The Impacts of Structural Difference and Temporality of Tweets on Retrieval Effectiveness

LIFENG JIA and CLEMENT YU, University of Illinois at Chicago
WEIYI MENG, Binghamton University

To explore the information seeking behaviors in microblogosphere, the microblog track at TREC 2011 introduced a real-time ad-hoc retrieval task that aims at ranking relevant tweets in reverse-chronological order. We study this problem via a two-phase approach: 1) retrieving tweets in an ad-hoc way; 2) utilizing the temporal information of tweets to enhance the retrieval effectiveness of tweets. Tweets can be categorized into two types. One type consists of short messages not containing any URL of a Web page. The other type has at least one URL of a Web page in addition to a short message. These two types of tweets have different structures. In the first phase, to address the structural difference of tweets, we propose a method to rank tweets using the divide-and-conquer strategy. Specifically, we first rank the two types of tweets separately. This produces two rankings, one for each type. Then we merge these two rankings of tweets into one ranking. In the second phase, we first categorize queries into several types by exploring the temporal distributions of their top-retrieved tweets from the first phase; then we calculate the time-related relevance scores of tweets according to the classified types of queries; finally we combine the time scores with the IR scores from the first phase to produce a ranking of tweets. Experimental results achieved by using the TREC 2011 and TREC 2012 queries over the TREC Tweets2011 collection show that: (i) our way of ranking the two types of tweets separately and then merging them together yields better retrieval effectiveness than ranking them simultaneously; (ii) our way of incorporating temporal information into the retrieval process yields further improvements, and (iii) our method compares favorably with state-of-the-art methods in retrieval effectiveness.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.4.m [**Information Systems Applications**]: Miscellaneous

General Terms: Experimentation, Performance

Additional Key Words and Phrases: Ad-hoc retrieval of tweets, learning to rank, query temporal categorization

ACM Reference Format:
Jia, L., Yu, C., and Meng, W. 2013. The impacts of structural difference and temporality of tweets on retrieval effectiveness. *ACM Trans. Inf. Syst.* 31, 4, Article 21 (November 2013), 38 pages.
DOI:http://dx.doi.org/10.1145/2500751

1. INTRODUCTION

Twitter, a worldwide popular microblog service, has a daily volume of over 340 million tweets,[1] which motivates research interests in studying the information seeking behaviors within microblogosphere. The microblog track at TREC 2011 introduced a real-time ad-hoc retrieval task, whereby a user wishes to see the most recent and

[1] `http://en.wikipedia.org/wiki/Twitter`

Authors' addresses: L. Jia and C. Yu, Computer Science Department, University of Illinois at Chicago, IL; email: ljia2@uic.edu; W. Meng, Computer Science Department, Binghamton University.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2013 ACM 1046-8188/2013/11-ART21 $15.00
DOI:http://dx.doi.org/10.1145/2500751

relevant information to a query within Twitter [Ounis et al. 2011]. To respond to a query with a timestamp t, the retrieved tweets should satisfy the following three conditions: (1) relevant to the query, (2) published on or before time t, and (3) ranked in reverse-chronological order of their publishing times.

Some studies have been done in information retrieval of tweets. These studies can be categorized into two major classes. The techniques in the first class [Choi et al. 2012; Duan et al. 2010; Han et al. 2012; Metzler and Cai 2011; Zhang et al. 2012] rank tweets by measuring the lexical similarities between tweets and queries. The methods in the second class [Amati et al. 2012; Choi and Croft 2012; Dong et al. 2010b; Efron and Golovchinsky 2011] rank tweets by exploring temporal information (the publishing times of tweets and the timestamps of queries). Some studies [Efron et al. 2012; Liang et al. 2012; Massoudi et al. 2011] employ both lexical similarity and temporality in ranking tweets. However, there are two important issues that are not well addressed by these existing works.

The first issue is the impact of the structural difference of tweets on retrieval effectiveness. Specifically, there are two types of tweets that have different structures. The first type (to be defined as T-tweet in Section 3.2) is just a short text message with no more than 140 characters. The second type (to be defined as TU-tweet in Section 3.2) contains at least one URL of a Web page in addition to a short text message. All existing studies simultaneously rank both types of tweets. However, we believe it is important to utilize the structural difference of tweets in retrieval. Let us illustrate the motivation by the following example.

Example 1. Consider a query q = "*phone hacking British politicians*", a tweet d_1 = "*@jamesrae andy Gray is suing the NOTW... just got fired from Sky for footage that should never have been seen. I smell Murdoch!*", a second tweet d_2 = "*Tensions simmer as 'frustrated' Rupert Murdoch flies in to face phone-hacking affair http://t.co/b3kOppY via @guardian*" and a third tweet d_3 = "*Windows Phone 7 gets USB Tethering Hack http://tinyurl.com/4lafss6*". d_1 is a T-tweet that only has a short message. d_1 is relevant to q but has no query terms. d_2 and d_3 are two TU-tweets. Each of them has not only a message but also a URL. d_2 is relevant to q. It contains two query terms "*phone*" and "*hacking*" in its message and all four query terms in the web page of the URL in d_2. d_3 is irrelevant to q. It contains two query terms "*Phone*" and "*hack*" in its message. The Web page of the URL in d_3 has no query terms. The content of a TU-tweet is the union of its short message and the contents of the Web pages of the URLs in it. It is intuitive that for a TU-tweet, the higher the percentage of query terms appearing in it is, the more likely the tweet is relevant. The relevant d_2 has more query terms than the irrelevant d_3. However, such an intuition does not apply for a T-tweet. d_1 has no query terms but it is relevant to q. This is because T-tweets are so short that some relevant T-tweets may not have any query terms. In addition, we find out that (see Section 6.1.2) the sets of the most important features for learning to rank the two types of tweets are very different.

Motivated by such an observation, we propose to use the divide-and-conquer strategy to address the structural difference of tweets. Specifically, we learn two rankers that are dedicated to ranking T-tweets and TU-tweets separately. This produces two tweet type-specific rankers. We then learn a classifier that determines a preference between any T-tweet and any TU-tweet with respect to a given query. The details about these two tweet type-specific rankers and the classifier are discussed in Sections 3.2 and 3.3, respectively. Given a query q, we first obtain a ranking of T-tweets, R_1, and a ranking of TU-tweets, R_2, by using the two type-specific rankers, respectively. Then we apply the classifier to determine the preference between each T-tweet from R_1 and each TU-tweet from R_2. Finally, we merge the tweets from R_1 and R_2 into a single

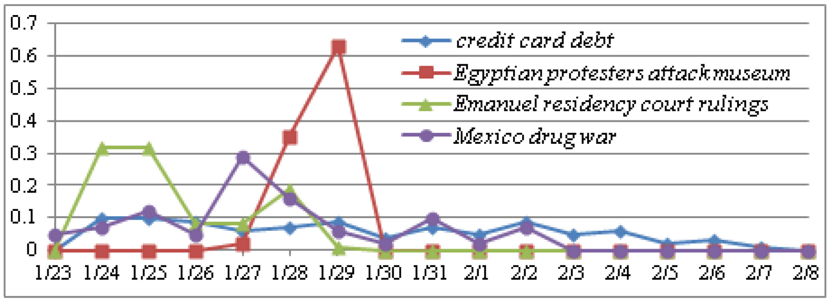


Fig. 1. The distributions of relevant tweets over time.

ranking. The merging process considers the preferences from the two rankers and the classifier. The discussion of how to merge two rankings of tweets is presented in Section 3.4.

The second issue is the impact of the temporal sensitivities of queries on the retrieval effectiveness of tweets. Queries can be categorized into time sensitive and time insensitive types [Dakka et al. 2012; Jones and Diaz 2007]. For ease of presentation, Figure 1 shows the temporal distributions of the relevant tweets with respect to four sample TREC queries. These distributions are plotted over the period from 1/23/2011 to 2/8/2011, when the TREC Tweets2011 collection was sampled from Twitter. The x-axis represents time in the unit of day [Efron and Golovchinsky 2011]. The y-axis represents the percentage of relevant tweets published on a particular day. By observing these distributions, we claim that there are three types of queries. The first type is insensitive to time, while the last two types are time sensitive.

— The first type of queries has a relatively flat (uniform) distribution of relevant tweets over time, indicating that these queries are insensitive to time. This is exemplified by the query "*credit card debt*."
— The second type of queries has a dominant peak in terms of their temporal distributions of relevant tweets. The dominant peak contains an extremely large portion of relevant tweets concentrating on a single day. This is exemplified by the query "*Egyptian protesters attack museum*." The attack happened during the night of 1/28/2011 and a dominant peak in the distribution is formed on 1/29/2011. An event related to the topic of such a query is usually the event of a breaking news story. The relevant tweets are so concentrated around the peak that the percentage of relevant tweets rapidly decreases beyond the peak. In this article, such queries are called *dominant peak queries*.
— The third type of queries has one or more nondominant peaks. Each peak contains a significant portion of relevant tweets on a day but the percentage of relevant tweets of a nondominant peak is not as high as that of a dominant peak. A nondominant peak of a query is caused by an event that is related to the query. These related events trigger people's intensive discussions about the query topic at different times. This is exemplified by two queries: "*Mexico drug war*" and "*Emanuel residency court rulings*." For "*Mexico drug war*," the nondominant peak on 1/27/2011 is caused by a related event, "*Pot-firing catapult found at Arizona-Mexico border*". For "*Emanuel residency court rulings*," the first two nondominant peaks on 1/24/2011 and 1/25/2011 correspond to the event: "*Illinois Court Throws Emanuel Off Chicago Mayoral Ballot*"; the third peak on 1/28/2011 corresponds to another related event: "*Illinois Supreme Court keeps Emanuel on ballot*." In this article, such queries are called *nondominant peak queries*.

These three types of queries depend on their temporal distributions of relevant tweets. In practice, it is unrealistic to know such distributions for given queries. In Efron and Golovchinsky [2011] and Jones and Diaz [2007], the temporal distribution of the relevant tweets with respect to a query q can be approximated by that of the top tweets with respect to q. These top tweets can be retrieved by a ranking model, such as BM25 [Robertson et al. 1996]. In this article, we classify queries into different types by the temporal distributions of their top tweets. For time-insensitive queries, there is no need to employ temporal information; for time sensitive queries, we propose two different techniques to calculate the temporal relevance of tweets to dominant peak queries and to nondominant peak queries, respectively. The degree of temporal relevance is measured by a time-related relevance score (to be given in Section 4.2). Our proposed method for categorizing queries and for computing the time-related relevance scores with respect to the two types of time sensitive queries are presented in Section 4.2. In this article, we only study these three types of queries. The studies of other types of queries, such as cyclic queries (e.g., "*Halloween*") are deferred to future work.

Our work has two novelties: 1) ranking the two types of tweets by a divide-and-conquer manner can improve retrieval effectiveness; and 2) our temporal classification of queries and two different ways of computing the time-related relevance scores with respect to the two different types of time sensitive queries are different from existing works. We now summarize the research questions we aim to answer in this article.

— Acknowledging that tweets can be classified into the two types by their different structures, is the retrieval effectiveness of tweets affected by their structural difference?
— How to leverage the structural difference of tweets to enhance their retrieval effectiveness?
— What are the effectiveness and the efficiency of the proposed algorithm?
— How can we improve retrieval effectiveness by taking into consideration the temporal information (publishing times) of tweets?
— How does our method perform compared to various state-of-the-art methods?

This article has the following contributions.

— We investigate the impact of the structural difference of tweets on retrieval effectiveness.
— We present a novel algorithm of ranking tweets by using the divide-and-conquer strategy. To our knowledge, our work is the first study that leverages the structural difference of tweets to enhance retrieval effectiveness.
— We present a novel categorization of queries by their sensitivities to time.
— We propose different techniques to calculate the degrees of temporal relevance of tweets with respect to the different categories of queries.

The remainder of this article is organized as follows. We review the related works in Section 2. Section 3 introduces our divide-and-conquer method for ranking tweets. Section 4 discusses our method for categorizing queries in terms of their temporal sensitivities and proposes different techniques to calculate the temporal relevance of tweets. Experimental setup and experimental results are provided in Section 5 and Section 6, respectively. The article is concluded in Section 7.

2. RELATED WORK

Recently, interests are rising in exploring Twitter for information retrieval of tweets by different criteria, such as lexical relevance [Choi et al. 2012; Duan et al. 2010; Han et al. 2012; Metzler and Cai 2011; Zhang et al. 2012], temporal relevance [Amati et al.

2012; Choi and Croft 2012; Dong et al. 2010b; Efron and Golovchinsky 2011] and jointly lexical and temporal relevance [Efron et al. 2012; Liang et al. 2012; Massoudi et al. 2011]. Beyond tweet retrieval, some studies [Amodeo et al. 2011; Dakka et al. 2012; Dong et al. 2010a; Jones and Diaz 2007; Keikha et al. 2011a, 2011b; Li and Croft 2003] also showed that incorporating the publishing times of documents into the retrieval process is beneficial for ad-hoc retrieval. Instead of using the publishing times of documents, some works [Berberich et al. 2010; Dai and Davison 2010; Elsas and Dumais 2010; Kulkarni et al. 2011] studied how to improve the ranking effectiveness by using the temporal information extracted from the contents of documents. Moreover, our study is also related to some works [Ailon et al. 2008; Bian et al. 2010; Dai et al. 2011; Hüllermeier and Fürnkranz 2010] in learning to rank. In the rest of this section, we review in greater detail the related works.

2.1. Lexical Relevance-Based Retrieval

The first thread of related works studied tweet retrieval by measuring their lexical similarities to queries. Duan et al. [2010] employed RankSVM [Herbrich et al. 2000; Joachims 2002] to rank tweets by their lexical relevance to queries. Metzler and Cai [2011] studied the real-time ad-hoc tweet retrieval problem by using RankSVM to rank tweets with respect to queries and rearranged the top-ranked tweets in reverse-chronological order. This work achieved the best results reported in TREC 2011. Choi et al. [2012] showed that the quality of tweets is correlated with their relevance and applied the quality features in relevance ranking. They assumed that high quality tweets are more likely to be retweeted than low quality ones and learned a model to estimate the probability of a tweet being retweeted by exploring its lexical content. Zhang et al. [2012] proposed a query-specific model to rank tweets by considering the characteristics unique to a query. Specifically, given a query q, they treated the top and the bottom tweets retrieved by a ranking model as positive and negative examples and then learned a ranking model specific to q. Efron et al. [2012] expanded each tweet d with respect to a query q as follows. The terms of the most similar tweets to d are added to d. The query q is then compared with the expanded tweets for the similarity computation, in order to improve retrieval effectiveness. Han et al. [2012] expanded each tweet d in a similar manner by the terms from other tweets that are lexically similar to d. Our work has two fundamental differences from the works reviewed earlier: 1) we consider the structural difference of the two types of tweets in the retrieval process while they ranked both types of tweets together; and 2) they only measured the lexical similarities of tweets to queries while we take into consideration both lexical similarities and temporal information.

