TREC 2011 Microblog Track Experiments at Kobe University

Taiki Miyanishi Naoto Okamura Xiaoxi Liu Kazuhiro Seki Kuniaki Uehara

Graduate School of System Informatics
Kobe University

miyanishi@ai.cs.kobe-u.ac.jp

Abstract: This paper describes our approach to real-time microblog search that returns tweets for a given query in reverse chronological order. The approach utilizes a learning-to-rank (L2R) algorithm that has been increasingly used for information retrieval (IR). Generally, L2R algorithms require features which represent the associations between a user query and a document (tweet in this case). However, it is more difficult for microblog search to obtain rich features than traditional document search because the contents of microblog are too short: limited to only 140 characters. In addition, there is no standard, publicly available training data for learning to rank microblogs. To solve these problems, we generate new features by clustering large microblog data (the Tweets2011 corpus). The features are defined for triplets ⟨user query, tweet, cluster⟩ and represent the relevance of the tweet with respect to both the query and its topic (cluster). An L2R model is learned using the generated features as well as other features on labeled training data manually created by our research group. The effectiveness of the proposed approach is demonstrated by comparative experiments.

1 Introduction

Microblog, most notably Twitter[1], is becoming increasingly popular and has been used world-wide, where *real-time search* is an important function to have a grasp of latest development or others' thoughts of a topic in which a user is interested. Such information play an important role in the field of event detection [10], real-time web search [4], or even safety information mining [9]. The difference between general web search and real-time search is that the former would return the results ranked by topical relevance to a user query, while the latter considers time sequence as well and presents relevant results in reverse chronological order (i.e., from the latest to the earliest).

The first year of the Microblog track addresses the *real-time adhoc task*. The following is the excerpt defining the task from the Microblog track guideline:

> In the first run of the Microblog track, we will be addressing a search task whereby a user's information need will be represented by a query at a specific time. In particular, we address a real-time search task, where the user wishes to see the most recent but relevant information to the query. Hence, the system should answer a query by providing a list of relevant tweets ordered from newest to oldest, starting from the time the query was issued. When selecting tweets to include in the list, systems should favor "interesting" but "newer" relevant tweets. Interestingness is subjective, but the issuer of a query might interpret it as providing somehow added value with respect to the query topic. For this year, the "novelty" between tweets will not be considered.

Our approach to this task consists of three steps: initial search, reranking, and filtering described in Figure 1. First, we index tweets and search them using the Indri search engine [8]. Second, in order to remove irrelevant tweets to this task, we filter them by http status codes and their languages according to the track guideline in this year. After filtering, we collect features for the retrieved tweets. Also, we generate another type of features, called semantic features, based on user clusters identified in the tweets corpus. Using these features, we train a Ranking SVM which was used for reranking the initial set of retrieved tweets to produce the final tweet list. Inside our system, we mainly focus on a feature generation for a learning to rank (L2R) algorithm which exploits tweet contents and authority features following the related work. Note that, it is important to have good features to be effective for L2R and, unlike other targets such as web pages, it is difficult to define such features from the contents of microblogs (tweets) as each post is limited to only 140 characters. We tackle this problem by generating new content-based features to represent the relevance of a tweet to a given query. To do this, we first cluster a large tweet corpus (Tweets2011) and then calculate a trigonal area for each triplet ⟨query, tweet, cluster⟩ in a

[1] http://twitter.com

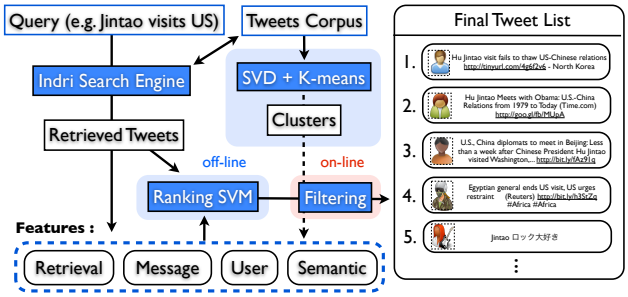

Figure 1: Overall system architecture

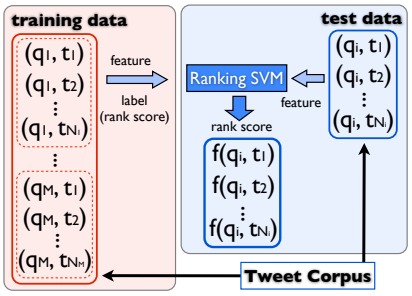

Figure 2: Ranking SVM

latent semantic space. In this representation, the relevance of a tweet to a given query is represented via each topically formed cluster. In addition, to learn a L2R model, small training data are manually created using the 12 example topics that the track organizers assembled and provided to the participants.

