailed in Table X. Instead of using the original LM score directly, we use the logarithm of the score, and normalize it by the size of the tweet.

Table X. Retweet LM Features Captured from Language Models

1	ϕ_{lm_sub}	the difference in scores of non-retweeted LM with retweeted LM
2	ϕ_{lm_div}	the non-retweeted LM score divided by the retweeted LM score
3	ϕ_{lm_nort}	the LM score using non-retweeted LM
4	ϕ_{lm_rt}	the LM score using retweeted LM

6. RANKING TWITTER TINYURL

6.1. Crawling Twitter TinyURL

There are several disadvantages to naively crawl all Twitter TinyURLs. Those URLs posted by Twitter users also include a significant amount of links to spam, adult, or self-promotion pages. Furthermore, real-time crawling and indexing of all Twitter TinyURLs would require considerable overhead.

In order to address this issue, we employ a couple of heuristics rules to filter out most of the undesired Twitter TinyURLs. Firstly, if a Twitter TinyURL is referred by the same Twitter user for more than 2 times, we discard this TinyURL since it is usually a spam or self-demotion link. Secondly, if a Twitter TinyURL is referred by only one Twitter user, we discard this TinyURL as well, since this link is not popular among Twitter users. The remaining Twitter TinyURLs are crawled and indexed as shallow crawling seeds, which could drive more high-quality fresh content.

We finish 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 totally 1,069,309 TinyURLS referred on Twitter. After we apply the first rule, there are 713,178 TinyURLs remaining, which is 66.7% of the original URLs. In other words, 33.3% of the TinyURLs are filtered out as they are very likely to be spam, adult, or self-promotion pages. After we apply the second rule, only 63,184 TinyURLs are left, that is to say, only 5.9% of the original Twitter TinyURLs are kept as shallow crawling seeds.

6.2. Query and Document

Our dataset of queries and tweets is collected over a few different days. We use a Web search index from Yahoo! search engine. Our Twitter stream consists of all tweets from Twitter Firehose[2]. On each sequential day of the study, we collect queries issued to the search engine between 23:00∼23:59 UTC, and we only consider queries which are classified as recency-sensitive queries using an automatic classifier [Dong et al. 2010b].

We construct two sets of URLs for each day.

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

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 text-matching rules on Twitter TinyURLs in order to remove nonrelevant URLs. For example, we remove URLs from consideration if there are no query term matches in title or body sections.

For the regular URLs, we consider top 10 URLs from the production ranking algorithm of the search engine. Recall that we train the ranking function $\mathcal{M}_{\text{content}}$ in Section 4.2, 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 TinyURLs and heuristically determine a ranking score threshold: if a Twitter TinyURL has higher ranking score than this threshold, we keep this Twitter TinyURL for the query; otherwise,

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

Table XI. Document Classes for Recency-Sensitive Queries

document class	example documents
time insensitive	wikipedia entries
time sensitive	
very fresh	very recent news articles
somewhat fresh	one-day-old news articles
somewhat outdated	old news articles
totally outdated	very old news articles

The *very fresh* documents are published on the same day as the query.

we discard this Twitter TinyURL. Therefore, we obtain Twitter TinyURLs with reasonable relevance to the query.

6.3. Editorial Label

Given these queries, we are interested in labeling the relevance of documents in both datasets. We ask human editors to label each tuple ⟨query, URL, grade⟩ with a relevance grade. We apply a five-grade scale on each query-URL pair: perfect, excellent, good, fair, and bad. For editors to judge the tuple, we ask them to first grade it by nontemporal relevance, such as intent, usefulness, content, user interface design, and domain authority.

Because we are interested in recency-sensitive queries, we categorize documents according to their temporal properties. We present the classes we consider in Table XI. 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* rules [Dong et al. 2010b].

—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 fair).