2.2. Temporal Relevance-Based Retrieval

The second thread of related works studied the impact of temporal information on retrieval effectiveness. Dong et al. [2010a, 2010b] proposed the recency ranking problem and studied the problem using Twitter data. Amati et al. [2012] assumed that the recent tweets with respect to (the timestamp of) a query q are more likely to be relevant than the old tweets. Massoudi et al. [2011] studied a query expansion method where the expanded query terms are selected from high-quality and recent tweets, instead of low-quality and old tweets. The quality of tweets can be estimated by some indicators, such as the number of followers of Twitter users. All the works we have mentioned in principle prefer recent tweets (or terms from recent tweets) to old ones. However, this is not always desirable. For example, in Figure 1, for the query "*Mexico drug war*," a significant portion of relevant tweets are published on 1/27/2011 and some relevant tweets are published on 2/2/2011. The tweets on 1/27/2011 are as relevant as those tweets on 2/2/2011. They should not be assigned lower priorities in retrieval. Our work classifies queries by the temporal distributions of their top tweets and then proposes different ways of utilizing temporal information of tweets according to the classified types of queries. Liang et al. [2012] studied the real-time ad-hoc tweet retrieval by a two-phase approach where 1) an ad-hoc retrieval of tweets is conducted and 2) tweets are re-ranked to promote the relevant and recent ones. Our two-phase method is different from theirs in two aspects. First, they ranked both types of tweets simultaneously while we leverage the structural difference of tweets. Second, they promoted recent tweets over old tweets while we classify queries by their time sensitivities before applying temporal information in different manners according to the classified types of queries. Choi and Croft [2012] obtained the top tweets (consisting of retweets and non-retweets) with respect to a query q from a ranking model. Then they explored the temporal distribution of the top retweets to measure the importance of each day with respect to q. The importance of a day t to q is proportional to the number of the top retweets published on t. Finally, they arranged non-retweets by considering the importance of each of their publishing days. Our work differs from theirs in that they use retweets to measure the importance of days while we use top tweets to determine the importance of days. Moreover, our calculation of the degrees of relevance between tweets and queries by temporality is quite different from theirs. Efron et al. [2012] obtained the top tweets with respect to a query q and then, for each tweet d, acquired the most similar (top) tweets to d. They calculated the temporal similarity between q and d based on the temporal distribution of q's top tweets and that of d's top tweets. Our work differs from their work in that we classify queries based on the temporal distributions of their top tweets and then calculate the temporal relevance of tweets to queries by their classified types.

Besides Twitter search, Li and Croft [2003] studied time sensitive queries and assumed that relevant documents are mostly recent documents. They proposed an exponential-based age penalty strategy where aged documents are penalized and then demoted to boost the ranking positions of recent documents. Efron and Golovchinsky [2011] studied the same problem and proposed a query-specific exponential-based age penalty method where aged documents are penalized differently with respect to different queries. Our classification, determination and handling of time sensitive queries are different from the given works. Moreover, their hypothesis [Efron and Golovchinsky 2011; Li and Croft 2003] that aged documents should be penalized more than recent documents is not necessarily true for some time sensitive queries. For example, in Figure 1, for the query "*Mexico drug war*", the relevant tweets on 1/27/2011 should not be penalized relative to those on 2/2/2011. Amodeo et al. [2011] and Keikha et al. [2011b] presented temporal query expansions by using the terms selected from the top (blog) documents (with respect to a query q) that are published on the days that are most relevant to q. The relevance of a day t to q is measured by the average similarity of the top documents published on t to q in Keikha et al. [2011b] or by the percentage of q's top documents published on t [Amodeo et al. 2011]. We do not use temporal information in query expansion. Keikha et al. [2011a] showed that blog feed retrieval can benefit from the usage of temporal information. They studied the retrieval of blog feeds. A blog feed consists of a set of blog documents published on different days. We study the retrieval of individual tweets. Although both studies use temporal information, the utilizations of temporality in these two studies are very different. Dakka et al. [2012] indicated that, for a time sensitive query q, a document d can be represented by two dimensions: the lexical content c_d and the publishing time t_d. They assumed the independence between c_d and t_d. Our work differs from theirs in that we assume the contents of documents (tweets) and their publishing times are not necessarily independent. For example, for the query "*Emanuel residency court rulings*," the relevant tweets published on 1/24/2011 and 1/25/2011 discuss the event "*Illinois Court*

Throws Emanuel Off Chicago Mayoral Ballot" while the relevant tweets published on 1/28/2011 discuss the event "*Illinois Supreme Court keeps Emanuel on ballot*." The contents of tweets with respect to a query can be influenced by other related events which happen at different times. Jones and Diaz [2007] categorized queries into time insensitive ones, temporally ambiguous queries such as "*Iraq War*" (referencing two different wars) and temporally unambiguous queries such as "*Turkish earthquake 1999*". Our work categorizes queries by their sensitivities to time, instead of their temporal ambiguities.

Exploring the temporal information from the contents of documents can improve retrieval effectiveness too. Berberich et al. [2010] proposed a language model supplemented with a temporal dimension where the temporal information from a query and that from documents are uniformly expressed and matched in retrieval. For example, the query, "*World Cups in 1990s*" should be matched by the documents containing "*1998 World Cup*," because "*1990s*" temporally covers "*1998*." Elsas and Dumais [2010] studied the relationship between the temporal dynamics of document contents and the relevance of documents. For example, they showed that the contents of the relevant documents for navigational queries, such as "*YouTube*," have great and frequent changes over time. Kulkarni et al. [2011] discussed the interaction among the temporal changes of query popularity, the temporal changes of document contents and query intents. Dai and Davison [2010] utilized the freshness of Web site contents for computing Web site authority by examining the frequency of Web site content changes and that of Web site hyperlink changes over time. Our work uses the publishing times of top documents (tweets) to improve retrieval effectiveness.

2.3. Learning to Rank

Our work is also related to some studies in learning to rank. Bian et al. [2010] provided a divide-and-conquer framework for learning to rank documents. Dai et al. [2011] extended the same divide-and-conquer framework for learning to rank documents by freshness and relevance simultaneously. Our work has a fundamental difference from theirs. Both works [Bian et al. 2010; Dai et al. 2011] divided (clustered) queries into different clusters where queries within a cluster have a similar set of important learning to rank features. However, we divide (partition) documents (tweets) into two sets by considering their structural difference. Given some different rankings of a same set of documents that yield inconsistencies, Ailon et al. [2008] studied how to obtain a ranking of the same set of documents that approximately minimizes the disagreement with the given rankings. In our work, we merge two rankings of two different sets of tweets, one for T-tweets and the other for TU-tweets. Hüllermeier and Fürnkranz [2010] studied the problem where each example (document) is assigned the probabilities of belonging to different classes. No ranking of examples (documents) is discussed in Hüllermeier and Fürnkranz [2010].

3. A DIVIDE-AND-CONQUER METHOD FOR RANKING TWEETS

In this section, we introduce a novel method for ranking tweets. This method explores the structural difference of tweets by the divide-and-conquer strategy. It is deployed as the first phase to produce a ranking of tweets, taking into consideration their lexical similarities to queries only.

3.1. Method Overview

In this method, we differentiate the following two types of tweets: the first type is a short plain message without URLs (T-tweet) and the second type is a message containing at least one URL (TU-tweet). A URL usually leads to a Web page with a substantially more content than a short message. To explore such a structural difference, we propose to rank these two types of tweets separately and then merge the two type-specific rankings of tweets into a single ranking. The proposed method has two tweet type-specific rankers and a classifier. The two type-specific rankers are dedicated to ranking T-tweets and TU-tweets. The classifier calculates the preference between any T-tweet and any TU-tweet with respect to a query.

In this article, we resort to the learning to rank algorithms to produce the two rankers. Specifically, RankSVM [Herbrich et al. 2000; Joachims 2002] is employed. It can consider not only various lexical similarities between queries and tweets, such as BM25 similarity [Robertson et al. 1996], but also some special social network characteristics that are independent of queries, such as the number of retweets of tweets. It leverages different criteria as features to learn the two type-specific rankers. We denote as *T-tweet Ranker* the RankSVM model that is dedicated to ranking T-tweets. It is learned over the training data consisting of a set of training queries Q and a set of labeled T-tweets with respect to Q. Let *TU-tweet Ranker* denote the RankSVM model that is TU-tweet oriented. It is learned over the training data consisting of the same set of training queries Q but a different set of labeled TU-tweets with respect to Q. A classifier is learned to determine a preference between each T-tweet and each TU-tweet. Specifically, it is learned by using the union of the two sets of labeled tweets with respect to the same training query set Q. The classifier indicates for each T-tweet d_1 and each TU-tweet d_2 whether d_1 is preferred over d_2 or vice versa.

The goal of this method is to produce the ranking of tweets for a set of test queries, $Q' = \{q'_1, q'_2, \ldots, q'_m\}$. For each test query q'_i, we apply the *T-tweet Ranker* to obtain a ranking of T-tweets R_1. Then we obtain a ranking of TU-tweets R_2 by the *TU-tweet Ranker*. For each pair of one T-tweet from R_1 and one TU-tweet from R_2, the classifier is employed to determine a preference relationship between them with respect to q'_i. There are three sets of preferences: 1) the preference between any two T-tweets which is indicated by their relative ranking positions in R_1; 2) the preference of any two TU-tweets from R_2; and 3) the preference between any T-tweet from R_1 and any TU-tweet from R_2 indicated by the classifier. Finally, the two rankings, R_1 and R_2, are merged into a ranking by considering all three sets of preferences.

Because these three sets of preferences are computed by three different models, there may be inconsistent preferences. For example, given two T-tweets d_i and d_j and a TU-tweet d_k, the *T-tweet Ranker* may indicate $d_i \succ d_j$, which denotes the preference of d_i over d_j. However, the classifier may indicate $d_k \succ d_i$ and $d_j \succ d_k$. In such a circular preference situation, no matter how these three tweets are ranked in the merged ranking, there is at least one inconsistency. Suppose that the degree of the preference of d_i over d_j is 0.5, that of d_j over d_k is 0.4, that of d_k over d_i is 0.3, and there are no other preferences. If we determine that d_i is ranked above d_j which is ranked above d_k, it will incur an inconsistency with the degree of 0.3. This is the smallest amount of inconsistency among all possible orderings of these three tweets. In an ideal situation, we want to merge the two type-specific rankings into an optimal ranking that agrees best with the three sets of preferences. However, such a problem is NP-complete [Cohen et al. 1998]. Therefore, we propose a greedy merging algorithm called *GreedyMerging*. This algorithm always picks the tweet to be ahead of the remaining tweets, if it incurs the least amount of inconsistency relative to any of the remaining tweets. If there is no inconsistency among the three sets of preferences, the algorithm will produce the optimal merged ranking consistent with all preferences.

3.2. Tweet Type-Specific Rankers

In this section, we present the two rankers: one ranks T-tweets while the other ranks TU-tweets. For ease of introduction, we first define T-tweets and TU-tweets.

Definition 3.1 (*T-Tweet*). A T-tweet is a tweet whose message body has no URLs. The structure of a T-tweet consists of only one field:
a) Tweet Message Field: the message body of the tweet.

Definition 3.2 (*TU-Tweet*). A TU-tweet is a tweet whose message body has at least one URL. A tweet whose message body has URLs only is very rare. The structure of a TU-tweet consists of three fields:

a) Tweet Message Field: the message body with the exclusion of the embedded URLs.
b) URL Title Field: the union of the titles of the Web pages of the embedded URLs.
c) URL Body Field: the union of the bodies of the Web pages of the embedded URLs.

In a learning problem, the features are essential. Table I presents all the features for learning to rank tweets. Some features in Table I are explained in detail in the following. For T-tweets, the applicable features are computed based on their tweet message fields, whereas for TU-tweets, they are computed based on their three fields as well as the union of the three fields. For example, the BM25 similarity between a query and a T-tweet d can be computed based on the tweet message field of d; for a TU-tweet, four BM25 similarities can be computed, one based on the tweet message field, one based on the URL title field, one based on the URL body field and the last one based on the union of these three fields. Different degrees of significance can be associated with the different fields by the learning model. It has been shown that improvement in ranking can be achieved by weighting the fields of documents (for example, the titles of documents vs. the bodies of documents) differently [Robertson et al. 2004]. In our opinion, the same can apply to the tweets. Thus, we propose the features whose calculations are based on the different fields of tweets together with queries. During the establishment of the rankers, different weights are learned for those different field-based features.

Moreover, the features can be categorized into two types: tweet-related (*TR* for short) and query-tweet-related (*QTR* for short). The former type is calculated purely based on the tweets themselves. For example, for feature F_{13}, it is a Boolean feature indicating whether the tweet has at least an embedded URL. Studies [Duan et al. 2010; McCreadie et al. 2011; Metzler and Cai 2011] showed that whether a tweet has a URL is an effective feature for ranking tweets. Intuitively, the Web pages of the URLs embedded in tweets often provide more information than tweets' 140 characters. Thus, a tweet with embedded URLs has a higher probability of being relevant than a tweet without embedded URLs [Duan et al. 2010].

Besides the tweet-related features, the query-tweet-related features are also used to calculate different lexical similarities between queries and tweets. In addition to capturing term similarities, such as BM25 similarities discussed before, our method also computes concept similarities as features. A concept is a proper noun (*PN*), a dictionary phrase (*DP*), a simple noun phrase (*SNP*), or a complex noun phrase (*CNP*). A dictionary phrase is a noun phrase that can be looked up in dictionaries such as Wikipedia but is not a proper noun. A simple noun phrase (complex noun phrase) consists of two (more than two) nonstop terms but is neither a proper noun nor a dictionary phrase. A concept is recognized in a document if all of its nonstop terms appear in the document within a text window of certain size, with the smallest window size for *PN*s, then a bigger window size for *DP*s, an even bigger window size for *SNP*s, and the largest window size for *CNP*s. Please refer to the papers [Liu et al. 2004; Zhang et al. 2007] for the details about these concepts. In this article, we adopt the phrase recognition tool [Zhang et al. 2007] to identify the four types of concepts from queries and tweets. This tool can achieve an accuracy of 92% in recognizing concepts.

Table I. Features for Ranking Tweets

ID	Type	Feature Description (q = query, T = tweet).	No.
F_1	*QTR*	The percentage of the terms of q contained by the hashtags of T. The hashtags are the keywords or topics of T and they appear in the tweet message field of T by prefixing the symbol "#".	1
F_2	*QTR*	The percentage of the expansion terms of q contained by the hashtags of T. The expansion terms are obtained by the pseudo relevance feedback method [Liu et al. 2004].	1
F_3	*QTR*	Whether the four fields (the three fields of a TU-tweet and their union) contain q as an *SNP* or *CNP* respectively.	4
F_4	*QTR*	The frequency of q in T as an *SNP* or *CNP*.	1
F_5	*QTR*	Whether the four fields contain a key term of q, if exist. The key term is the nonverb term in q, satisfying the following two conditions: 1) it has the least document frequency among all query terms; 2) it is not a term in a *PN* or a *DP* concept.	4
F_6	*TR*	The length of the tweet message field of T. [Duan et al. 2010; McCreadie et al. 2011; Metzler and Cai 2011]	1
F_7	*QTR*	Whether the four fields contain all *PN* or *DP* query concepts.	4
F_8	*QTR*	The sum of the frequencies of all *PN* or *DP* query concepts in T.	1
F_9	*QTR*	The percentage of the nonverb terms of q contained in the four fields.	4
F_{10}	*QTR*	The (weighted) percentage of the query concepts contained in the four fields. All query concepts are either equally weighted or weighted by their inverse document frequencies.	8
F_{11}	*QTR*	BM25 and TFIDF similarities between q and the four fields. [Duan et al. 2010; McCreadie et al. 2011]	8
F_{12}	*TR*	Whether T (or the Web pages of embedded URLs) has more than 50% content in English. [McCreadie et al. 2011; Metzler and Cai 2011]	1
F_{13}	*TR*	Whether T has at least one URL in its tweet message field. [Duan et al. 2010; McCreadie et al. 2011; Metzler and Cai 2011]	1
F_{14}	*TR*	The count of the Twitter user of T mentioned by the tweets in the collection. [Duan et al. 2010]	1
F_{15}	*TR*	Whether T is a retweet (or a reply tweet). [Duan et al. 2010; Metzler and Cai 2011]	2
F_{16}	*QTR*	The percentage of the related concepts of q contained in the four fields. The related concepts of q are the top three frequent *PN* concepts among the top 10 web documents retrieved by Google with respect to q.	4
F_{17}	*QTR*	The percentage of the related nouns of q contained in the four fields. The related nouns are the nouns with the top three document frequencies among the top 10 web documents retrieved by Google with respect to q.	4
F_{18}	*QTR*	Whether the order of query terms appearing in the four fields is the same as that in q.	4

A query can be represented by a set of concepts as illustrated by the following example.

Example 2. Given a query of "*Australian Open Djokovic vs. Murray*", it contains five concepts. They are three *PN* concepts, "*Australian Open*," "*Djokovic*" and "*Murray*," an *SNP* concept, "*Djokovic Murray*" ("*vs.*" is omitted as a stop word) and a *CNP* concept, "*Australian Open Djokovic Murray*."