The rest of this paper is organized as follows: Section 2 describes how we collected our data, including Tweets2011, to be used in this work. Section 3 presents the detail of our approach. We report on our experiments and the results in Section 4. Finally, Section 5 concludes this paper with a summary of findings and possible future directions.

2 Tweet Corpus, Indexing, and Initial Search

We built the Tweets2011 corpus using the twitter-corpus-tools[2] based on the 16,141,812 tweet seeds provided by the track organizers. Table 1 shows the distribution of the HTTP status codes of the results. These tweets were posted by 5,356,842 distinct users in total, of which 5,194,623 (97.0%) correspond to the status codes 200 (OK) or 302 (Found). Unlike traditional test collections, each participant's corpus is slightly different from the other participants due to the self-archiving process. It is not clear how it affects the final results but supposedly negligible.

Table 1: Number of fetched tweets

Status code	# of tweets (%)
200 (OK)	14,230,073 (88.1)
302 (Found)	1,131,329 (7.00)
403 (Forbidden)	207,373 (1.28)
404 (Not Found)	573,034 (3.54)

We created the following two types of indices based on the resulting HTML files using the Indri retrieval engine[3] with default settings. Neither stemming nor stop-word removal was employed.

1. Index containing entire corpus (I_1).
2. Index containing only the tweets posted before the specific time associated with each topic (I_2). This index was created for each topic, resulting in 50 different indices.

The former index I_1 used the entire corpus disregarding the timestamp associated with each query. In other words, this index contains future tweets that did not exist when a query was issued. Strictly speaking, this is not appropriate for real-time search because using this index means we are using future information for term weights, such as IDF values. Conversely, the latter index I_2 was created to simulate a realistic real-time search setting, where no future information is available when a query is issued. We made the same number of indices as the number of queries having different timestamps, separately.

Using these indices, we obtained three different initial search results $T_{example}$, T_{topic1}, and T_{topic2} as follows:

1. $T_{example}$: Queried the 12 example topics against I_1 and retained top 300 tweets posted before the given topic time.
2. T_{topic1}: Queried the 50 test topics against I_1 and retained top 1000 tweets posted before the given topic time.
3. T_{topic2}: Queried the 50 test topics against I_2 and retained top 1000 tweets (tweets posted before the given topic time do not exist).

The aforementioned 12 example topics (summarized in Table 2) were distributed in advance by the track organizers. For tweets contained in $T_{example}$ and T_{topic1}, we additionally obtained their JSON files as auxiliary data.

3 Proposed Approach

Our approach uses an L2R algorithm and is similar to Duan et al. [5] except that we employ filtering and propose semantic feature generation. L2R model requires training data which consists of a set of queries (q's) and, for each query, a list of tweets (t's) from the tweet corpus manually judged in relevance order. We trained Ranking SVM as an L2R model and ranked

[2] https://github.com/lintool/twitter-corpus-tools
[3] http://lemurproject.org/indri/

Table 2: Example queries.

Number	Querytime	Title	Querytweettime
Example001	Thu Feb 03 22:06:42 2011	Chavez expropriate property	33285274087723010
Example002	Mon Feb 07 19:29:22 2011	Jintao visit US	34695232985645056
Example003	Fri Feb 04 20:03:25 2011	Saleh Yemen overthrow	33616636950880256
Example004	Mon Feb 07 13:53:01 2011	Sudan independence vote	34610585857433600
Example005	Mon Feb 07 17:37:44 2011	natural disasters Australia	34667136454631424
Example006	Wed Feb 02 22:09:28 2011	Kepler discovers new planets	32923583248338944
Example007	Wed Feb 02 15:47:04 2011	Texas school robot	32827348227198976
Example008	Wed Jan 26 03:25:35 2011	State of the Union and social media	30104034627031041
Example009	Tue Jan 25 03:05:55 2011	Cavaliers record	29736694160826368
Example010	Mon Jan 24 00:40:20 2011	Sian Massey comments	29337669070749696
Example011	Fri Jan 28 22:55:25 2011	Mets, Madoff victims lawsuit	31123207859740672
Example012	Sun Feb 06 14:13:31 2011	Bjorn Qatar Masters	34253356427911168