6.4. Evaluation Metrics

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) [Jarvelin and Kekalainen 2002]

$$\mathrm{DCG}_n = \sum_{i=1}^{n} \frac{G_i}{\log_2(i+1)}, \tag{13}$$

where i is the position in the document list, and G_i is the function of relevance grade. Because the range of DCG values is not consistent across different queries, we adopt the *Normalized Discounted Cumulative Gain* (NDCG) as our primary ranking metric

$$\mathrm{NDCG}_n = Z_n \sum_{i=1}^{n} \frac{G_i}{\log_2(i+1)}, \tag{14}$$

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

Our recency demotion guidelines combine relevance and recency. In order to evaluate freshness in isolation, we also include a freshness metric, *Discounted Cumulative Freshness* (DCF)

$$\mathrm{DCF}_n = \sum_{i=1}^{n} \frac{F_i}{\log_2(i+1)}, \tag{15}$$

where i is the position in the document list, and F_i is the freshness label. A query may have multiple *very fresh* documents, for example when multiple news sources simultaneously publish updates of some ongoing news story. Since DCF is a recency measurement which is independent of overall relevance, when we evaluate a ranking, we should first consider demoted NDCG which represents the overall relevance, and then inspect the value of the DCF. We define *Normalized Discounted Cumulative Freshness* (NDCF) in the same way as in Eq. (14).

In our experiments, we use the following freshness criterion: if the main content of a document is created on the same day as the 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 recency-sensitive queries, it is possible that a document becomes stale after only a few hours because a lot of related documents are created with significantly newer contents. Yet, this criterion appropriately reflects the distribution of fresh documents for most of the recency-sensitive queries.

6.5. Training and Testing Data

There are two training datasets. One set is used to train ranking functions for regular URLs, the other set is to train ranking functions for Twitter TinyURLs. For the regular training dataset, we collect a large amount of 206,249 query-URL pairs. Content features and aggregate features are extracted from this training set. For the Twitter training dataset, we collect the Twitter data from two days in October 2009. The time-window and procedure are described in Section 6.2. The data from these two days are combined together, and there are 8,025 query-URL pairs in total. We also remove those queries from this training set which are similar to or same as the queries in the testing set. After removing these similar or same queries, the Twitter training dataset consists of 5006 query-URL pairs and there are 1800 associated unique queries. Content features and Twitter features are extracted.

We collect our testing dataset from the search engine and Twitter stream on a different day in October 2009. The testing dataset consists of 3781 regular query-URL pairs and 723 Twitter query-TinyURL pairs, in which there are 392 unique queries. For regular query-URL pairs, content features and aggregate features are extracted. For Twitter query-TinyURL pairs, content features and Twitter features (Section 5) are extracted.

6.6. Training Data Analysis

As we use an automatic classifier [Dong et al. 2010b] to extract our candidate queries, we are interested in validating the accuracy of this pool of queries. For the queries in the testing set, we randomly select 242 queries and ask editors to judge whether these queries are recency-sensitive queries or not. Our criterion for recency-sensitive query is stricter than those used to train the automatic classifier [Dong et al. 2010b] in Section 6.2. Specifically, we ask editors to label a query as a recency-sensitive query only if there is at least one new document created within the last 24 hours which is relevant to the query. Our editorial experiment confirms that 91.7% of the queries in the testing set are recency-sensitive queries.

Table XII. Data Distribution in Sense of Relevance Grade and Recency Label

	Perfect	Excellent	Good	Fair	Bad
Regular	0.7%	17.0%	44.9%	26.6%	10.8%
Twitter	1.4%	35.5%	42.6%	16.9%	3.6%

(a) relevance grade (demoted)

	Perfect	Excellent	Good	Fair	Bad
Regular	1.1%	25.1%	45.8%	18.6%	9.4%
Twitter	1.4%	35.5%	42.6%	16.9%	3.6%

(b) relevance grade (non-demoted)