We propose the features (say F_{10}) involving query concepts because they capture the similarities between queries and tweets better than query terms as illustrated by the following example.

Example 3. Given the query q = "*Australian Open Djokovic vs. Murray*", a T-tweet d_1 = "*and Djokovic it is.... Murray becoming more like England football team...failing where it matters...*" and a T-tweet d_2 = "*Can't stop watching the Australian Open!*", d_1 contains two query terms, "*Djokovic*" and "*Murray*" and d_2 also contains two query terms, "*Australian*" and "*Open*". But d_1 is relevant to q while d_2 is irrelevant. In terms of query concepts, d_1 contains three out of five query concepts, "*Djokovic*", "*Murray*" and "*Djokovic Murray*" but d_2 contains only one query concept, "*Australian Open*".

There are eight features with the ID of F_{10}. One of the features is the percentage of the query concepts contained in the tweet message field. As illustrated by Example 3, the more query concepts a tweet contains, the more likely the tweet is relevant to the query. The value of this feature for d_1 is 3/5 while that for d_2 is 1/5. Another member feature is the weighted percentage of the query concepts contained in the tweet message field. Since a concept can be weighted by its inverse document frequency (*idf* for short), the weighted percentage of the query concepts contained in the tweet message field is the ratio of the sum of the *idf*s of the query concepts contained in the tweet message field over the sum of the *idf*s of all query concepts. If we consider the four fields of TU-tweets (the three fields and their union), eight such features can be calculated over the four fields of TU-tweets accordingly.

The features with the IDs of F_{16} and F_{17} calculate the numbers of the related concepts and the related nouns of queries in the different fields of tweets. A person who writes a tweet specifies an event by a set S_1 of concepts or terms. A person who queries the same event may utilize another set S_2 of concepts or terms. The concepts or terms in S_1 are related to those in S_2. Let us illustrate these features with the following Example.

Example 4. Given a query "*White House spokesman replaced*" and a T-tweet d_1 = "*Jay Carney named as Barack Obama's press secretary*," d_1 is relevant to the query, although it does not contain any query concepts or terms. "*Jay Carney*" is a related concept to the query, as it is one of the three most frequent *PN* concepts from the top 10 Web documents retrieved by Google with respect to the query. Therefore, the match of "*Jay Carney*" is an indicator of d_1's relevance to the query.

To build the two tweet type-specific rankers, we partition TREC relevance judgments of tweets into a set of labeled T-tweets and a set of labeled TU-tweets. We use the former set of T-tweets as the training data for learning a *T-tweet Ranker* and the latter set of TU-tweets for learning a *TU-tweet Ranker*, respectively. For building a *T-tweet ranker*, we convert each training example (T-tweet) into a vector of the proposed features that are applicable for T-tweets. Then, we feed the vectors of features into RankSVM to generate a *T-tweet Ranker*. We repeat the same procedure as before by using the training data for TU-tweets to generate a *TU-tweet Ranker*.

3.3. Preference Classifier

The two tweet type-specific rankers only provide the preference between two tweets of the same type. In order to merge the rankings of T-tweets and TU-tweets, a classifier is proposed to determine the preference of each T-tweet with respect to each TU-tweet. We employ the SVM model [Joachims 1999] to perform such determination. In particular, each training example is a triple of $< d_1, d_2, label >$, where d_1 is a T-tweet, d_2 is a TU-tweet and the *label* indicates whether d_1 is preferred over d_2 or vice versa. We again use TREC relevance judgments as the training data. Specifically, for a training query, a labeled T-tweet d_1 and a labeled TU-tweet d_2 form a training example (pair), only if their labels of relevance to that query are different. The different labels of d_1 and d_2 imply that d_1 is preferred over d_2 or vice versa.

To learn such a classifier, we reuse the features in Table I and they are referred to as *ranking features*. We also propose a set of new features that captures the difference of the corresponding (ranking) features of d_1 and d_2 with respect to a query. Let us call this set of new features *dependent features*. Each dependent feature aims at a direct comparison of relevance between d_1 and d_2. It is calculated by a T-tweet (ranking) feature minus a corresponding TU-tweet (ranking) feature. For example, given the feature group F_{11}, a T-tweet feature is the BM25 similarity between a query q and the tweet message field of d_1. But four corresponding TU-tweet features are the BM25 similarities between q and the four fields of d_2, respectively. Thus, four dependent features are obtained by subtracting the four TU-tweet features from the T-tweet feature, respectively. A (preference) classifier can be learned by using these features and the training examples. In our preliminary experiments, the classifier using both ranking features and dependent features performed better than the classifiers that just use either ranking features or dependent features.

3.4. Greedy Merging Algorithm

After we build the two tweet type-specific rankers and the preference classifier, we can rank tweets with respect to a test query q'. First, we use these two rankers to rank T-tweets and TU-tweets with respect to q' separately. Then, we employ the preference classifier to compute the preference between any two tweets, one from each ranking. This constitutes three sets of preferences: one for any two T-tweets, one for any two TU-tweets and one for any T-tweet and any TU-tweet. The goal is to merge the two rankings into a ranking that agrees with these three sets of preferences as much as possible. Cohen et al. [1998] showed that the problem of finding the ordering that agrees best with a given set of preferences is NP-complete. Therefore, we propose a quadratic greedy merging algorithm. To merge a ranking of T-tweets and a ranking of TU-tweets, this algorithm always picks the tweet that has the smallest sum of the degrees of the preferences of other tweets (that have not been picked) over it. This makes the merged ranking consistent with the three sets of preferences, if there is no inconsistency among the three sets of preferences.

Let T and TU be a ranking of T-tweets and a ranking of TU-tweets, respectively. They are defined as follows. We assign (numerical) subscripts to the T-tweets in T so that the T-tweets with smaller subscripts have higher preferences. The same applies to TU. For convenience of presentation, we give the T-tweets in T the subscripts from 1 to m and the TU-tweets in TU the subscripts from $m+1$ to $m+n$. But the comparison between a subscript of a T-tweet and that of a TU-tweet does not indicate a preference between them.

$$\begin{array}{ll} T = \left[d_1, \ldots, d_m\right] & \text{s.t. } d_i \succ d_j, 1 \le i < j \le m \\ TU = \left[d_{m+1}, \ldots, d_{m+n}\right] & \text{s.t. } d_i \succ d_j, m+1 \le i < j \le m+n. \end{array} \tag{1}$$

Let $f_p : \Omega^T \times \Omega^{TU} \to R$ be a preference function which maps a pair of a T-tweet d_i and a TU-tweet d_j to a real number. Ω^T and Ω^{TU} are the T-tweet space and the TU-tweet space, respectively. If the real number is positive, $d_i \succ d_j$; if it is negative, the reverse is true; if it is zero, there is no preference between d_i and d_j. The magnitude of the number indicates the degree of the preference. We assume that the real number being zero does not occur, which is true in practice. This function corresponds to the preference classifier (see Section 3.3). Let D be the union of T and TU, $D = T \cup TU = [d_1, \ldots, d_m, d_{m+1}, \ldots, d_{m+n}]$. Let $Pref(d_i, d_j)$ denote the preference between a tweet d_i and another tweet d_j in D. $Pref(d_i, d_j)$ can be defined as follows.

$$Pref(d_i, d_j) = \begin{cases} d_i \succ d_j & 1 \le i < j \le m \\ d_i \succ d_j & m+1 \le i < j \le m+n \\ d_i \succ d_j & 1 \le i \le m < m+1 \le j \le m+n \text{ and } f_p(d_i, d_j) > 0 \\ d_j \succ d_i & 1 \le i \le m < m+1 \le j \le m+n \text{ and } f_p(d_i, d_j) < 0. \end{cases} \tag{2}$$

Let $RP(i)$ be the ranking position of a tweet d_i in T or TU. Due to the subscript assignments given to the T-tweets in T and the TU-tweets in TU, $RP(i)$ is defined as follows.

$$RP(i) = \begin{cases} i & 1 \leq i \leq m \\ i - m & m+1 \leq i \leq m+n. \end{cases} \tag{3}$$

Let $M = [M_{ij}]_{(m+n)\times(m+n)}$ be the preference matrix for D as defined here. It is consistent with Equation (2) and has the following interpretation: 1) $M_{ij} > 0$ indicates $d_i \succ d_j$; 2) $M_{ij} < 0$ indicates $d_j \succ d_i$; 3) the absolute value of M_{ij} represents the degree of the preference, which is normalized between 0 and 1. Moreover, we propose three weighting parameters, $\lambda_T (> 0)$, $\lambda_{TU} (> 0)$ and $\lambda_{Pairwise} (> 0)$, to be set to the degrees that we trust the three sets of preferences.

$$[M_{ij}]_{(m+n)\times(m+n)} = \begin{cases} \lambda_T \cdot \frac{RP(j)-RP(i)}{\max\{RP(i),\, RP(j)\}} & 1 \leq i, j \leq m \\ \lambda_{TU} \cdot \frac{RP(j)-RP(i)}{\max\{RP(i),\, RP(j)\}} & m+1 \leq i, j \leq m+n \\ \lambda_{Pairwise} \cdot \frac{f_p(d_i,d_j)}{\max_{1\leq s\leq m<m+1\leq t\leq m+n}\{|f_p(d_s,d_t)|\}} & 1 \leq i \leq m < m+1 \leq j \leq m+n \\ -M_{ji} & 1 \leq j \leq m < m+1 \leq i \leq m+n. \end{cases} \tag{4}$$

We now explain why M is defined in such a manner. Specifically, we elaborate the intuition of each of the four components of M.

(1) The first component $\left(\lambda_T \cdot \frac{RP(j)-RP(i)}{\max\{RP(i),\, RP(j)\}}\right)$ indicates the preference between any two T-tweets, d_i and d_j. If $1 \leq i < j \leq m$, then $RP(i) < RP(j)$ and therefore $M_{ij} > 0$, indicating $d_i \succ d_j$; if $1 \leq j < i \leq m$, then $RP(j) < RP(i)$ and therefore $M_{ij} < 0$, indicating $d_j \succ d_i$. The degree of the preference is normalized between 0 and 1 by $\max\{RP(i), RP(j)\}$. Moreover, it is also easy to verify that $M_{ij} < M_{i(j+1)}$ if $1 \leq i \leq m, 1 \leq j \leq m-1$. This is reasonable, because as the separation between two T-tweets increases, so is the degree of the preference. We propose such a heuristic method to measure the degree of the preference between two T-tweets, because most learning to rank algorithms, such as RankSVM, produce the ranking scores that have no meaning in an absolute sense and can only be used for ordering.

(2) The second component $\left(\lambda_{TU} \cdot \frac{RP(j)-RP(i)}{\max\{RP(i),\, RP(j)\}}\right)$ has the same interpretation as the first component, except that it indicates the preference between any two TU-tweets, d_i and d_j.

(3) The third component $\left(\lambda_{Pairwise} \cdot \frac{f_p(d_i,d_j)}{\max_{1\leq s\leq m<m+1\leq t\leq m+n}\{|f_p(d_s,d_t)|\}}\right)$ indicates the preference between a T-tweet d_i and a TU-tweet d_j. If $f_p(d_i, d_j) > 0$, then $M_{ij} > 0$ and $d_i \succ d_j$; if $f_p(d_i, d_j) < 0$, then $M_{ij} < 0$ and $d_j \succ d_i$. The degree of the preference is normalized between 0 and 1 by $\max_{1\leq s\leq m<m+1\leq t\leq m+n}\{|f_p(d_s, d_t)|\}$.

(4) The fourth component indicates that the preference between a TU-tweet d_i and a T-tweet d_j is the negation of the preference between d_j and d_i.

Let us illustrate the preference matrix M with the following example.

Example 5. Given two T-tweets, d_1 and d_2 and three TU-tweets: d_3, d_4 and d_5, the three sets of the preferences of these tweets are shown in Table II.

Table II. Three Sets of Preferences for Example 5

Rankers and Classifier	Tweet Preferences and Their Ranking Positions
T-tweet Ranker	$d_1 \succ d_2$; $RP(d_1) = 1$ and $RP(d_2) = 2$;
TU-tweet Ranker	$d_3 \succ d_4 \succ d_5$; $RP(d_3) = 1$, $RP(d_4) = 2$ and $RP(d_5) = 3$;
Preference Classifier	$f_P(d_1, d_3) = -0.9 (d_3 \succ d_1)$; $f_P(d_2, d_3) = -1 (d_3 \succ d_2)$ $f_P(d_1, d_4) = -0.7 (d_4 \succ d_1)$; $f_P(d_2, d_4) = -0.8 (d_4 \succ d_2)$ $f_P(d_1, d_5) = 0.6 (d_1 \succ d_5)$; $f_P(d_2, d_5) = 0.5 (d_2 \succ d_5)$

For simplicity, we assume that the three weighting parameters: λ_T, λ_{TU} and $\lambda_{Pairwise}$ are all equal to 1. The preference matrix for Example 5 is shown here.

$$M = \begin{bmatrix} 0 & 0.5 & -0.9 & -0.7 & 0.6 \\ -0.5 & 0 & -1 & -0.8 & 0.5 \\ 0.9 & 1 & 0 & 0.5 & 0.67 \\ 0.7 & 0.8 & -0.5 & 0 & 0.33 \\ -0.6 & -0.5 & -0.67 & -0.33 & 0 \end{bmatrix}.$$

To merge the two rankings, we propose a greedy merging algorithm. To explain the proposed merging algorithm, we first define *Dispreferness*.

Definition 3.3 (*Dispreferness*). Given the preference matrix M and a tweet d_i, the *Dispreferness* of the tweet d_i is calculated by

$$Dispreferness(M, d_i) = \sum_j |\min\{0, M_{ij}\}|. \tag{5}$$

Given a tweet d_i, if it is preferred over a tweet d_j, then $M_{ij} > 0$ and $|\min\{0, M_{ij}\}| = 0$ will not contribute to $Dispreferness(M, d_i)$. On the other hand, if d_j is preferred over d_i, then $M_{ij} < 0$ and $|\min\{0, M_{ij}\}|$ contributes a positive value to $Dispreferness(M, d_i)$. $Dispreferness(M, d_i)$ is the sum of the degrees of the preferences of other tweets over d_i. The greedy merging algorithm, called *GreedyMerging*, merges two rankings of tweets by placing the tweet d with the least $Dispreferness(M, d)$ in the first position of the merged ranking L. Placing d in such a position of L may incur a certain amount of inconsistency and this amount is $Dispreferness(M, d)$. Compared to any other tweet placed at the first position, this amount of inconsistency is the least. Then, after removing d from the matrix M and re-computing the *Dispreferness* of other tweets, it iteratively places the tweet that has the least *Dispreferness* in the next position in L. The algorithm always picks the tweet that incurs the least amount of inconsistency at the time it is picked. Details of the algorithm are shown in Algorithm 1.

The following proposition demonstrates that the proposed algorithm is theoretically reasonable, because if there is no inconsistency among the three sets of preferences, the optimal ranking of tweets will be achieved by *GreedyMerging*.