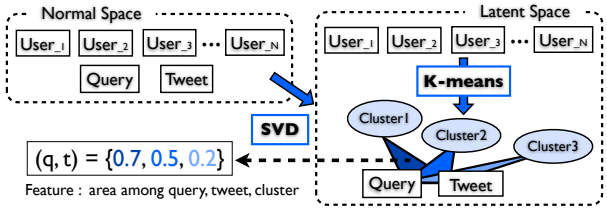

Figure 3: Feature generation system

a set of tweets in estimated relevance order. These learning and reranking of tweet steps are illustrated in Figure 2. Finally, three type filters are applied after reranking. The following sections describe the main components of our approach.

3.1 Feature Definitions

For L2R, it is quite important to choose good features. Considering the related work, we arbitrarily chose four types (called *scopes*) of features as shown in Table 3 for tweet representation. These scope types are: "Retrieval" being search scores produced by different IR models, "Message" concerning tweet itself, such as the number of characters or words and the number of URLs in the contents, "User" concerning user who posted a tweet, such as the number of followers and friends, and "Semantic" indicating the conceptual difference among a query, a tweet and a user cluster. These scopes, except for the last scope, were used for judgment of tweet credibility [2].

For binary features (i.e., true or false), their values are represented as 1 and 0. For numerical features, they are converted to z-scores. Missing values are filled with the mean of the normalized values of the other features in the same scope.

3.2 Feature Generation

Generally, L2R requires features representing the relationship between a user query and a tweet to generalize the model to unseen queries. A commonly used feature in this regard is to specify whether the query appears in a document (tweet). However, since each tweet is limited to 140 characters and there may be synonyms and polysemous words, simply looking at surface matches may not be sufficient to generate a reliable feature. For these reasons, we use K-means to identify semantic clusters representing topics such as politics, sport, science, etc. and calculate the semantic similarity among query, tweet, and a semantic cluster to create semantic features. To calculate the similarity between query and tweet in a latent concept space by the word vector projection, we use the view of Latent Semantic Indexing (LSI) [3]. As the latent semantic space divides multiple topics into respective clusters, we can calculate semantic similarities corresponding to each topic. An example of the semantic feature generation is shown in Figure3.

As the first step of making semantic clusters, we build a user-word matrix with the tf.idf[4] term weighting [7] using the entire Tweets2011 corpus. We remove infrequent words (less than three) and non-ASCII characters and apply stemming by Snowball[5]. The resulting matrix has n=4,654,678 rows (users) and m=879,192 columns (words). We apply singular value decomposition (SVD) to perform dimensionality reduction and noise reduction on the high-dimensional user-word matrix. Specifically, we use redsvd[6] to perform SVD to reduce it to p dimensional space.

For clustering, we use the mini-batch K-means [11] implemented as sofia-kmeans[7] using the reduced user-topic matrix $U_{n\times p}$ as input. Specifically, we initialize the matrix by K-means++ [1] with mini-batches of size 1000 and 10000 iterations and cluster the data into k clusters. The centroid of each cluster $\vec{c}_p$ can be seen as a representative vector for the cluster. We call this centroid a *topic vector*.

Thus, we calculate a trigonal area among the topic vector $\vec{c}_p$ determined by clustering tweets, the query vector $\vec{q}_p$ and the tweet vector $\vec{t}_p$ (the latter two are obtained by projecting their original word vectors on

[4]A user is regarded as a document d in terms of idf.
[5]http://snowball.tartarus.org/
[6]http://code.google.com/p/redsvd/
[7]http://code.google.com/p/sofia-ml/wiki/SofiaKMeans

Table 3: Five scopes of features representing a tweet.