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

(c) recency label

We can also measure the quality of Twitter TinyURLs in aggregation by inspecting the freshness and relevance grades of our Twitter TinyURLs and regular URLs. We present the distribution of grades in Table XII. We observe that the quality of Twitter TinyURLs is better than regular URLs in sense of both relevance and recency. In Twitter TinyURLs, 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 TinyURLs. This confirms our assumption that the URLs extracted from Twitter data are generally very fresh. At same time, the overall relevance quality of Twitter TinyURLs is also higher than regular URLs. The percentages of perfect and excellent Twitter TinyURLs are higher than those of regular URLs, while the percentages of fair and bad Twitter TinyURLs are lower than those of regular URLs. This means Twitter TinyURLs are potentially useful to improve ranking for recency-sensitive queries.

6.7. Ranking Results

As shown in Table XIII, our proposed approach which blends Twitter content into the standard ranked list significantly improves ranking in the 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 TinyURLs from being promoted because Twitter TinyURLs suffer from feature impoverishment.

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 TinyURLs 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 TinyURLs, ($\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{content}}$) improves both relevance and recency metrics. If we enrich the representation of the Twitter TinyURLs, ($\mathcal{M}_{\text{regular}}, \mathcal{M}_{\text{twitter}}$) reaches the best

Table XIII. Ranking Results Comparison

	Top 1 Results					
	$NDCG_{demote,1}$		$NDCG_{nodemote,1}$		$NDCF_1$	
$(\mathcal{M}_{regular}, \mathcal{M}_{regular})$	0.5928	n/a	0.6163	n/a	0.4745	n/a
$(\mathcal{M}_{content}, \mathcal{M}_{content})$	0.5747	−3.1%	0.6164	+0.0%	0.5128	+8.1%
$(\mathcal{M}_{regular}, \mathcal{M}_{content})$	0.6034	+1.8%	0.6209	+0.7%	0.5204	+9.7%
$(\mathcal{M}_{regular}, \mathcal{M}_{twitter})$	**0.7158**	+20.7%	**0.6922**	+12.3%	**0.7372**	+55.4%
$(\mathcal{M}_{regular}, \mathcal{M}_{composite})$	0.7065	+19.2%	0.6885	+11.7%	0.6862	+44.6%
	Top 5 Results					
	$NDCG_{demote,1}$		$NDCG_{nodemote,1}$		$NDCF_1$	
$(\mathcal{M}_{regular}, \mathcal{M}_{regular})$	0.6734	n/a	0.6915	n/a	0.5185	n/a
$(\mathcal{M}_{content}, \mathcal{M}_{content})$	0.6582	−2.3%	0.6904	−0.2%	0.5870	+13.2%
$(\mathcal{M}_{regular}, \mathcal{M}_{content})$	0.6852	+1.8%	0.6974	+0.9%	0.5697	+9.9%
$(\mathcal{M}_{regular}, \mathcal{M}_{twitter})$	**0.7394**	+9.8%	0.7215	+4.3%	**0.7502**	+44.7%
$(\mathcal{M}_{regular}, \mathcal{M}_{composite})$	0.7380	+9.6%	**0.7222**	+4.4%	0.7286	+40.5%

All the improvements are statistically significant (p-value < 0.01).

Table XIV. An Example of Recency Ranking improvement

(a) Ranking result by baseline approach $(\mathcal{M}_{regular}, \mathcal{M}_{regular})$.				
rank (in (b))	URL	grade	fresh	from Twitter
1 (14)	http://en.wikipedia.org/wiki/Swine_flu	good	no	no
2 (13)	http://www.cdc.gov/swineflu/	excellent	no	no
3 (11)	http://www.wrestlingmuseum.com/pages/bios/halloffame/albanobio.html	good	no	no
(b) Ranking result by new approach $(\mathcal{M}_{regular}, \mathcal{M}_{Twitter})$.				
rank (in (a))	URL	grade	fresh	from Twitter
1 (14)	http://www.washingtonpost.com/wp-dyn/content/article/2009/10/14/	excellent	yes	yes
2 (11)	http://www.baltimoresun.com/health/swine-flu/bal-homicideflu1014,0,5398298.story	excellent	yes	yes
3 (9)	http://www.thenewamerican.com/index.php/usnews/health-care/2079-swine-flu-the-risks-and-efficacy-of-vaccines -swine-flu-death-in-us -more-news-latest- updates/	excellent	yes	no

The query is *who swine flu*, and the query issue time is during 23:00∼23:59 UTC on October 14th, 2009.

performance across all metrics. This means that we are able to successfully incorporate real-time Web content without hurting relevance, and actually relevance is improved.