PROPOSITION 3.4. *If there is no inconsistency among all the preferences from the T-tweet Ranker, the TU-tweet Ranker and the pairwise classifier, GreedyMerging produces the optimal ranking.*

PROOF. Assuming no inconsistency among all the preferences, there must be a linear order of tweets in terms of their preferences: $d_{i_1} \succ d_{i_2} \succ \cdots \succ d_{i_n}$. This linear order is an optimal ranking of tweets because any pair of tweets is ordered by their preferences. The first tweet d_{i_1} has zero *Dispreferness* because no tweet has preference over it. Moreover, no other tweet, say d, has zero *Dispreferness*, since d_{i_1} is preferred over d, causing $Dispreferness(M, d) > 0$. *GreedyMerging* inserts d_{i_1} into the first position of the merged ranking L. After d_{i_1} is chosen and the matrix is updated by deleting the

ALGORITHM 1: The *GreedyMerging* Algorithm

Input: A ranking of T-tweets: T; a ranking of TU-tweets: TU; the preferences of pairs of a T-tweet and a TU-tweet, f_p; Three weighting parameters: λ_T, λ_{TU} and $\lambda_{Pairwise}$;
Output: A merged ranking of tweets L;
1. Union two rankings of tweets $D = T \cup TU$;
2. Create the preference matrix $M_{|D|\times|D|}$ for D, based on T, TU, f_p, λ_T, λ_{TU} and $\lambda_{Pairwise}$;
3. **while**$(D \neq \emptyset)$
4.   Find the tweet d with the least $Dispreferness(M, d)$;
5. $d = \arg\min_{d\in D}\{Dispreferness(M, d)\}$;
6.   Insert d into the merged ranking L;
7.   Update D and M:
8. $D = D - \{d\}$;
9. $M_{|D-1|\times|D-1|} = M_{|D|\times|D|} - [d]$; // *deleting the row and column representing d;*
10. **end**

row and the column representing d_{i_1}, the second tweet d_{i_2} has no tweet preferred over it among the remaining tweets and only its *Dispreferness* is zero. *GreedyMerging* inserts d_{i_2} into the second position of L. The same argument is applied repeatedly until all tweets are inserted into L. □

After a ranking of T-tweets and a ranking of TU-tweets are merged by *GreedyMerging*, we obtain a ranking L of both types of tweets but their IR scores are absent. We need to assign some (pseudo) IR scores to the tweets in L so that the time-related relevance scores of tweets (to be given in Section 4.2) can be combined with the IR scores to yield the similarity scores for the final ranking of tweets (see Section 4.3). The ranking of the tweets in descending order of their pseudo IR scores should be identical to L. We adopt the conversion proposed in Lee [1997]. Given a ranking of n tweets, $L = [d_1, \ldots, d_n]$, where the subscript i of tweet d_i is its ranking position, we assign d_i an IR score $IR(d_i)$ as follows.

$$IR(d_i) = 1 - \frac{i-1}{n}. \tag{6}$$

4. TEMPORAL USAGE IN RETRIEVAL

In the first phase, tweets are ranked by only considering their lexical similarities to queries. In this section, we discuss how to use the temporal information (publishing times) of tweets to improve retrieval effectiveness.

4.1. Time Representation

In this section, we describe the temporal representation of tweets with respect to queries. Each query q has a timestamp t and only the tweets published on or before t are considered to be relevant. Given a tweet d with a publishing time t_d, we adopt the time representation $f(t_d, t)$ proposed in Efron and Golovchinsky [2011] with the interpretation that $f(t_d, t) = 0$ means the tweet d is published on the same day as t and $f(t_d, t) = n (n > 0)$ indicates the tweet d is published n days before t.

4.2. Query Type Determination

In this section, we first propose a method to classify queries by the temporal distributions of their top tweets and then present different ways to measure the temporal relevance of tweets to classified queries.

There are three types of queries as discussed in Section 1. We utilize the top tweets from the first phase to classify a query into one of these three types. Specifically, for a query q with a timestamp t, let $D = \{d_1, \ldots, d_K\}$ be the top K tweets retrieved by the divide-and-conquer method in the first phase. Let $T = \{t_1, \ldots, t_K\}$ be the set of publishing times associated with those top K tweets, where each publishing time t_i presents either the same time t or a time before t. Let $T_D = \{t'_i | t'_i = f(t_m, t), t_m \in T\}$ be the set of the unique time representations of their publishing times. Let $I(t_j, t, t'_i)$ be an indicator function.

$$I(t_j, t, t'_i) = \begin{cases} 1 & f(t_j, t) = t'_i \\ 0 & otherwise. \end{cases} \quad (7)$$

The type of q can be classified as follows.

— q is a time insensitive query if the largest proportion of the top K tweets published on a single day is less than or equal to a certain threshold $p(\leq 0.5)$, that is, Equation (8) holds.

$$\max_{t'_i \in T_D} \left\{ \frac{1}{K} \sum_{t_j \in T} I(t_j, t, t'_i) \right\} \leq p \leq 0.5. \quad (8)$$

— q is a dominant peak query if the largest proportion of the top K tweets published on a certain single day (say t') is greater than a threshold $s(> p)$, that is, Equation (9) holds. Its dominant peak is on t'.

$$\max_{t'_i \in T_D} \left\{ \frac{1}{K} \sum_{t_j \in T} I(t_j, t, t'_i) \right\} > s > p. \quad (9)$$

— q is a nondominant peak query if the largest proportion of the top K tweets published on a single day is less than or equal to s but greater than p, that is, Equation (10) holds. It can have a set of nondominant peaks and the proportion of the top K tweets at each peak is less than or equal to s but greater than p.

$$s \geq \max_{t'_i \in T_D} \left\{ \frac{1}{K} \sum_{t_j \in T} I(t_j, t, t'_i) \right\} > p. \quad (10)$$

The parameters K, p and s are estimated empirically. After a query q is classified into one of the three types, the tweets from the first phase are assigned time-related relevance scores (*TRS*s for short) to q as follows.

— If q is a time-insensitive query, all the tweets retrieved from the first phase are not assigned any *TRS*s. This implies that time has no impact on ranking the tweets with respect to q.

— If q is a dominant peak query, that is, the temporal distribution of its top K tweets has a dominant peak on t'_i (the t'_i days before t), a tweet d (published on t_d) is assigned a *TRS* as follows.

$$TRS(t_d, t) = \frac{1}{2\delta} \exp\left\{ -\frac{|f(t_d, t) - t'_i|}{\delta} \right\}. \quad (11)$$

This function is in the form of the Laplace distribution [Laplace 1774]. When the tweet occurs at the peak, its *TRS* is normalized by $max_{t_d}\{TRS(t_d, t)\}$ to be 1. The farther the tweet d is temporally away from the peak, the smaller the *TRS* of d is. In other words, tweets temporally closer to the peak are given higher *TRS*s. We tested different exponential functions and found that the Laplace-like function performed best. It has a single peak on t'_i and its variance $2\delta^2$ can be estimated by the maximum likelihood method.

$$\hat{\delta} = \frac{1}{|T_D|} \sum_{t'_i \in T_D} \left| \frac{1}{K} \sum_{t_j \in T} I(t_j, t, t'_i) - \hat{\mu} \right| \text{ s.t. } \hat{\mu} = \frac{1}{|T_D|} \sum_{t'_i \in T_D} \left(\frac{1}{K} \sum_{t_j \in T} I(t_j, t, t'_i) \right). \quad (12)$$

— If q is a nondominant peak query, that is, the temporal distribution of its top K tweets has a set of nondominant peaks at a set of time representations $P = \{t'_1, \ldots, t'_{|P|}\}$, a tweet (published on t_d) is assigned a TRS as follows.

$$TRS(t_d, t) = \begin{cases} \frac{\sum_{t_j \in T} I(t_j,t,t'_n)}{\max_{t'_m \in P}\left\{\sum_{t_j \in T} I(t_j,t,t'_m)\right\}} \cdot \frac{\sum_{d' \in D_{t'_n}} BM25(d,d')}{|D_{t'_n}|} & f(t_d,t) \notin P \\ \frac{\sum_{t_j \in T} I(t_j,t,f(t_d,t))}{\max_{t'_m \in P}\left\{\sum_{t_j \in T} I(t_j,t,t'_m)\right\}} & f(t_d,t) \in P \end{cases} \quad (13)$$

$$\text{s.t. } D_{t'_m} = \left\{d' | f(t_{d'}, t) = t'_m, t'_m \in P\right\}, t'_n = \arg\max_{t'_m \in P} \left\{ \frac{\sum_{d' \in D_{t'_m}} BM25(d,d')}{|D_{t'_m}|} \right\}$$

Let us explain the intuition of Equation (13) as follows.

(1) Suppose that the distribution of q's top K tweets has multiple nondominant peaks.

   (a) For a tweet (published on t_d) belonging to the highest peak at time $f(t_d, t)$, its TRS is assigned to be 1, that is, $\max_{t'_m \in P}\left\{\sum_{t_j \in T} I(t_j, t, t'_m)\right\} = \sum_{t_j \in T} I(t_j, t, f(t_d, t)) \Rightarrow \frac{\sum_{t_j \in T} I(t_j,t,f(t_d,t))}{\max_{t'_m \in P}\left\{\sum_{t_j \in T} I(t_j,t,t'_m)\right\}} = 1$

   (b) For a tweet d (published on t_d) belonging to a nonhighest peak at time $f(t_d, t)$, its TRS is the ratio of the number of the top K tweets at that peak to that at the highest peak, that is, $TRS(t_d, t) = \frac{\sum_{t_j \in T} I(t_j,t,f(t_d,t))}{\max_{t'_m \in P}\left\{\sum_{t_j \in T} I(t_j,t,t'_m)\right\}}$.

   (c) For a tweet d (published on t_d) not belonging to any peak, we first determine which peak contains the tweets that are most similar to d. We use *BM25* to measure the average similarity of d to the tweets at a peak.[2] Then we pick the peak with the highest average similarity to d, say the peak at time t'_n. Let $S_2 \left(= \frac{\sum_{d' \in D_{t'_n}} BM25(d,d')}{|D_{t'_n}|} \right)$ denote that highest average similarity. Each tweet in that picked peak is assigned the same TRS. Let $S_1 \left(= \frac{\sum_{t_j \in T} I(t_j,t,t'_n)}{\max_{t'_m \in P}\left\{\sum_{t_j \in T} I(t_j,t,t'_m)\right\}} \right)$ denote that TRS of a tweet in that picked peak. Finally we assign d a TRS that is the product of S_1 and S_2. In other words, the tweets in different peaks describe different events related to q. We first determine which related event d is likely to describe. The likelihoods of d describing different events are measured by the average similarities of d to those tweets at different peaks. We then assign d a TRS that is equal to the highest average similarity multiplied by the TRS of a tweet describing the same related event as d does.

[2]We utilize the tweet message field without exploring the Web pages of URLs if present.

(2) Suppose that the distribution of q's top K tweets has a single nondominant peak, the same approach is used.
   (a) For a tweet belonging to the unique peak, its TRS is assigned to be 1.
   (b) For a tweet d that does not belong to that peak, the average similarity of d to the tweets in that peak is computed. It is multiplied by the TRS of any tweet at the peak (having a value of 1 due to the single peak) to yield the TRS of d.

4.3. Aggregation of IR Scores and Time-Related Relevance Scores

The first phase calculates the IR scores of tweets with respect to a query q. The second phase of the method calculates the time-related relevance scores of tweets by using temporal information. Given a tweet d, let $IR(d)$ and $TRS(d)$ be the IR score of d and the time-related relevance score of d, respectively. An aggregation score $AGS(d)$ can be calculated in the manner of F-measure [Rijsbergen 1979] (see Equation (14)). The tweets are arranged in descending order of the aggregation scores. Although the F-measure is usually used as an evaluation measure, it can be employed to balance $IR(d)$ and $TRS(d)$. The parameter β aims at balancing the contributions of $IR(d)$ and $TRS(d)$ to the aggregation score. The appropriate value of β is estimated in the experiments. Experimental results demonstrate that such an aggregation outperforms other aggregations, such as CombSUM and CombMNZ [Shaw et al. 1994].

$$AGS(d) = (1 + \beta^2)\frac{IR(d) \cdot TRS(d)}{\beta^2 \cdot IR(d) + TRS(d)} \tag{14}$$

5. EXPERIMENT SETUP

5.1. TREC Tweets2011 Collection

TREC 2011 released a tweet collection called Tweets2011 for the real-time ad-hoc retrieval task of the microblog track. The collection consists of about 16 million tweets sampled from Twitter over 17 days (from 1/23/2011 to 2/8/2011). Instead of directly giving those tweets, TREC 2011 provided two tools for participating groups to crawl the collection. One tool employing a Twitter API provides an information-rich collection of tweets in the JSON format. The other one just crawls the HTML pages of tweets. The efficiency of the first tool is very low, crawling about 150 tweets per hour due to the limitation of the Twitter API. The second tool only crawls the HTML pages of tweets and it is far more efficient than the first tool. However, some social information, such as Twitter user profile, is absent in the HTML collection of tweets. We utilize the second tool in this article. Since Twitter users might delete their tweets at any time, change their usernames or change the public sharing properties of their tweets, it is possible that some tweets are successfully crawled by some groups while become unavailable when other groups are crawling. The statistics of our crawled tweet collection is shown in Table III. In the TREC Tweets2011 collection crawled by us, 16.7% of tweets are TU-tweets. We crawled the Web pages whose URLs are linked by the TU-tweets in the collection, which results in another collection of about 2.3 million Web pages.[3]

5.2. TREC 2011 and 2012 Queries and TREC Relevance Judgments

TREC 2011 released 50 queries and TREC 2012 released 60 queries. TREC required both sets of queries to be retrieved over the TREC Tweets2011 collection. Each query represents an information need at a specific time. An example query is shown in Figure 2. The num tag encloses the ID of the query. The query tag encloses the query.

[3]Some URLs given by the TU-tweets are not available during our crawling.

Table III. The Statistics of Our Crawled TREC Tweets2011 Collection

HTTP Response Code	Tweet Count	Description
200 (OK)	14437978	Successfully downloaded tweets.
302 (Found)	1612080	Downloaded retweets via redirects.
403 (Forbidden)	339147	The tweets without public sharing properties.
404 (Not Found)	707403	The tweets no longer available.

```
<top>
<num> Number: MB075 </num>
<query> Aguilera super bowl fail </query>
<querytime> Tue Feb 08 21:56:22 +0000 2011 </querytime>
<querytweettime> 35094611483426816 </querytweettime>
</top>
```

Fig. 2. An example of TREC query.

The querytime tag gives the timestamp of the query in the form of ISO standard. Each tweet is assigned a unique tweet ID. The descending ordering of the IDs of tweets can be interpreted as the reverse-chronological order of their publishing times. The querytweettime tag represents the timestamp of the query. In response to a query with a timestamp t, only the tweets whose IDs are not greater than t need to be considered.

TREC also provided the relevance judgments of tweets with respect to those two sets of queries. TREC assessors read tweets, then followed the URLs inside them and finally labeled them in a three point scale: "highly relevant," "relevant," and "irrelevant." For the TREC 2011 queries, 49 (out of 50) queries have at least one relevant or highly relevant tweet and 33 (out of 50) queries have at least one highly relevant tweet. For the TREC 2012 queries, 59 (out of 60) queries have at least one relevant or highly relevant tweet and 56 (out of 60) queries have at least one highly relevant tweet. "Highly relevant" tweets are preferred over "relevant" tweets that are preferred over "irrelevant" tweets. For the set of TREC 2011 queries, we use the set of TREC 2012 queries as the training query set and their corresponding TREC relevance judgments as the training data and vice versa.

5.3. Relevance Criteria

There are two relevant criteria: 1) both relevant and highly relevant tweets are considered relevant; 2) only the highly relevant tweets are considered relevant. In our experiments, we denote these two relevant criteria as the *relevant criterion* and the *highly relevant criterion*, respectively. Our results are evaluated by these two criteria.

5.4. Evaluation Measures

In this article, we employ the precision at top 30 tweets (P30 for short), the mean average precision (MAP for short) and the normalized discounted cumulative gain at top 30 tweets (NDCG@30 for short) as the evaluation measures. To evaluate the retrieval effectiveness of our method that does not involve ranking tweets in reverse-chronological order, we use MAP as the primary measure and P30 and NDCG@30 as the secondary measures. However, we use P30 as the primary measure and MAP as the secondary measure to evaluate the performance of our method in ranking tweets in reverse-chronological order, as TREC 2011 stipulated that P30 is the official measure for the reverse-chronological rankings of tweets [Ounis et al. 2011]. In this article, we only consider statistical significance at $p < 0.05$ according to one-sided paired t-test.

6. EXPERIMENTAL RESULTS

In this section, we evaluate our method by using both TREC 2011 and TREC 2012 queries over the TREC Tweets2011 collection. Two sets of experiments are conducted to evaluate our two-phase method. One set evaluates the retrieval performance of the divide-and-conquer method in the first phase; the other set evaluates that of utilizing temporal information of tweets in the second phase. We also compare the performance of our two-phase method with various state-of-the-art methods. In particular, we conduct the experiments to reveal the answers to the following research questions.