Scope	Feature
Retrieval	Cosine similarity between query and topic
	TFIDF
	Okapi
	Language Model with Dirichlet Smoothing
	Language Model with Jelinek-Mercer Smoothing
Message	Length of the text of the tweet in characters
	Length of the text of the tweet in number of words
	Contains a question mark "?" or exclamation mark "!"
	Contains a personal pronoun in 1st, 2nd, or 3rd person. (Three features)
	Fraction of capital letters in the tweet
	Number of URLs contained on the tweet
	Tweet contains "RT"
	Day of the week in which this tweet was written
User	Time passed since the author registered his/her account in days
	Number of people following this author at posting time
	Number of people this author is following at posting time
	Whether or not the author has a verified account at posting time
	Whether or not the author has a non-empty bio at posting time
	Whether or not the author has a non-empty homepage URL at posting time
	Fraction of tweets containing more than 30% of characters in uppercase
Semantic	Area among query, tweet, and cluster in a semantic space

Table 4: Examples of the manually created training data.

Query	Relevance	Interestingness	Tweet
Chavez expropriate property	3	3	Venezuela's Chavez threatens to seize bank:/EFE)President Hugo Chavez threatened to expropriate the Venezuelan... http://bit.ly/dQJFZT
	1	2	INTERVIEW - Venezuela's "sweetheart" champions Chavez: Venezuelan President Hugo Chavez speaks during a meeting
Jintao visit US	3	3	U.S., China diplomats to meet in Beijing: Less than a week after Chinese President Hu Jintao visited Washington,... http://bit.ly/fAz91q
	1	3	Has China really saved American consumers $600 billion, as Hu Jintao claims? http://econ.st/hMCeCb

the latent semantic space). The resulting area represents the relevance of the tweet with respect to both the query and its topic (cluster), and is used as a feature. The area is computed by Heron's formula shown as follow:

$$S = \sqrt{s(s-a)(s-b)(s-c)} \quad (1)$$

where $a = ||\vec{q_p} - \vec{c_p}||$, $b = ||\vec{t_p} - \vec{c_p}||$, $c = ||\vec{q_p} - \vec{t_p}||$, and s is defined as $1/2(a+b+c)$. This area is computed for each topic cluster, resulting in k different features.

3.3 Filtering

For the real-time adhoc task, non-English tweets are not considered relevant even if they contain relevant information in foreign languages. Retweets are not considered relevant, either. Thus, it is expected to improve search performance, specifically precision, to filter out all non-English tweets or retweets. In addition, low-ranked tweets are removed because the official performance metric is precision at 30 (P@30) and retrieved tweets are ordered in reverse chronological order, where it is important to retain only (deemed) highly relevant tweets. To this end, we apply the filtering procedure as follows:

1. Remove all the tweets starting with "`RT`" or those with the HTTP status code 302.
2. Remove all the tweets containing non-ASCII characters more than 15 percentages of their length.
3. Remove all the tweets whose lang element, extracted from their JSON format, is non-`en` & which are determined as non-English by Google Language Detection API[8]

[8] `http://code.google.com/apis/language/translate/v2/getting_started.html`

4. Remove low-ranked tweets.

3.4 Relevance Assessment

After defining features for L2R, we need training data to train a learning to rank model. As there was no available training data we could find, we created a small amount of training data by ourselves using the 12 example topics distributed by the track organizers. For each example topic, we obtained top 300 tweets from $T_{example}$ (see Section 2) retrieved by Indri with default settings.

Then, a non-native English speaker annotated all retrieved tweets according to two criteria: relevance and interestingness each on a scale of 1 to 3 corresponding to 'bad', 'neutral', and 'good', respectively. More relevant/interesting tweets were assigned higher scores. Table 4 shows some example tweets and their scores (labels).

We defined relevance as containing user query words or their synonyms in a tweet, and interestingness as containing informative information (which is subjective). The final tweet score was defined as a product of these scores. With the training data, a ranking model was learned using Ranking SVM.

4 Experiments and Results

4.1 Experimental Setup

To learn a ranking function (ranker) which provides tweets' ranking in relevance order, we prepared the training data based on the features as described in Sections 3.1 and 3.2. We use SVM^{rank} as a ranker, an implementation of Ranking SVM [6]. No kernel was used in order to speed up the learning process and to reduce the number of parameters to be optimized. After learning, we performed the leave-one-out cross-validation on the 12 example topics to determine the optimum parameter C of Ranking SVM, which controls the trade-off between empirical loss and regularization. We tested 0.001, 0.003, 0.005, 0.008, and 0.01.

With the optimum parameters ($C = 0.001$), we reranked both search results T_{topic1} and T_{topic2} for the 50 test queries. As the output, we considered only top 30 tweets after reranking.