Our experiments do not confirm that $(\mathcal{M}_{regular}, \mathcal{M}_{composite})$ 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 $\mathcal{M}_{twitter}$, a model built with much less training data. Table XIV qualitatively illustrates the behavior of our algorithms. Compared with the baseline result, our Twitter-based algorithm $(\mathcal{M}_{regular}, \mathcal{M}_{regular})$ 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 nondemotion grades are always equal.

Table XV. Feature Importance List for URL Ranking

Feature	Category	Importance rank
$\phi_{\text{attr-1}}$	TinyURL attribute (Section 5.3)	6
ϕ_{tweet}	Retweet (Section 5.4)	7
ϕ_{unit}	Authority (Section 5.1)	8
$\phi_{\text{attr-7}}$	TinyURL attribute (Section 5.3)	9
$\phi_{\text{attr-17}}$	TinyURL attribute (Section 5.3)	13

Only proposed features are listed.

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 are highly valued by the ranking model. As presented in Algorithm 1, the Twitter ranking function $\mathcal{M}_{\text{Twitter}}$ uses both content features and Twitter features. We can compute the importance of each feature by the method proposed in Friedman [2001]. We rank features by descending order of importance, and show the top five Twitter features in Table XV.

We observe that $\phi_{\text{attr-1}}$, $\phi_{\text{attr-7}}$, $\phi_{\text{attr-17}}$ play important roles to boost Twitter URLs, because these features represent the authority and activity of the users that are related to the Twitter TinyURLs from different aspects. Retweet user feature ϕ_{tweet} is also an important feature, which has the similar nature as the features in Twitter TinyURL attribute features category. Another useful Tweet-TinyURL textual feature is ϕ_{unit}, which is the unit match feature between query and tweet text as defined in Eq. (10). This means the text similarity between a query and a tweet in general highly correlates with the relevance between the query and the Twitter TinyURL posted in the tweet.

7. RANKING TWEETS

Tweet ranking is another important application for recency ranking: given a query, how to rank the most relevant and fresh tweet on the top position. To explore feature effectiveness in the tweet ranking problem, we adopt standard procedures of learning-to-rank approach (Section 4.1). We learn a ranking function with existing training data, while at query time, this ranking function is applied to all candidate tweets to compute each individual ranking score. Training data consists of a set of $\langle$query, Tweet, label$\rangle$ tuples, each of which has corresponding rank features and relevance label provided by human editors.

We ask human editors to label each tuple $\langle$query, Tweet$\rangle$ with a relevance grade. We still apply five judgment grades on each tuple: perfect, excellent, good, fair, and bad. For editors to judge the tuple, we ask them to grade it by freshness, intent, usefulness, content, and domain authority. For tweets with a TinyURL, we ask the editors to navigate to the TinyURL, and consider the additional relevance of the TinyURL for overall grade. For training set, we collect 40,531 query-tweet pairs with 3,800 unique queries; for testing set, there are 16,438 query-tweet pairs with 1,880 unique queries. All of the data is collected in January and February 2010. The sampled queries are from the class of recency-sensitive queries [Dong et al. 2010b] tagged by time.

For evaluation, we utilize the NDCG metric, which is described in Section 6.4. We train different ranking functions based on different ranking feature sets, so that we can compare and explore their utilities for the tweet ranking task. All the ranking functions are learned using the GBDT algorithm as introduced in Section 4.1 with 500 trees for each model, 18 nodes per tree, with a shrinkage parameter of 0.06.