— Is it beneficial to apply the divide-and-conquer strategy on ranking tweets? In other words, would there be any benefit to rank the two types of tweets separately, compared with the method of ranking them simultaneously? Experiments are conducted to verify the motivation of leveraging the structural difference of tweets.
— What are the important features for learning to rank tweets? We study the degrees of importance of the proposed features for ranking T-tweets, TU-tweets and both types of tweets together.
— What are the effectiveness and the efficiency of the proposed divide-and-conquer algorithm for ranking tweets?
— How many queries do benefit from the divide-and-conquer algorithm and how many queries do not? In particular, we conduct a result analysis of the proposed algorithm and discuss the reasons why our algorithm helps or hurts some typical queries.
— Is it necessary to have two different types of time sensitive queries (dominant peak queries vs. nondominant peak queries)? Experiments are conducted to validate the benefit of our proposed categories of temporal queries.
— How to estimate the parameters K, p, s and β that are used by our temporal classification of queries?
— Does the utilization of temporal information provide further improvement over the algorithm using the divide-and-conquer strategy?
— How many queries do benefit from the usage of temporal information and how many queries do not? We analyze the performance of our method query by query and discuss the reasons why our method improves or deteriorates the performance of some queries.
— How is the performance of our two-phase method that combines the usage of temporal information with the divide-and-conquer approach, compared with various state-of-the-art methods?

6.1. Relevance Ranking Analysis

In this section, we first demonstrate the necessity of considering the structural difference of tweets. Second, we study the degrees of importance of the proposed features for ranking tweets. Third, we study the effectiveness of the divide-and-conquer method by comparing it with various baselines. Fourth, we discuss the efficiency of the proposed method. Finally, we conduct a result analysis and discuss why some queries are helped or hurt by our method.

6.1.1. The Motivation of Considering Structural Difference of Tweets. To validate the motivation of using the divide-and-conquer strategy to address the structural difference of tweets, we analyze a uniform ranker (denoted by *Uniform Ranker*) and the two tweet type-specific rankers (denoted by *T-tweet Ranker* and *TU-tweet ranker* respectively). The *Uniform Ranker* is constructed by using RankSVM. It is learned over the training data consisting of a set of training queries and both types of labeled tweets. It ranks both types of tweets simultaneously. We first apply the *Uniform Ranker* to produce a ranking of tweets. This ranking R consists of both types of tweets and is then

Table IV. *Uniform Ranker* vs. Tweet Type-Specific Rankers

	TREC 2011					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
Uniform Ranker (for T-tweets)	0.0613	0.1497	0.1142	0.0231	**0.0202**	0.1030
T-tweet Ranker	**0.0768** †	**0.1639**	**0.1327** †	**0.0297**	0.0152	**0.1151**
Uniform Ranker (for TU-tweets)	0.4440	0.5013	0.4762	0.3966	**0.2364**	0.4831
TU-tweet Ranker	**0.4715** †	**0.5102**	**0.4952** †	**0.4042**	0.2242	**0.4923**

	TREC 2012					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
Uniform Ranker (for T-tweets)	0.0474	0.1266	0.0670	0.0182	0.0411	**0.0706**
T-tweet Ranker	**0.0510**	**0.1373**	**0.0678**	**0.0184**	**0.0446**	0.0696
Uniform Ranker (for TU-tweets)	0.2882	0.4226	0.2798	0.2447	0.2435	0.2722
TU-tweet Ranker	**0.2926** †	**0.4367** †	**0.2949**	**0.2489** †	**0.2548**	**0.2829**

Note: † indicates statistically significant improvements over the corresponding baselines

partitioned into two rankings, R_1 for T-tweets and R_2 for TU-tweets. The relative order of the tweets in each $R_i (i = 1, 2)$ is the same as that in R. Two tweet type-specific rankers are constructed by using RankSVM too. The *T-tweet Ranker* is learned by only using the portion of T-tweets in the training data and the *TU-tweet ranker* is learned by only using the portion of TU-tweets. They are used to rank the two types of tweets separately. Finally, we compare the performance of these two rankings R_1 and R_2 with those of the two corresponding rankings from the two tweet type-specific rankers. The performance is evaluated by using both relevant criteria. The comparison of their performance is shown in Table IV.

We make three observations based on the information shown in Table IV. First, the *Uniform Ranker* achieves decent performance in ranking TU-tweets but it performs poorly in ranking T-tweets with respect to both sets of TREC 2011-2012 queries. Second, the two tweet type-specific rankers consistently outperform the *Uniform Ranker* in terms of MAP, P30 and NDCG@30 by the relevant criterion over both sets of queries. Third, for the highly relevant criterion, the two type-specific rankers show somewhat stronger performance than the *Uniform Ranker*. Specifically, for the TREC 2011 queries, the two rankers consistently outperform the *Uniform Ranker* in MAP and NDCG@30 but get marginal deteriorations in P30. For the TREC 2012 queries, the *TU-tweet Ranker* consistently outperforms the *Uniform Ranker* in all three measures. The *T-tweet Ranker* outperforms the *Uniform Ranker* in terms of MAP and P30 but gets a negligible deterioration in NDCG@30. These three observations validate the motivation and the necessity of treating the two types of tweets separately.

6.1.2. Feature Analysis. It is worth investigating the degrees of importance of the proposed features for learning to rank tweets. We sort the proposed features in descending order of their degrees of importance that are calculated by RankSVM [Bian et al. 2010]. Specifically, we study the degrees of importance of the features applicable for the *T-tweet Ranker*, the *TU-tweet Ranker* and the *Uniform Ranker*. Table V shows the top 10 important features for each of these three rankers.

From Table V, several observations can be made. First, *QTR* features (the features whose calculations depend on tweets and queries) are more important than *TR* features (the features whose calculations depend on tweets only) in ranking T-tweets, TU-tweets or ranking them simultaneously, because *QTR* features dominate the top 10 features for these three rankers. Second, the top 10 features for the *T-tweet Ranker* are very different from those for the *TU-tweet Ranker*. In particular, only 3 of the top

Table V. Top 10 Features for *T-tweet Ranker*, *TU-tweet Ranker*, and *Uniform Ranker*

Top 10 Features for *T-tweet Ranker*				Shared by Rankers Below	
Rank	ID	Type	Feature Description	TU-Tweet	Uniform
1	F_{17}	*QTR*	The percentage of related nouns of Q contained in the tweet message field.	√	√
2	F_{10}	*QTR*	The percentage of query concepts contained in the tweet message field	√	√
3	F_{10}	*QTR*	The weighted percentage of query concepts contained in the tweet message field	√	√
4	F_{12}	*TR*	Whether the tweet message field has more than 50% content in English.		
5	F_{16}	*QTR*	The percentage of related concepts of Q contained in the tweet message field.		
6	F_{18}	*QTR*	Whether the order of query terms in the tweet message field is the same as that of in the query		
7	F_3	*QTR*	Whether the tweet message field contains the whole query as a *SNP* or *CNP*.		
8	F_1	*QTR*	The percentage of query terms contained by the hashtags in the tweet.		
9	F_5	*QTR*	Whether the tweet message field contains the key query term.		
10	F_2	*QTR*	The percentage of expansion terms contained by the hashtags in the tweet.		

Top 10 Features for *TU-tweet Ranker*				Shared by Rankers Below	
Rank	ID	Type	Feature Description	T-Tweet	Uniform
1	F_{10}	*QTR*	The weighted percentage of query concepts contained in the union of all three fields.		√
2	F_{10}	*QTR*	The percentage of query concepts contained in the URL title field.		√
3	F_{10}	*QTR*	The percentage of query concepts contained in the URL body field.		√
4	F_{10}	*QTR*	The percentage of query concepts contained in the tweet message field.	√	√
5	F_3	*QTR*	Whether the URL title field contains the whole query as a *SNP* or *CNP*.		√
6	F_{17}	*QTR*	The percentage of related nouns of Q contained in the tweet message field.	√	√
7	F_{10}	*QTR*	The weighted percentage of query concepts contained in the URL body field.		√
8	F_{10}	*QTR*	The weighted percentage of query concepts contained in the tweet message field.		√
9	F_{10}	*QTR*	The percentage of query concepts contained in the union of all three fields.	√	√
10	F_3	*QTR*	Whether the URL body field contains the whole query as a *SNP* or *CNP*.		√

Top 10 Features for *Uniform Ranker*				Shared by Rankers Below	
Rank	ID	Type	Feature Description	T-Tweet	TU-tweet
1	F_{10}	*QTR*	The percentage of query concepts contained in the tweet message field.	√	√
2	F_{10}	*QTR*	The percentage of query concepts contained in the URL title field.		√
3	F_{10}	*QTR*	The weighted percentage of query concepts contained in the tweet message field.	√	√
4	F_{17}	*QTR*	The percentage of related nouns of Q contained in the tweet message field.	√	√

Table V. Continued

Top 10 Features for *Uniform Ranker*				Shared by Rankers Below	
Rank	ID	Type	Feature Description	T-Tweet	TU-tweet
5	F_{10}	*QTR*	The percentage of query concepts contained in the union of all three fields.		√
6	F_{10}	*QTR*	The percentage of query concepts contained in the URL body field.		√
7	F_3	*QTR*	Whether the URL title field contains the whole query as a *SNP* or *CNP*.		√
8	F_{10}	*QTR*	The weighted percentage of query concepts contained in the union of all three fields.		√
9	F_{10}	*QTR*	The weighted percentage of query concepts contained in the URL body field.		√
10	F_3	*QTR*	Whether the URL body field contains the whole query as a *SNP* or *CNP*.		√

10 features for the *T-tweet Ranker* appear among the top 10 important features for *TU-tweet Ranker* and they are not among the top 3 features for the *TU-tweet Ranker*. This observation shows that the *T-tweet Ranker* and the *TU-tweet Ranker* emphasize different features and thus again verifies the motivation and the necessity of ranking these two types of tweets separately. Third, the top 10 important features for the *T-tweet Ranker* are quite different from those for the *Uniform Ranker* while the top 10 important features for the *TU-tweet Ranker* are very similar to those for the *Uniform Ranker*. In particular, only 3 of the top 10 features for the *T-tweet Ranker* appear among those for the *Uniform Ranker* while all the top 10 features for the *TU-tweet Ranker* are the same as those for the *Uniform Ranker* but with a different order. This observation explains why the *Uniform Ranker* achieves decent performance in ranking TU-tweets but suffers poor performance in ranking T-tweets.

6.1.3. The Impact of the Divide-and-Conquer Method. To study the impact of our divide-and-conquer method, four systems are configured. The first system is BM25 similarity [Robertson et al. 1996]. We empirically learn the two parameters b and k for BM25. In particular, the parameter b is learned from 0.5 to 1 with an interval of 0.05 and the parameter k is learned from 1.2 to 2.0 with an interval of 0.1. The combination of these two parameters that optimizes the performance of the TREC 2011 queries is applied to the TREC 2012 queries and vice versa. The second system is the *Uniform Ranker* (see Section 6.1.1). These two methods act as the baselines. The third system is the proposed divide-and-conquer method equipped with a simple merging (called *SimpleMerging*) algorithm. It can act as an alternative to the *GreedyMerging* algorithm to merge the rankings of T-tweets and TU-tweets. The *SimpleMerging* algorithm works as follows. Given a ranking of T-tweets, a ranking of TU-tweets and the preferences of T-tweets relative to TU-tweets, *SimpleMerging* compares the preference between the first T-tweet and the first TU-tweet. If the first T-tweet is preferred over the first TU-tweet, *SimpleMerging* puts the first T-tweet into the merged ranking and then compares the preference between the second T-tweet and the first TU-tweet. Otherwise, *SimpleMerging* puts the first TU-tweet into the merged ranking and then compares the first T-tweet with the second TU-tweet. Repeat the given comparison until all tweets are merged into the final ranking. *SimpleMerging* guarantees to preserve the relative ranking positions of the T-tweets and those of the TU-tweets. Its time complexity is linear. The fourth system is the divide-and-conquer method equipped with the *GreedyMerging* algorithm and its time complexity is quadratic. The three parameters, λ_T, λ_{TU} and $\lambda_{Pairwise}$, of *GreedyMerging* are estimated as follows. We stipulate that the sum of the three parameter values be 1 and each parameter can only be

Table VI. The Comparison of the Divide-and-Conquer Method of *SimpleMerging* or *GreedyMeging* with *Uniform Ranker* and BM25

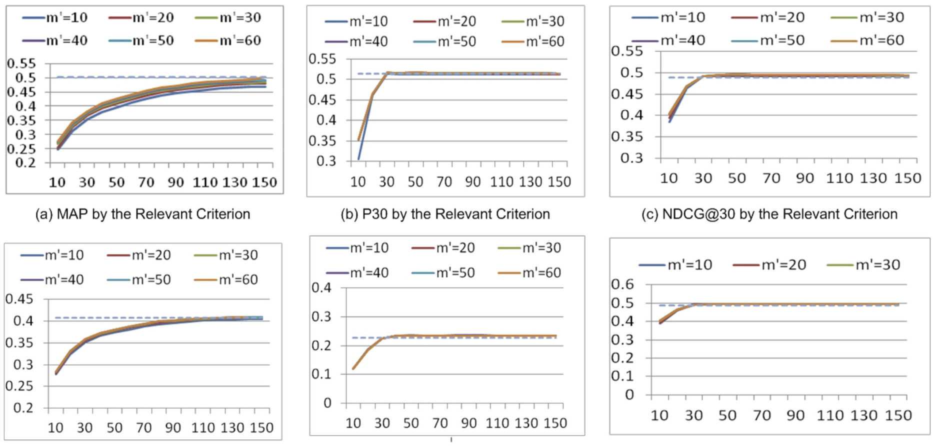
	TREC2011					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
BM25	0.3693	0.3966	0.3747	0.2488	0.1576	0.3474
Uniform Ranker	0.4778 ↑	0.4905 ↑	0.4880 ↑	0.3788 ↑	0.2000 ↑	0.4793 ↑
SimpleMerging	0.4953 ↑	0.5109 ↑	0.4914 ↑	0.3912 ↑	0.2152 ↑	0.4882 ↑
GreedyMerging	**0.5006** ↑ ‡	**0.5143** ↑	**0.4939** ↑	**0.4090** ↑ ‡	**0.2283** ↑ †	**0.4933** ↑
	TREC2012					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
BM25	0.2603	0.3791	0.2207	0.1910	0.2167	0.2319
Uniform Ranker	0.3077 ↑	0.4175 ↑	0.2705 ↑	0.2345 ↑	0.2375 ↑	0.2633 ↑
SimpleMerging	0.3206 ↑	0.4130 ↑	0.2832 ↑	0.2409 ↑ †	0.2357 ↑	0.2710 ↑
GreedyMerging	**0.3259** ↑	**0.4367** ↑	**0.2966** ↑ †‡	**0.2590** ↑ †‡	**0.2583** ↑	**0.2852** ↑ †‡

Note: ↑, †, and ‡ indicate statistically significant improvements over *BM25*, *Uniform Ranker* and *SimpleMergeing*, respectively.

assigned one of 10 possible values: 0.1, ..., 1.0. The combination of these three parameters that optimizes the performance of the TREC 2011 queries is applied to the TREC 2012 queries and vice versa. The performances of these systems are shown in Table VI.

Several observations can be made from the information in Table VI. First, all three learning to rank models, the *Uniform Ranker*, the divide-and-conquer method with the *SimpleMerging* algorithm (the *SimpleMerging* algorithm for short) and the divide-and-conquer method with the *GreedyMerging* algorithm (the *GreedyMerging* algorithm for short) consistently and significantly outperform the BM25 baseline in all measures by both criteria with respect to the two sets of queries. This indicates that using learning to rank techniques benefits the retrieval effectiveness of tweets. Second, for the set of TREC 2011 queries, the *SimpleMerging* algorithm consistently outperforms the *Uniform Ranker* in all the measures by both criteria; for the set of TREC 2012 queries, the *SimpleMerging* algorithm consistently outperforms the *Uniform Ranker* in MAP and NDCG@30 but gets negligible deteriorations in P30 by both criteria. For all the measures with respect to the two sets of TREC queries, the *GreedyMerging* algorithm consistently outperforms the *Uniform Ranker* baseline by both relevant criteria. This observation validates that the retrieval effectiveness of tweets benefits from the employment of the divide-and-conquer strategy for handling the structural difference of tweets. Third, the *GreedyMerging* algorithm consistently outperforms the *SimpleMerging* algorithm in all the measures by both relevant criteria with respect to the two sets of queries. This indicates the performance of the *GreedyMerging* algorithm is superior to that of the *SimpleMerging* algorithm.

6.1.4. The Efficiency of GreedyMerging Algorithm. To merge a ranking of m T-tweets and a ranking of n TU-tweets, the time complexity of the *GreedyMerging* algorithm is $O((m+n)^2)$. It consists of the construction of the preference matrix M and the merging process based on M. Compared with the *SimpleMerging* algorithm, the *GreedyMerging* algorithm is not very efficient when m and n are large. However, its quadratic time complexity should not be problematic when only merging the top m' T-tweets and the top n' TU-tweets, where $m' \ll m$ and $n' \ll n$. Merging the top m' T-tweets and the top n' TU-tweets makes the construction of the preference matrix efficient, since we only construct the submatrix based on these top tweets. It also makes the merging process



(a) MAP by the Relevant Criterion (b) P30 by the Relevant Criterion (c) NDCG@30 by the Relevant Criterion

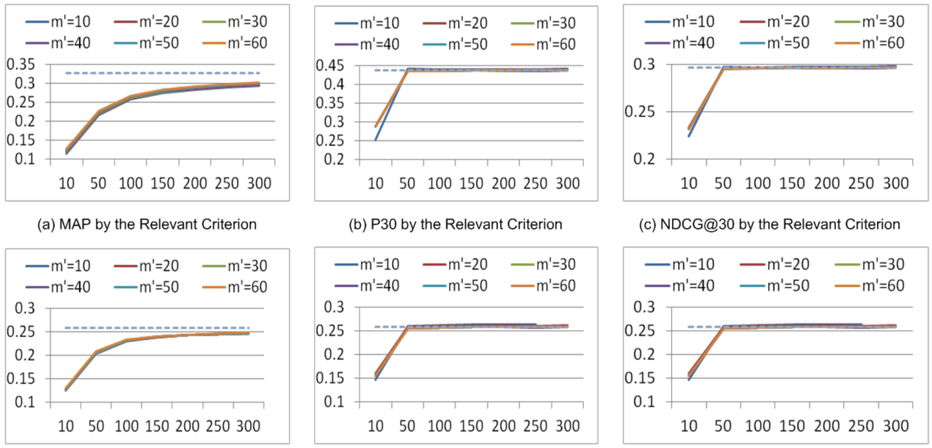
(d) MAP by the Highly Relevant Criterion (e) P30 by the Highly Relevant Criterion (f) NDCG@30 by the Highly Relevant Criterion

Fig. 3. Performance of *GreedyMerging* with the Varying Values of m' and n' with respect to the TREC 2011 queries.

efficient, because the *GreedyMerging* algorithm merges tweets by their *Dispreferness* that now is calculated on this small submatrix.

We study the effectiveness of the *GreedyMerging* algorithm when m' and n' are assigned small values. Figures 3 and 4 show the MAP, P30 and NDCG@30 performance of the *GreedyMerging* algorithm with varying small values of m' and n' for both sets of TREC 2011 and TREC 2012 queries, respectively. For the TREC 2011 queries, the value of m' varies from 10 to 60 and that of n' varies from 10 to 150. For the TREC 2012 queries, the value of m' varies from 10 to 60 and that of n' varies from 10 to 300. In all the component figures of Figures 3 and 4, the x axes represent the varying values of n' and the y axes represent the MAP, P30 and NDCG@30 performance by either the relevant criterion or the highly relevant criterion. The different curves represent the varying values of m'. The dash lines represent the corresponding performance of the *GreedyMerging* algorithm by merging all m T-tweets with all n TU-tweets. For ease of presentation, let us denote as *FullGreedyMerging* the *GreedyMerging* algorithm that merges all m T-tweets and all n TU-tweets.

Figure 3 shows the performance of the *GreedyMerging* algorithm by both relevant criteria with respect to the set of TREC 2011 queries. According to Figure 3(a), which shows the MAP performance by the relevant criterion, the *GreedyMerging* algorithm achieves a comparable MAP score of 0.4987 when merging only the top 60 ($m' = 60$) T-tweets and the top 150 ($n' = 150$) TU-tweets, relative to a MAP score of 0.5006 achieved by *FullGreedyMerging*. A similar observation can be made based on Figure 3(d) where the MAP performance is evaluated by the highly relevant criterion. The *GreedyMerging* algorithm achieves a comparable MAP score of 0.4017 when merging only the top 40 ($m' = 40$) T-tweets and the top 90 ($n' = 90$) TU-tweets, compared with a MAP score of 0.4090 achieved by *FullGreedyMerging*. If the users are interested in the top tweets, we can achieve comparable performance in terms of P30 and NDCG@30, when merging very few T-tweets and TU-tweets. According to Figure 3(b) (Figure 3(e)) where the P30 performance is evaluated by the (highly) relevant criterion, we can achieve a P30 score of 0.5122 (0.2263) when just merging the top 10 ($m' = 10$) T-tweets and the top 30 ($n' = 30$) TU-tweets, compared with the P30 score of 0.5143 (0.2283) achieved by *FullGreedyMerging*. Similar observations can be made based on the NDCG@30 performance shown by Figure 3(c) and Figure 3(f). This indicates that we can make the



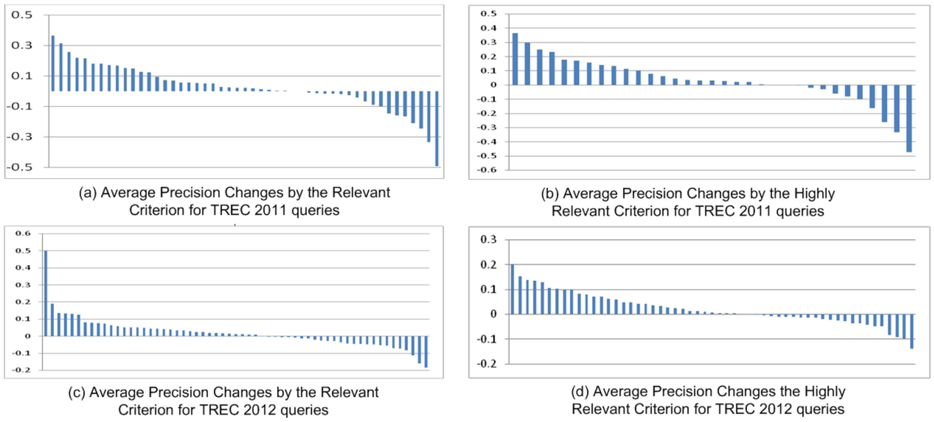
(a) MAP by the Relevant Criterion
(b) P30 by the Relevant Criterion
(c) NDCG@30 by the Relevant Criterion
(d) MAP by the Highly Relevant Criterion
(e) P30 by the Highly Relevant Criterion
(f) NDCG@30 by the Highly Relevant Criterion

Fig. 4. Performance of *GreedyMerging* with the Varying Values of m' and n' with respect to the TREC 2012 queries.

GreedyMerging algorithm much more efficient without significantly hurting its effectiveness for the TREC 2011 queries.

Figure 4 shows the performance of the *GreedyMerging* algorithm by both relevant criteria with respect to the set of TREC 2012 queries. As shown in Figure 4(a) (Figure 4(d)) where the MAP performance is evaluated by the (highly) relevant criterion, the *GreedyMerging* algorithm achieves a reasonable MAP score of 0.3013 (0.2476) by merging only the top 60 ($m' = 60$) T-tweets and the top 300 ($n' = 300$) TU-tweets, relative to a MAP score of 0.3256 (0.2590) achieved by *FullGreedyMerging*. We note that the TREC 2012 queries are harder than the TREC 2011 queries to achieve good performance, which explains why we only achieve reasonable MAP performance for the TREC 2012 queries by merging more top tweets than the TREC 2011 queries. If only the top tweets are interested by users, we can achieve comparable performance in P30 and NDCG@30 by merging very few top T-tweets and top TU-tweets. In particular, according to Figure 4(b) (Figure 4(e)) where the P30 performance is evaluated by the (highly) relevant criterion, the *GreedyMerging* algorithm achieves a comparable P30 score of 0.4340 (0.2571) by merging only the top 10 ($m' = 10$) T-tweets and the top 30 ($n' = 30$) TU-tweets, relative to the P30 score of 0.4367 (0.2583) achieved by *FullGreedyMerging*. Similar observations can be made based on the NDCG@30 performance shown in Figure 4(c) and Figure 4(f). All these observations indicate that the *GreedyMerging* algorithm can be much more efficient by achieving reasonable MAP performance and comparable P30 and NDCG@30 performance for the TREC 2012 queries.

6.1.5. Result Analysis. In this section, we conduct an analysis for both sets of TREC queries. Specifically, we compare the MAP performance of the *Uniform Ranker* with that of the divide-and-conquer method using the *GreedyMerging* algorithm (see Table VI). This comparison shows whether our way of handling the structural difference of tweets can improve retrieval effectiveness. We analyze the average precision (*AP* for short) changes query by query. Figure 5 shows the *AP* changes by both relevant criteria with respect to the two sets of queries. For example, Figure 5(a) shows the *AP* changes for the TREC 2011 queries by the relevant criterion. The changes are displayed from the most improved query (on the left) to the most deteriorated query (on the right). This displaying style continues from Figure 5(b) to 5(d). According to Figure 5, our



(a) Average Precision Changes by the Relevant Criterion for TREC 2011 queries

(b) Average Precision Changes by the Highly Relevant Criterion for TREC 2011 queries

(c) Average Precision Changes by the Relevant Criterion for TREC 2012 queries

(d) Average Precision Changes the Highly Relevant Criterion for TREC 2012 queries

Fig. 5. AP Changes of the TREC 2011-2012 queries (*Uniform Ranker* vs. Divide-and-Conquer Method).

proposed method based on the divide-and-conquer strategy can improve the majority of the queries by both relevant criteria for the two sets of queries. This validates the effectiveness of our method.

We perform a deeper analysis of our results to find out how many queries are significantly improved or hurt in their *AP*s ($\Delta AP \geq 0.1$) by our method and discuss the corresponding reasons. For the TREC 2011 queries, Figure 5(a) shows that 13 queries are significantly improved in their *AP*s while 7 queries are significantly hurt according to the relevant criterion. Figure 5(b) shows that 10 queries are significantly improved while 4 queries are significantly hurt according to the highly relevant criterion. For the TREC 2012 queries, Figure 5(c) shows that 6 queries are significantly improved in their *AP*s while 3 queries are significantly hurt according to the relevant criterion. Figure 5(d) shows that 9 queries are significantly improved while only 1 query is significantly hurt according to the highly relevant criterion.

One reason why our method improves some queries in their *AP*s is that the *T-tweet Ranker* (see Section 6.1.1) outperforms the *Uniform Ranker* in ranking T-tweets for them. Let us illustrate this reason with an example.

Example 6. The query q = "*Assange Nobel peace nomination*" has four concepts: two *PN* concepts, "*Assange*" and "*Nobel peace*," and two *CNP* concepts, "*Nobel peace nomination*" and "*Assange Nobel peace nomination*." Given a T-tweet d_1 = "*Nobel war prize for wikileaks... only if the nukes are fired... #cablegate #wikileaks #assange #anonymous*" and another T-tweet d_2 = "*#unlikelyheadlines GEORGE BUSH WINS NOBEL PEACE PRIZE! Ha*," d_1 is relevant to q while d_2 is irrelevant to q. The *T-tweet Ranker* ranks d_1 on top of d_2, because its most important feature is "the percentage of related nouns of the query contained in the tweet message field" (see Table V). d_1 contains one related noun, "*wikileaks*", while d_2 does not contain any related nouns. The merged ranking preserves the ranking of d_1 ahead of d_2. However, the *Uniform Ranker* ranks d_2 above d_1. For the most important feature of the *Uniform Ranker*, "the percentage of query concepts contained in the tweet message field" (see Table V), d_1 contains a *PN* concept, "*Assange*" in its message field and d_2 contains another *PN* concept "*Nobel peace*" in its message field too. d_1 and d_2 are tied. The second most important feature of the *Uniform Ranker*, "the percentage of query concepts contained in the URL title field" (see Table V), is not applicable for T-tweets. For the third most important feature of the *Uniform Ranker*, "the weighted percentage of query concepts contained in the tweet message field" (see Table V), d_2 beats d_1, because the weight of "*Nobel peace*" is higher than that of "*Assange*." We use the inverse document frequency of a concept as its weight. There are more tweets containing "*Assange*" than the tweets containing "*Nobel peace*" in our collection. Therefore, the *Uniform Ranker* ranks d_2 above d_1.

Another reason why our method improves some queries in their *AP*s is that the *TU-tweet Ranker* (see Section 6.1.1) outperforms the *Uniform Ranker* in ranking TU-tweets for them. Let us illustrate this reason with an example.

Example 7. The query q = "*Supreme Court cases*" has two concepts, a *PN* concept "*Supreme Court*" and a *CNP* concept "*Supreme Court cases*". Given a TU-tweet d_1 = "*@enrogers Only FOX news... http://www.foxnews.com/opinion/2010/01/22/ken-klukowski-supreme-court-amendment-mccain-feingold/*" and another TU-tweet d_2 = "*Letter to Julia Gillard by Peter H Kemp - Solicitor of the Supreme Court of NSW http://wlcentral.org/node/1175 #assange #wikileaks*," d_1 is relevant while d_2 is irrelevant. The *Uniform Ranker* ranks d_2 on top of d_1, because its most important feature is "the percentage of query concepts contained in the tweet message field" (see Table V). d_1 has no query concept in its message field while d_2 has a query concept "*Supreme Court*" in its message field. The *TU-tweet Ranker* ranks d_1 above d_2, because its most important feature is "the weighted percentage of query concepts contained in the union of all three fields" (see Table V). The Web page linked by the URL in d_1 contains both query concepts, "*Supreme Court*" and "*Supreme Court cases*" while the Web page linked by the URL in d_2 only contains "*Supreme Court*". The merged ranking preserves the ranking of d_1 ahead of d_2.

The *T-tweet Ranker* and the *TU-tweet Ranker* are superior to the *Uniform Ranker* in ranking T-tweets and TU-tweets. Our merging algorithm preserves most of the preferences indicated by those two tweet type-specific rankers. So our divide-and-conquer method improves the majority of the queries.

The reason why some queries suffer significant drops in their *AP*s is that the *TU-tweet Ranker* falsely ranks some irrelevant TU-tweets over some relevant TU-tweets with respect to them. This happens for a small minority of queries, because the *TU-tweet Ranker* is not perfect. Let us illustrate this reason with a query "*Michelle Obama fashion*" whose performance is hurt most by both relevant criteria among the TREC 2012 queries.

Example 8. The query q = "*Michelle Obama fashion*" has two concepts, a *PN* concept, "*Michelle Obama*" and a *CNP* concept, "*Michelle Obama fashion*". Given a TU-tweet d_1 = "*Michelle Obama & Jill Biden Coordinate With Pearls On Monday (PHOTOS, POLL) http://huff.to/h4PmSg*" and a TU-tweet d_2 = "*Fashionista: Fashion News Roundup: Franca Sozzani Trashes Fashion Bloggers, Cathy Horyn Throws Down Over Michell... http://bit.ly/fTKBEs*," d_1 is relevant but d_2 is irrelevant. The Web page linked by the URL in d_1 presents the following excerpt: "*And the pair did a little co-ordinating of their own – blazers and pearls. Michelle opted for a gray suit, with a necklace secured with a safety pin (super punk rock!), while Jill mixed cream with metallics and long necklace strands.*". This excerpt implicitly talks about the fashion aspect of "*Michelle Obama*", although the query term "fashion" does not occur at all. However, consider an excerpt from the Web page linked by the URL in d_2, "*Fashion News Roundup: Franca Sozzani Trashes Fashion Bloggers, Cathy Horyn Throws Down Over Michelle Obamas McQueen, and Naomi Campbells a No-Show in Court*". This excerpt contains all the query terms that form a *CNP* concept but is irrelevant to the query. The *TU-tweet Ranker* ranks d_2 on top of d_1, because its most importance feature is "the weighted percentage of query concepts contained in the union of all three fields" (see Table V). d_2 contains all the query concepts in the union of its three fields while d_1 only has a query concept, "*Michelle Obama*" in the union of its three fields. The ranking of d_2 ahead of d_1 is preserved in the merged ranking. However, the *Uniform Ranker* ranks d_1 above d_2, because its most important feature is "the percentage of query concepts contained in the tweet message field" (see Table V). d_1 has a query concept "*Michelle Obama*" in its message field while d_2 does not have any query concepts in its message field.