Intuitively, text-matching features are the most straightforward features that capture relevance between query and tweet. Therefore, we adopt a set of text-matching features as the baseline feature set. More specifically, we use 10 features that simulate

Table XVI. Ranking Function NDCG Comparison Using Different Feature Set

Feature Set	$NDCG_5$	$\Delta NDCG_5$	$NDCG_1$	$\Delta NDCG_1$
F_0 (baseline)	0.820	n/a	0.763	n/a
$F_0 \cup F_1$	0.824	+0.47%	0.774	+1.43%
$F_0 \cup F_2$	0.829	+1.06%	**0.783**	**+2.71%**
$F_0 \cup F_3$	0.822	+0.31%	0.769	+0.80%
$F_0 \cup F_1 \cup F_2$	**0.831**	**+1.40%**	**0.782**	**+2.48%**
$F_0 \cup F_1 \cup F_2 \cup F_3$	**0.833**	**+1.64%**	**0.788**	**+3.26%**

Bold font refers to statistically significant improvements ((p-value $<$ 0.05)). F_0: *tf-idf* text-matching features (baseline feature set). F_1: retweet user features (Table V). F_2: retweet language model features (Table X). F_3: Twitter user social network feature (ϕ_{user_rank}) (Section 5.1).

Table XVII. Feature Importance List, for New Proposed Twitter Features

Feature	Category	Importance rank
ϕ_{lm_nort}	Retweet LM feature (Table X)	6
ϕ_{lm_div}	Retweet LM feature (Table X)	7
ϕ_{lm_rt}	Retweet LM feature (Table X)	8
ϕ_{lm_sub}	Retweet LM feature (Table X)	9
ϕ_{tweet}	Retweet user feature (Table V)	11
ϕ_{user_rank}	Twitter user social network feature (Section 5.1)	13
ϕ_{mean_rt}	Retweet user feature (Table V)	14
ϕ_{rt}	Retweet user feature (Table V)	15
ϕ_{sd_rt}	Retweet user feature (Table V)	19

tf-idf features [Manning et al. 2008] as the basic text-matching features. We then combine different categories of proposed features with the baseline *tf-idf* features to examine the utility of developed features.

Table XVI compares the ranking function performance based on different feature sets, on the test set. The value in each table slot is NDCG value and percentage of improvement on the testing dataset. The gains with the bond font indicate p-value smaller than 0.05. We observe that the class of retweet language model features improve ranking mostly, while retwitter user features are also quite effective. Both the classes of designed features are better than the Twitter user social network feature ϕ_{user_rank}, and the combination of all available features ($F_0 \cup F_1 \cup F_2 \cup F_3$) yields best ranking function.

Table XVII lists the feature importance, from which we observe that retweet language model features are the most informative features among the new feature proposed, while *tf-idf* baseline features are still the most important feature. This is consistent with the results in Table XVI.

To further evaluate the effectiveness of different retweet-related features, we conduct another simple experiment for retweet prediction. It is formulated as a binary classification problem to predict whether a newly generated tweet would be retweeted or not in the future, and we use GBDT for the experiments. For training, we generate all our candidate features based on historical retweet behavior from September 2009. For testing, we generate a sample of around 800,000 tweets from the first week of October 2009. The ROC plot in Figure 4, based on 10-fold cross-validation, makes some interesting observations. While the Twitter user features are highly predictive of retweets, they are less useful for the tweet ranking task. The Twitter LM features, on the other hand, show a reversed trend, that is, they are more useful for the tweet

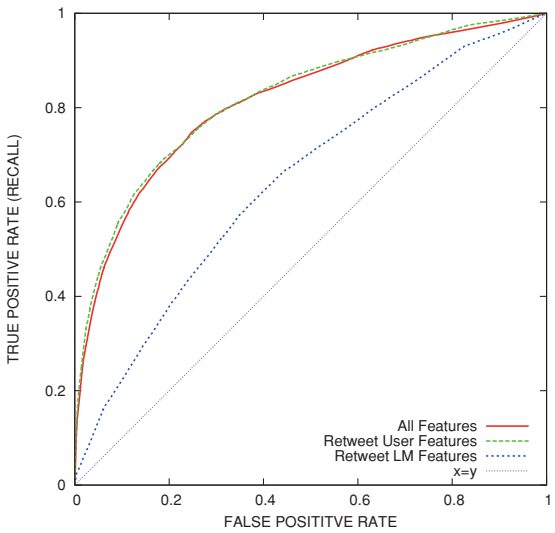

Fig. 4. Comparison of ROC AUC plots for retweet prediction using different features.

Table XVIII. An Example of Tweet Ranking Improvement for the Query *marbella, spain*

(a) Ranking result by baseline approach.		
rank (in (b))	tweet text	grade
1 (6)	RT @selecterkuley @RobboRanx ROBBO YOU GO HAAAAARD!WE A LISTEN A SOUTH SPAIN!MARBELLA DEYAH...	bad
2 (3)	http://bit.ly/bSiObp - A beautiful south facing duplex penthouse consisti - Marbella - Malaga - Spain - Apartments ...	fair
3 (1)	Spain property corruption scandal Marbella to end regional government approving a new plan. http://bit.ly/cr2Pjb	good
(b) Ranking result by new approach.		
rank (in (a))	tweet text	grade
1 (3)	Spain property corruption scandal Marbella to end regional government approving a new plan. http://bit.ly/cr2Pjb	good
2 (6)	http://bit.ly/bcagiF - Magnificent three bedroom penthouse in a luxury ga - Marbella - Malaga - Spain - Apartments ...	fair
3 (2)	http://bit.ly/bSiObp - A beautiful south facing duplex penthouse consisti - Marbella - Malaga - Spain - Apartments ...	fair

ranking task. One explanation is that the ranking improvements are mainly from those tweets issued by less known users.

Table XVIII illustrates an example, where the ranking function using all features is better the function only using the baseline features. This example illustrates how informal text is generally demoted in the improved rankings.

8. RELATED WORKS

8.1. Twitter-Ranking-Related Products

Most of the prior works on Twitter ranking handle the ranking of individual tweets. Although Twitter maintains a specialized search engine (http://search.twitter.com/), there exist several vertical search engines which index content across real-time data

(e.g., blogs, collaborative bookmarking sites)[3]. While these search engines are able to return very fresh documents, they often suffer from coverage or ranking issues. For example, considering a breaking-news query, the majority of related tweets consist of very brief comments on the news topic from random Twitter users. While such content may satisfy some search users, other users 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 might also hurt the user experience for those who are not familiar with Twitter. Besides ranking individual tweets, Bing Twitter Search[4] also provides search results for those URLs referred to by Twitter users. The contents of this vertical search engine 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.

8.2. 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 [Diaz 2009; Konig et al. 2009; Moon et al. 2010] . As we already mentioned in Section 1, 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[Dong et al. 2010a]. We extend this work by using Twitter to quickly update our index and generate new features.

In addition, Weng et al. [2010] propose an algorithm to identify influential users in Twitter services. They extend the PageRank algorithm by incorporating topical similarity between users into link structure. This is achieved based on the observation of homophily phenomenon in Twitter community. Duan et al. [2010] propose a ranking strategy which uses not only the content relevance of a Tweet, but also the account authority and tweet-specific features such as whether a TinyURL link is included in the tweet. The major finding is that whether a tweet contains a TinyURL or not, the length of the tweet and account authority are the best conjunction. Kulkarni et al. [2011] explore how queries, their associated documents, and the query intent change over time, and provide some features to improve search results. Dai and Davison [2010] propose a temporal Web link-based ranking scheme to incorporate features from historical author activities, and show improvements over PageRank in both relevance and freshness of the search results.