Again our merging algorithm can preserve most of the preferences of TU-tweets indicated by the *TU-tweet Ranker*. Given a query q, if its *AP* performance of TU-tweets achieved by the *TU-tweet Ranker* is significantly deteriorated, compared with that of the *Uniform Ranker*, q suffers a significant drop in the *AP* performance of its merged ranking.

6.2. Improving Relevance Ranking via Temporal Information

In this section, we present a set of experiments to evaluate our method that handles temporality. Specifically, we first validate our proposed three temporal categories of queries. Then we evaluate our proposed F-measure aggregation method (see Section 4.3) by comparing it with two baseline aggregation methods, combSUM and combMNZ [Shaw et al. 1994]. Third, we show that the incorporation of the temporal information of tweets can further improve the retrieval effectiveness of the divide-and-conquer method in the first phase. Finally, we present a result analysis and discuss the reasons why our way of using temporality helps or hurts some queries.

6.2.1. The Validation of Temporal Query Categorizations. In this section, we validate our three temporal categories of queries. We conduct the experiments in three scenarios where queries are classified into either time insensitive ones or time sensitive ones. In the first scenario, a classified time sensitive query is always treated as a dominant peak query, no matter how its top tweets are temporally distributed. The time-related relevance scores of the tweets with respect to it are calculated by the Laplace-like function given by Equation (11). In the second scenario, a classified time sensitive query is always treated as a nondominant peak query, irrespective of the temporal distribution of its top tweets. The time-related relevance scores of the tweets with respect to it are thus computed by Equation (13). In the third scenario, a time sensitive query is classified to be either a dominant peak query or a nondominant peak query. The time-related relevance scores of the tweets with respect to that query are calculated by Equation (11) or Equation (13), depending on its type. This is what we propose in this article. By comparing the results from the third scenario with those from the first two scenarios, we can conclude whether our temporal query categorization is necessary.

Several parameters are proposed to temporally categorize queries, so we first discuss how to estimate them, which is followed by a description of how to configure the three scenarios. There are four parameters to be estimated. They are K, p, s and β (see Equations (8) to (10) and (14)). Given a query q, we first empirically use the top K tweets of q to approximate the temporal distribution of the relevant tweets to q; then we categorize q into one of three classes, after comparing the proportion of its top K tweets at each day by p and s; we calculate the time-related relevance scores of the tweets according to the classified type of q; finally, we aggregate the IR scores of the tweets with their time-related relevance scores by β. We perform a grid search for estimating them. Specifically, K is estimated within the range from 10 to 60 at intervals of 10; the parameter p is estimated within the range from 0 to 0.5 at intervals of 0.1 to ensure $p \leq 0.5$; the parameter s is estimated within the range from $p + 0.1$ to 1 at intervals of 0.1 to ensure $s > p$; the parameter β is estimated within the range from 0 to 1 using the same interval length as p and s. For the TREC 2011 queries, we employ the TREC 2012 queries and their relevance judgments as the training data to estimate these four parameters and vice versa. We present the parameter estimation method here.

(1) Given a combination of a K value, a p value, and an s value within their ranges, the training query set Q can be categorized into three subsets of queries: Q_A (time insensitive queries), Q_B (time sensitive dominant peak queries) and Q_C (time sensitive nondominant peak queries). The class of a training query $q(\in Q)$ is determined by exploring the temporal distribution of its top K tweets (from the first phase) jointly with the p value and the s value.
(2) For the subset of the training queries, Q_A, parameter β is not estimated. Let MAP_A denote the MAP performance of the ranking of the tweets by their IR scores with respect to Q_A.
(3) For the subset of the training queries, Q_B, we iteratively test all possible values of the parameter β for Q_B. Let β_B denote this parameter β for Q_B.
    (a) For each possible value of β_B, we first calculate the time-related relevance scores (*TRS*s) of tweets with respect to Q_B by Equation (11), then aggregate the IR scores of the tweets (with respect to Q_B) with their *TRS*s by Equation (14) by using the β_B value; finally we obtain a ranking of the tweets in descending order of their aggregated scores. The performance of this ranking can be measured by a MAP score.
    (b) Find the β_B value (from Step 3.a) that corresponds to the highest MAP score. Let MAP_B denote this highest MAP score for Q_B.
(4) For the subset of the training queries Q_C, we iteratively test all possible values of β for Q_C. Let β_C denote this parameter β for Q_C. Apply a similar method to Step 3 on Q_C except that the *TRS*s of the tweets with respect to Q_C are computed by Equation (13). Find the β_C value that corresponds to the highest MAP score for Q_C (denoted by MAP_C).
(5) Union the K value, the p value and the s value from Step 1, the β_B value from Step 3, and the β_C value from Step 4 into a combination of five parameters. This combination corresponds to a MAP performance for all training queries $Q(= Q_A \cup Q_B \cup Q_c)$ that can be calculated as follows. Let MAP_Q denote this MAP score.

$$MAP_Q = \frac{MAP_A \cdot |Q_A| + MAP_B \cdot |Q_B| + MAP_C \cdot |Q_C|}{|Q_A| + |Q_B| + |Q_C|} \tag{15}$$

(6) Iteratively repeat Step 1–Step 5 with another combination of a K value, a p value and an s value until all their possible combinations are iterated. Find the combination of K, p, s, β_B and β_C that corresponds to the highest MAP_Q. This combination is the set of the estimated parameter values.

We provide some explanations for this method. Given a possible combination of a value of K, a value of p and a value of s, we find out the value of the parameter β that maximizes the MAP performance of all the training queries. Since the calculations of the time-related relevance scores for dominant peak queries and nondominant peak queries are defined differently, the parameter β for them should be estimated differently. Therefore, we technically have five parameters to estimate: K, p, s, β_B and β_C. After the parameters are estimated, we can apply them to test queries. Specifically, given the top K tweets of a test query q', we categorize q' into one of three classes by exploring the temporal distribution of the top K tweets via the estimated p value and the estimated s value. If q' is categorized to be a time insensitive query, there is no estimated parameter β for q'; if q' is categorized to be a dominant peak query, the estimated parameter β_B value is used to aggregate the IR scores of the tweets for q' with their *TRS*s; if q' is categorized to be a nondominant peak query, the estimated parameter β_C value is used to aggregate the IR scores of the tweets for q' with their *TRS*s.

Table VII. The Comparison of Three Systems

	TREC 2011					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
System I ($p = s$)	0.5062	**0.5224**	0.4983	0.4116	**0.2323**	0.4981
System II ($s = 1$)	0.5142	**0.5224**	0.5061	0.4141	0.2273	0.4962
System III	**0.5270**†‡	0.5218	**0.5076**	**0.4357**	0.2283	**0.5125**

	TREC 2012					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
System I ($p = s$)	0.3236	0.4362	0.2939	0.2564	0.2554	0.2817
System II ($s = 1$)	0.3360	0.4492	0.2933	0.2644	0.2589	0.2858
System III	**0.3415**†	**0.4695**†‡	**0.3018**	**0.2719**	**0.2738**†‡	**0.2911**

Note: † and ‡ indicate statistically significant improvements over System I and System II respectively.

Our proposed system that uses the given estimation method corresponds to the third scenario. Let System III denote the system in the third scenario. System III is compared against two systems, each having only one type of time sensitive queries.

System I is obtained by stipulating $p = s < 1$, ignoring the restrictions of $p \leq 0.5$ and $s > p$. It assumes that if a query does not satisfy Equation (8), it is time sensitive. Because Equation (10) cannot hold when $p = s$, all time sensitive queries are assumed to be the dominant peak queries, regardless of the distributions of their top tweets. If a query has multiple peaks, then the highest peak serves as the dominant peak. System I has two parameters $p(= s)$ and β that can be estimated in a similar way as discussed before. In particular, by assuming $p = s < 1$, all training queries can be partitioned into a set of time insensitive queries and a set of dominant peak queries. The combination of a p value and a β value that yields the largest MAP score for the training queries is utilized to categorize a test query q'. If q' is a time sensitive query, then the Laplace-like function is used to calculate the time-related relevance scores for the tweets for q', as this is the only type of time sensitive queries for this system. System I corresponds to the proposed system in the first scenario described earlier.

System II is configured by stipulating $s = 1$. Because Equation (9) cannot hold when $s = 1$, all time sensitive queries are assumed to be the nondominant peak queries. Equation (13) is applied to calculate the time-related relevance scores. The two parameters p and β are estimated using a similar method to that used by System I. For each test query q', the combination of a p value and a β value which yields the largest MAP score for the training queries is applied to q'. System II corresponds to the proposed system used in the second scenario. Table VII presents their performances.

As shown in Table VII, for the set of TREC 2011 queries, compared with System I, System III suffers slight deteriorations in P30 by both relevant criteria. However, System III consistently outperforms System I in MAP and in NDCG@30 by both relevant criteria. We also see that System III consistently outperforms System II in almost all measures by both relevant criteria except a negligible deterioration in P30 by the relevant criterion. For the set of TREC 2012 queries, System III consistently outperforms System I and System II in all measures by both relevant criteria. These improvements validate our temporal query categorizations.

6.2.2. The Evaluation of Aggregation Method. In this section, we evaluate our proposed aggregation method (i.e., System III in Table VII). We compare its performance with two baselines, CombSUM and CombMNZ [Shaw et al. 1994]. In particular, given a tweet d with an IR score $IR(d)$ and a time-related relevance score $TRS(d)$, the

Table VIII. The Comparison of Various Aggregations

	TREC 2011					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
CombSUM	0.5028	**0.5245**	0.4951	0.4025	**0.2394**	0.4917
CombMNZ	0.4909	0.5156	0.4851	0.3931	0.2343	0.4882
Our Aggregation	**0.5270**	0.5218	**0.5076**	**0.4357**	0.2283	**0.5125**
	TREC 2012					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
CombSUM	0.3358	0.4616	0.2915	0.2495	0.2631	0.2849
CombMNZ	0.3412	0.4638	**0.3020**	0.2660	0.2690	0.2897
Our Aggregation	**0.3415**	**0.4695**	0.3018	**0.2719**	**0.2738**	**0.2911**

CombSUM method calculates an aggregated score for d, $CombSum(d) = IR(d) + TRS(d)$; the CombMNZ method calculates an aggregated score for d, $CombMNZ(d) = CombSum(d) \cdot m_d$, where m_d is the number of nonzero scores for d. Specifically, if d has a nonzero $IR(d)$ score and a nonzero $TRS(d)$ score, then $m_d = 2$. If a query q is time insensitive, no *TRS*s are assigned to the tweets with respect to q. Table VIII shows the comparisons of the three aggregation methods. For the TREC 2012 queries, our aggregation method consistently outperforms CombSUM in all measures by both relevant criteria. Our method also outperforms CombMNZ in almost all measures by both relevant criteria except a negligible deterioration in NDCG@30 by the relevant criterion. For the TREC 2011 queries, compared with the two baselines, our method suffers marginal deteriorations in P30 by both relevant criteria. But it outperforms the two baselines in all other measures by both relevant criteria. Overall, our aggregation method shows the strongest performance among all three aggregation methods. However, the improvements over CombSUM and CombMNZ by our aggregation method are not statistically significant.

6.2.3. The Impact of Temporal Information on Retrieval Effectiveness. We now study the impact of incorporating temporal information on retrieval effectiveness. In this experiment, we use two baselines. The first baseline is our divide-and-conquer method using the *GreedyMerging* algorithm (i.e., its performance in Table VI), because we want to see whether the inclusion of temporal information can further improve the performance of this baseline or not. Let BASELINEI denote the first baseline. The second baseline is the algorithm proposed by [Efron and Golovchinsky 2011]. Given a query q with a timestamp t, this method ranks the tweets published before or on t by using their temporal information. Specifically, it prefers the recent tweets close to t to the old tweets and calculates a score $P(d|q)$ for a tweet d (publishing at t_d) by Equation (16).

$$P(d|q) \propto P(q|d) \cdot r \cdot e^{-r \cdot f(t_d, t)}, \tag{16}$$

where r is the rate parameter of the exponential distribution. $P(q|d)$ is an IR score provided by a retrieval model. $f(t_d, t)$ is the same time representation we adopt in this article. Efron and Golovchinsky [2011] proposed to do the maximum posterior estimation for the parameter r for each q as follows. Let $D_q = \{d_1, \ldots, d_k\}$ be the top k tweets for q by a ranking model. Let $T_{D_q} = \{t_1, \ldots, t_k\}$ be the set of the time representations of the publishing times associated with D_q. Then $r_q^{MAP} = \frac{\rho + k - 1}{\sigma + \sum_{i=1}^{k} t_i}$. This estimation involves three parameters, k, ρ and σ. In order to compare our method with this method (denoted by BASELINEII), we use the IR scores of the tweets from the first phase as $P(q|d)$ for BASELINEII. Moreover, we also do the maximum posterior estimation of

Table IX. The Impacts of Temporal Information

	TREC 2011					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
BASELINEI (Divide-and-Conquer Method)	0.5006	0.5143	0.4939	0.4090	**0.2283**	0.4933
BASELINEII [Efron and Golovchinsky 2011]	0.3958↓	0.3891↓	0.3909↓	0.3391↓	0.1505↓	0.4117↓
Our Method (System III)	**0.5270**†‡	**0.5218**‡	**0.5076**‡	**0.4357**†‡	**0.2283**‡	**0.5125**‡

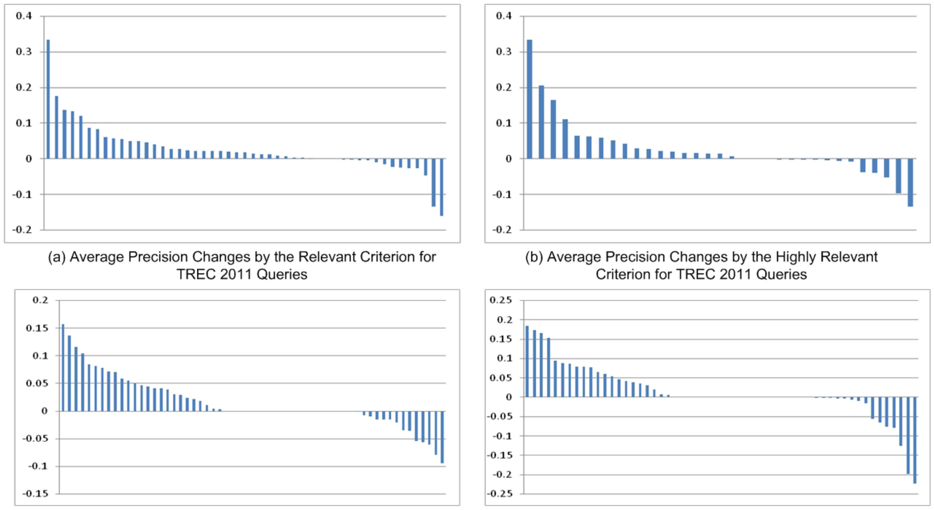
	TREC 2012					
	Relevant			Highly Relevant		
	MAP	P30	NDCG@30	MAP	P30	NDCG@30
BASELINEI (Divide-and-Conquer Method)	0.3259	0.4367	0.2966	0.2590	0.2583	0.2852
BASELINEII [Efron and Golovchinsky 2011]	0.2305↓	0.3283↓	0.2082↓	0.1698↓	0.1923↓	0.2011↓
Our Method (System III)	**0.3415**†‡	**0.4695**†‡	**0.3018**‡	**0.2719**‡	**0.2738**†‡	**0.2911**‡

Note: † and ↓ indicate statistically significant improvements and deteriorations over BASELINEI; ‡ indicates statistically significant improvements over BASELINEII.

r for each test query. Efron and Golovchinsky [2011] showed that their suggested parameter values are effective for the proposed maximum posterior estimations across two collections, including a Twitter collection. In our experiments, for each parameter, we try different values for that parameter including their suggested value. For example, for the parameter k, we try 10, 20, and 30, where 20 is their suggested value. The method using their suggested values for the three parameters k, ρ and σ achieves the best performance and thus we just report the best performance in this article. Table IX presents the comparison of our method with two different baselines.