8.3. Twitter-Analysis-Related Research

Twitter as a research topic has been investigated by researchers in social network analysis. Java et al. [2007] study the topological and geographical properties of the 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. [2008] use Twitter data to confirm that users' attentions limit the number of people with whom they interact in a social network. Hughes et al. [Hughes and Palen 2009] examine Twitter usage as

[3]http://collecta.com/.
http://www.oneriot.com/.
http://www.yourversion.com/.
[4]http://bing.com/social.

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. [2009] investigate Twitter as a form of sharing consumer opinions concerning brands, and find the implications for corporations using micro-blogging as part of their overall marketing strategy. Krishnamurthy et al. [2008] 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.

In addition, Shamma et al. [2009] 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. Teevan et al. [2011] compare query log between Twitter search and Web search, and find Twitter results including more social chatter and social events while Web results contain more basic facts and navigational content. Zaman et al. [2010] train a probabilistic collaborative filter model to predict future retweets, and find that the most important features for prediction are the identity of the source of the tweet and retweeter. Ramage et al. [2010] present a labeled LDA model to map the content of Twitter to substance, style, status, and social characteristics of posts. Furthermore, Twitter has also been studied in the context of education [Borau et al. 2009; Dunlap and Lowenthal 2009], communication [Zhao and Rosson 2009], and collaboration [Honeycutt and Herring 2009].

9. CONCLUSION

In this article, we present extensive evidence to support the claim that Twitter data can be exploited to improve both web Ranking and tweet ranking for recency-sensitive queries. We demonstrated that both relevance-based and freshness-based metrics can be improved with our approaches.

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 real-time user behavior for those researchers who lack access to more sensitive log data.

In the future, we are interested in tweet spam detection and Twitter TinyURL spam detection, enriching the features extracted from Twitter using regular URL information and results from Section 8.3, and incorporating diversity. Furthermore, we are interested in synthesizing signals from Twitter streams with other sources of real-time evidence into a cohesive recency ranking module. Finally, if demographic information about Twitter users can be extracted or predicted, this resource can also be used for conducting personalization experiments.

REFERENCES

Agichtein, E., Brill, E., and Dumais, S. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Alonso, O., Carson, C., Gerster, D., Ji, X., and Nabar, S. U. 2010. Detecting uninteresting content in text streams. In *Proceedings of the SIGIR Workshop on Crowdsourcing for Search Evaluation.*

Bai, J., Chang, Y., Cui, H., Zheng, Z., Sun, G., and Li, X. 2008. Investigation of partial query proximity in web search. In *Proceedings of the 17th International Conference on World Wide Web.*

Bonacich, P. 1972. Factoring and weighting approaches to clique identification. *J. Math. Sociol. 2,* 113–120.

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

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

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. 2000. Graph structure in the web. In *Proceedings of the 9th International World Wide Web Conference on Computer Networks.* 309–320.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. 2005. Learning to rank using gradient descent. In *Proceeding of the International Conference on Machine Learning.*

Cao, Z., Qin, T., Liu, T., Tsai, M., and Li, H. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning.*

Chapelle, O. and Chang, Y. 2011. Yahoo! learning to rank challenge overview. *J. Mach. Learn. Res., Proc. Track 14,* 1–24.

Dai, N. and Davison, B. D. 2010. Freshness matters: In flowers, food, and web authority. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Diaz, F. 2009. Integration of news content into web results. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining (WSDM).* 182–191.

Dong, A., Chang, Y., Zheng, Z., Mishne, G., Bai, J., Zhang, R., Buchner, K., Liao, C., and Diaz, F. 2010a. Towards recency ranking in web search. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM).*

Dong, A., Zhang, R., Kolari, P., Bai, J., Diaz, F., Chang, Y., Zheng, Z., and Zha, H. 2010b. Time is of the essence: Improving recency ranking using twitter data. In *Proceedings of the 19th International Conference on World Wide Web.*

Duan, Y., Jiang, L., Qin, T., Zhou, M., and Shum, H. 2010. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics.*

Dunlap, J. C. and Lowenthal, P. R. 2009. Tweeting the night away: Using twitter to enhance social presence. *J. Inf. Syst. Educ.*

Dunning, T. 1993. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist. 19,* 1, 61–74.

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

Friedman, J. H. 2001. Greedy function approximation: A gradient boosting machine. *Ann. Statist. 29*, 5, 1189–1232.

Honeycutt, C. and Herring, S. C. 2009. Beyond microblogging: Conversation and collaboration via twitter. In *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS'09).* 1–10.

Huberman, B. A., Romero, D. M., and Wu, F. 2008. Social networks that matter: Twitter under the microscope. http://arxiv.org/pdf/0812.1045.pdf

Hughes, A. L. and Palen, L. 2009. 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.*

Jansen, B. J., Zhang, M., Sobel, K., and Chowdury, A. 2009. Twitter power: Tweets as electronic word of mouth. *J. Amer. Soc. Inf. Science Technol. 1.* 20.

Jarvelin, K. and Kekalainen, J. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst. 20*, 422–446.

Java, A., Song, X., Finin, T., and Tseng, B. 2007. Why we twitter: Understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD Workshop on Web Mining and Social Network Analysis (WebKDD/SNA-KDD '07).* ACM, New York, 56–65.

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

Konig, A. C., Gamon, M., and Wu, Q. 2009. Click-through prediction for news queries. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Informational Retrieval.* 347–354.

Krishnamurthy, B., Gill, P., and Arlitt, M. 2008. A few chirps about twitter. In *Proceedings of the 1st Workshop on Online Social Networks (WOSP'08).* 19–24.

Kulkarni, A., Teevan, J., Svore, K. M., and Dumais, S. T. 2011. Understanding temporal query dynamics. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM).*

Lafferty, J., McCallum, A., and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning.* 282–289.

Manning, C., Raghavan, P., and Schutze, H. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

METZLER, D., DUMAIS, S. T., AND MEEK, C. 2007. Similarity measures for short segments of text. In *Proceedings of the 29th European Conference on IR Research.* 16–27.

MISHNE, G. A. 2007. Applied text analytics for blogs. Ph.D. thesis, University of Amsterdam, Amsterdam.

MOON, T., LI, L., CHU, W., LIAO, C., ZHENG, Z., AND CHANG, Y. 2010. Online learning for recency search ranking using real-time user feedback. In *Proceedings of the 19th International Conference on Information and Knowledge Management.*

PONTE, J. M. AND CROFT, W. B. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York, 275–281.

RAMAGE, D., DUMAIS, S., AND LIEBLING, D. 2010. Characterizing microblogs with topic models. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM).*

SHAMMA, D., KENNEDY, L., AND CHURCHILL, E. 2009. Tweet the debates: Understanding commu- nity annotation of uncollected sources. In *Proceedings of the ACM International Conference on Multimedia.*

STOLCKE, A. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing.*

TEEVAN, J., RAMAGE, D., AND MORRIS, M. R. 2011. Twittersearch, a comparison of microblog search and web search. In *Proceedings of International conference on Web Search and Data Mining (WSDM).*

WENG, J., LIM, E., JIANG, J., , AND HE., Q. 2010. Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM).*

ZAMAN, T. R., HERBRICH, R., VAN GAEL, J., AND STERN, D. 2010. Predicting information spreading in twitter. In *NIPS Workshop on Computational Social Science and the Wisdom of Crowds.*

ZHAI, C. AND LAFFERTY, J. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst. 22,* 2, 179–214.

ZHAO, D. AND ROSSON, M. B. 2009. How and why people twitter: The role that micro-blogging plays in informal communication at work. In *Proceedings of the ACM International Conference on Supporting Group Work.* ACM, New York, 243–252.

ZHENG, Z., ZHA, H., CHEN, K., AND SUN., G. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Received June 2011; revised December 2011; accepted May 2012