As shown in Table IX, compared with BASELINEI, BASELINEII deteriorates in all measures by both relevant criteria for both sets of TREC queries. We are not surprised by this results, because BASELINEII always prefers recent tweets to old tweets. Such a recency-preferred strategy does not apply for our queries, as we discussed in Section 1, which is the reason why we propose our three temporal categorizations of queries. For comparing our method with BASELINEI with respect to the set of TREC 2011 queries, our method ties with BASELINEI in P30 by the highly relevant criterion but outperforms BASELINEI in all other measures by both relevant criteria. For the set of TREC 2012 queries, it consistently achieves improvements over BASELINEI in all measures using both relevant criteria. This demonstrates that our proposed method using temporality can effectively further improve the retrieval effectiveness of our divide-and-conquer method in the first phase. Our method also consistently and statistically significantly outperforms BASELINEII in all measures by both relevant criteria for the two sets of queries, which demonstrates the effectiveness of our proposed temporality usage.

6.2.4. Result Analysis. In this section, we conduct an analysis for our utilization of the temporal information of tweets. In particular, we do a query-by-query analysis by comparing the MAP performance of BASELINEI with that of our method (see Table IX). Figure 6 shows the average precision (*AP* for short) changes for the TREC 2011 and 2012 queries by both relevant criteria. It displays the changes from the most improved query to the most deteriorated query. According to Figure 6, our usage of temporality improves the average precisions for the majority of the TREC 2011 and 2012 queries. This demonstrates the effectiveness of our proposed method.



(a) Average Precision Changes by the Relevant Criterion for TREC 2011 Queries

(b) Average Precision Changes by the Highly Relevant Criterion for TREC 2011 Queries

(c) Average Precision Changes by the Relevant Criterion for TREC 2012 Queries

(d) Average Precision Changes by the Highly Relevant Criterion for TREC 2012 Queries

Fig. 6. The average precision changes for temporality.

We provide a deeper analysis to see how many queries are significantly improved or hurt ($\Delta AP > 0.05$) in their *AP*s by our temporality usage and present the reasons why their *AP*s are improved or deteriorated. For the TREC 2011 queries, 10 (8) queries are significantly improved while 2 (3) queries are significantly hurt according to the (highly) relevant criterion. For the TREC 2012 queries, 11 (13) queries are significantly improved while 5 (7) queries are significantly hurt according to the (highly) relevant criterion. Let us illustrate the reasons with two examples. For example, the query q = "*Aguilera super bowl fail*" is a dominant peak query. 73% of its relevant tweets were published on 2/7/2011. The first phase of our method achieves a very good precision at the top tweets for q, as 26 out of the top 30 tweets are relevant to q. Then our method correctly classifies q as a dominant peak query and predicts its dominant peak is on 2/7/2011. This query is significantly improved in its *AP*. On the contrary, if the first phase of our method fails to achieve a decent precision by its top tweets with respect to a query q, then our classification of q is inaccurate, which may lead to a deterioration in the *AP* of q after we apply our temporal method on q. For example, for the query "*Michelle Obama fashion*," we are not surprised to see a significant performance drop for this query again, because the first phase of our method achieves a poor precision at its top tweets, as 10 out of the top 30 tweets are relevant to q. Our classification for this query and our prediction of the peaks of this query are inaccurate. Overall, the performance of our usage of temporal information depends on an accurate classification of each query, which in turn depends on how well its top tweets using the first phase of our method approximate its relevant tweets.

6.3. Comparison with Related Works

In this section, we compare the performance of our method with those of some related works. TREC 2011 required the retrieved tweets to be ordered in reverse-chronological order [Ounis et al. 2011]. In this experiment, we evaluate the performance of our two-phase method in ranking tweets in reverse-chronological order. Since our method mainly aims at ranking tweets in terms of relevance, we adopt a simple strategy to produce the reverse-chronological ranking of tweets. In particular, we take the top 30

Table X. Comparison of Our Method vs. State-of-the-Art Methods with Respect to the TREC 2011 Queries

	Relevant		Highly Relevant	
	Reverse Chronological Order			
	P30	MAP	P30	MAP
[Liang et al. 2012]	0.4177	0.2365	0.1979	0.2722
[Choi et al. 2012]	0.5068	**0.3068**	-	-
Our Method	**0.5218**	0.3018	**0.2283**	**0.3189**
	Descending Order of Relevance			
	P30	MAP	P30	MAP
[Amati et al. 2012]	-	0.3950	-	-
Our Method	0.5218	**0.5270**	0.2282	0.4357
	Descending Order of Relevance Top 100 Tweets			
	P30	MAP	P30	MAP
[Efron et al. 2012]	-	0.2350	-	-
Our Method (Top 100)	0.5218	**0.4892**	0.2282	0.4262

tweets and rearrange them in reverse order of time. This strategy is the most popular strategy adopted by the participants in TREC 2011 [Ounis et al. 2011]. The primary evaluation measure is P30 for the reverse-chronological ranking of tweets. Metzler and Cai [2011] achieved the best P30 score in TREC 2011 but their results were obtained in the absence of TREC relevance judgments as training data. So we omit the comparison of our results with theirs, because we use TREC relevance judgments as training data. We compare our results with other published results with respect to the set of TREC 2011 queries. Liang et al. [2012] achieved improvements over the results of Metzler and Cai [2011] only by the highly relevant criterion. Moreover, [Choi et al. 2012] only reported their performance by the relevant criterion and their results outperform the TREC 2011 best results. Some studies [Amati et al. 2012; Efron et al. 2012] reported their MAP performance by ranking tweets in descending order of relevance to the TREC 2011 queries, without addressing the requirement of the reverse chronological order. We compare our results with these published results in Table X. As shown in Table X, our method consistently and significantly outperforms the results from Liang et al. [2012] in terms of P30 and MAP by both relevant criteria. According to the primary evaluation measure P30, our results outperform theirs by 24.9% using the relevant criterion and by 15.4% using the highly relevant criterion. Both works explore the Web pages whose URLs are embedded in tweets. According to the primary measure P30, our results outperform the results from Choi et al. [2012] by the relevant criterion and obtains a competitive performance in MAP. For ranking tweets in descending order of relevance, our results also significantly outperform the results from Amati et al. [2012] and Efron et al. [2012]. Their results were obtained without exploring the Web pages linked by tweets while our results use the information from those Web pages. Efron et al. [2012] reported their results by only evaluating top 100 tweets with respect to a given query.

For the set of TREC 2012 queries, we compare our results with the best known results reported by the TREC 2012 overview paper [Soboroff et al. 2012]. Unlike TREC 2011, TREC 2012 required tweets to be ranked in descending order of relevance, instead of reverse chronological order. Moreover, TREC 2012 only evaluated up to top 1000 tweets by the highly relevant criterion. The "*hitURLrun3*" run from [Han et al. 2012] achieved the best P30 and MAP scores [Soboroff et al. 2012]. For the TREC 2012 participants, the relevance judgments with respect to the TREC 2011 queries are

Table XI. Comparison of Our Method vs. Best Results with Respect to the TREC 2012 Queries

	TREC 2012	
	Highly Relevant	
	MAP	P30
hitURLrun3 [Han et al. 2012]	0.2640	0.2701
Our Method	**0.2719**	**0.2738**

available as training data, so we compare our corresponding results with the reported best results. Since TREC 2012 required tweets to be ranked in descending order of relevance, MAP is more important than P30. Table XI shows that our results compare favorably with the best results in both measures. Both methods use the Web pages whose links are provided by tweets.

7. CONCLUSION AND FUTURE WORK

In this article, we studied the problem of real-time ad-hoc retrieval of tweets introduced by TREC 2011. We proposed a two-phase approach to retrieve tweets. Motivated by the observation that tweets have different structures where one type of tweets contains just short plain messages (called T-tweets) and the other type of tweets contains short messages with at least one embedded URL (called TU-tweets), we proposed a divide-and-conquer based method for the first phase. Specifically, the method consists of two tweet type-specific rankers and a classifier. We first used the two rankers to obtain a ranking of T-tweets and a ranking of TU-tweets. Then we utilized the classifier to determine a preference for every two tweets, one from each type. Finally, we proposed a greedy algorithm to merge the two type-specific rankings into a single ranking for both types of tweets. The merging process takes into consideration all the preferences from the two rankers and the classifier. Experiments showed that our proposed method yields better retrieval effectiveness than the ranker that ranks the two types of tweets simultaneously. We also showed how our method can be made efficient by performing a merging of only the top tweets. In the second phase, we proposed to classify temporal queries by the temporal distributions of their top tweets and calculate the time-related relevance scores of the tweets with respect to different classes of queries accordingly. A ranking of tweets is produced by combining their IR scores from the first phase with their time-related relevance scores. Experimental results demonstrated that the utilization of the temporal information can further improve the retrieval effectiveness of the first phase. Our method is also compared favorably with some state-of-the-art methods.

For future work, we plan to investigate whether we can further improve the performance of the divide-and-conquer method by the social aspects of tweets. Such information can be found in the JSON version of the TREC Tweets2011 collection. We also plan to study other categorizations of queries, such as cyclic queries and trending queries.

REFERENCES

Ailon, N., Charikar, M., and Newman, A. 2008. Aggregating inconsistent information: Ranking and clustering. *J. ACM 55*, 5, 23:1–23:27.

Amati, G., Amodeo, G., and Gaibisso, C. 2012. Survival analysis for freshness in microblogging search. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12)*. ACM, New York, 2483–2486.

Amodeo, G., Amati, G., and Gambosi, G. 2011. On relevance, time and query expansion. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM'11)*. ACM, New York, 1973–1976.

Berberich, K., Bedathur, S., Alonso, O., and Weikum, G. 2010. A language modeling approach for temporal information needs. In *Proceedings of the 32nd European conference on Advances in Information Retrieval (ECIR'10)*. 13–25.

Bian, J., Li, X., Li, F., Zheng, Z., and Zha, H. 2010. Ranking specialization for web search: A divide-and-conquer approach by using topical ranksvm. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*. 131–140.

Choi, J. and Croft, W. B. 2012. Temporal models for microblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12).* ACM, New York, 2491–2494.

Choi, J., Croft, W. B., and Kim, J. Y. 2012. Quality models for microblog retrieval. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12).* ACM, New York, 1834–1838.

Cohen, W. W., Schapire, R. E., and Singer, Y. 1998. Learning to order things. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS'97)*. 451–457.

Dai, N. and Davison, B. D. 2010. Freshness matters: In flowers, food, and web authority. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'10)*. 114–121.

Dai, N., Shokouhi, M., and Davison, B. D. 2011. Learning to rank for freshness and relevance. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11).* 95–104.

Dakka, W., Gravano, L., and Ipeirotis, P. G. 2012. Answering general time-sensitive queries. *IEEE Trans. Knowl. Data Eng. 24*, 220–235.

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'10)*. 11–20.

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 (WWW'10).* 331–340.

Duan, Y., Jiang, L., Qin, T., Zhou, M., and Shum, H.-Y. 2010. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*. 295–303.

Efron, M. and Golovchinsky, G. 2011. Estimation methods for ranking recent information. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11).* 495–504.

Efron, M., Organisciak, P., and Fenlon, K. 2012. Improving retrieval of short texts through document expansion. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'12).* ACM, New York, 911–920.

Elsas, J. L. and Dumais, S. T. 2010. Leveraging temporal dynamics of document content in relevance ranking. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM'10).* 1–10.

Han, Z., Li, X., Yang, M., Qi, H., Li, S., and Zhao, T. 2012. Hit at trec 2012 microblog track. In *Proceedings of Text REtrieval Conference*.

Herbrich, R., Graepel, T., and Obermayer, K. 2000. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, P. J. Bartlett, B. Schölkopf, D. Schuurmans, and A. J. Smola Eds., 115–132.

Hüllermeier, E. and Fürnkranz, J. 2010. On predictive accuracy and risk minimization in pairwise label ranking. *J. Comput. Syst. Sci. 76*, 1, 49–62.

Joachims, T. 1999. *Advances in Kernel Methods*. 169–184.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*. 133–142.

Jones, R. and Diaz, F. 2007. Temporal profiles of queries. *ACM Trans. Inf. Syst. 25*, 3.

Keikha, M., Gerani, S., and Crestani, F. 2011a. Temper: A temporal relevance feedback method. In *Proceedings of the 33d European Conference on Advances in Information Retrieval (ECIR'11)*. Springer, 436–447.

Keikha, M., Gerani, S., and Crestani, F. 2011b. Time-based relevance models. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11).* ACM, New York, 1087–1088.

Kulkarni, A., Teevan, J., Svore, K. M., and Dumais, S. T. 2011. Understanding temporal query dynamics. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM'11)*. 167–176.

Laplace, P.-S. 1774. Mémoire sur la probabilité des causes par les évènements. Mémoires de l'Academie Royale des Sciences Presentés par Divers Savan., 621–656.

Lee, J. H. 1997. Analyses of multiple evidence combination. In *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97).* ACM, New York, 267–276.

Li, X. and Croft, W. B. 2003. Time-based language models. In *Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM'03).* 69–475.

Liang, F., Qiang, R., and Yang, J. 2012. Exploiting real-time information retrieval in the microblogosphere. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'12)*. 267–276.

Liu, S., Liu, F., Yu, C., and Meng, W. 2004. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04).* 266–272.

Massoudi, K., Tsagkias, M., de Rijke, M., and Weerkamp, W. 2011. Incorporating query expansion and quality indicators in searching microblog posts. In *Proceedings of the 32nd European conference on Advances in Information Retrieval (ECIR'10).* Springer, 362–367.

McCreadie, R., MacDonald, C., Santos, R., and Ounis, I. 2011. University of glasgow at trec 2011: Experiments with terrier in crowdsourcing, microblog, and web tracks. In *Proceedings of Text REtrieval Conference*.

Metzler, D. and Cai, C. 2011. Usc/isi at trec 2011: Microblog track (notebook version). In *Proceedings of Text REtrieval Conference*.

Ounis, I., MacDonald, C., Lin, J., and Soboroff, I. 2011. Overview of the trec 2011 microblog track. In *Proceedings of Text REtrieval Conference*.

Rijsbergen, C. J. V. 1979. *Information Retrieval* 2nd Ed. Butterworth-Heinemann, Newton, MA.

Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M., and Gatford, M. 1996. Okapi at TREC-3. 109–126.

Robertson, S., Zaragoza, H., and Taylor, M. 2004. Simple bm25 extension to multiple weighted fields. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management (CIKM'04).* 42–49.

Shaw, J. A., Fox, E. A., Shaw, J. A., and Fox, E. A. 1994. Combination of multiple searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC-2)*. 243–252.

Soboroff, I., Ounis, I., and Lin, J. 2012. Overview of the trec 2012 microblog track. In *Proceedings of Text REtrieval Conference.*

Zhang, W., Liu, S., Yu, C., Sun, C., Liu, F., and Meng, W. 2007. Recognition and classification of noun phrases in queries for effective retrieval. In *Proceedings of the 16th ACM International Conference on Information and Knowledge Management (CIKM'07).* ACM, New York, 711–720.

Zhang, X., He, B., Luo, T., and Li, B. 2012. Query-biased learning to rank for real-time twitter search. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12)*. ACM, New York, 1915–1919.

Received September 2012; revised April 2013; accepted July 2013