4.2 Evaluation

The assessors at NIST judged the relevance of pooled tweets from the 184 runs submitted by the total of 58 participating groups. Among the 50 test topics, it was reported that the 50th topic had no relevant tweets. Among the rest, 33 topics was reported to have highly relevant tweets. We separately report their results in Table 5, where "all" are results for the 49 topics, and "high" for the 33 topics.

The primary evaluation measure for this task is the average of Precision at 30 (P@30). R-Precision (R-prec) and Mean Average Precision (MAP) are also shown in the table for reference. The run names in the first column corresponds to particular experimental setting as follows:

- Lucene-1000[9]: Top 1000 ranked results by a disjunctive baseline run using Lucene[10] provided by the organizers.
- Lucene-30: Top 30 ranked results of Lucene-1000.
- Indri$_1$-1000: Top 1000 ranked results retrieved by Indri search with index I_1 (i.e., T_{topic1}).
- Indri$_2$-1000: Top 1000 ranked results retrieved by Indri search with index I_2 (i.e., T_{topic2}).
- Indri$_2$-1000-rt: Top 1000 tweets of T_{topic2} after retweet filtering (see Section 3.3).
- Indri$_2$-1000-rtlang: Top 1000 tweets of T_{topic2} after retweet and language filtering.
- Indri$_2$-30: An official run submitted as "normal". Top 30 tweets of T_{topic2}.
- Indri$_1$-30-rtlang: Top 30 tweets of T_{topic1} after retweet and language filtering.
- Indri$_1$-30-lr-rtlang: Another official run submitted as "ri". Top 30 of T_{topic1} after reranking and filtering. In this case, we set reduced dimension with $p = 100$ due to the restriction of computational resources and the number of clusters with $k = 12$ because the 12 example topics exist.

We used the two types of indexes I_1 and I_2 to obtain the results Indri$_1$-1000 and Indri$_2$-1000, respectively. Both results were obtained from Indri search engine and were evaluated for the top 1000 tweets against the official relevance judgment. Indri$_1$-1000 is the results for the index using the entire corpus, and Indri$_2$-1000 is for the indexes built for individual topics. Our expectation was that the index for the entire corpus would produce better results as it contains more information but clearly it did not in this case. Individual indexes were found to be better despite of less information. This result means that future information distorted statistics such as term weights, resulting in the inferior performance for real-time search.

Comparing Indri$_2$-1000 to Indri$_2$-1000-rt, Indri$_2$-1000-rtlang, and Indri$_2$-30, we can see how much improvement we gained through the three filters (i.e. retweet, non-English, and low-ranked filters). In P@30, retweet filter and non-English filter improved the performance by 26% and 33%, respectively. Also, filtering low-ranked tweets, Indri$_2$-30, dramatically improved the performance by 227%. The same result holds for a different search engine, Lucene. Lucene-30 (the top 30 tweets taken from Lucene-1000) improved the performance by 225% comparing to Lucene-1000.

[9] http://trec.nist.gov/act_part/tracks.new11.html
[10] http://lucene.apache.org/

Table 5: Results comparing different settings.

Run	P@30 (all)	R-prec (all)	MAP (all)	P@30 (high)	R-prec (high)	MAP (high)
Lucene-1000	0.0986	0.1486	0.1411	-	-	-
Lucene-30	0.3204	0.2130	0.1645	-	-	-
Indri_1-1000	0.0728	0.0710	0.0723	-	-	-
Indri_2-1000	0.0959	0.1505	0.1403	-	-	-
Indri_2-1000-rt	0.1204	0.1784	0.1620	-	-	-
Indri_2-1000-rtlang	0.1272	0.1907	0.1699	-	-	-
Indri_2-30 (**normal**)	0.3136	0.2123	0.1608	0.0869	0.1767	0.1582
Indri_1-30-rtlang	0.3871	0.1950	0.1526	-	-	-
Indri_1-30-lr-rtlang (**ri**)	0.4265	0.2635	0.2227	0.1303	0.2232	0.2079

This result suggests that topical relevance of tweets is much more important than when they were posted, which is counter-intuitive for real-time search. The improvement is presumably due to the fact that the corpus contains tweets only for 17 days from Jan. 23 to Feb. 7, 2011. As a whole, these filters are simple but essential on this year's rule.

Lastly, the result of the proposed approach (filtering after reranking), indri_1-30-lr-rtlang, performed better than indri_2-30-rtlang. Reranking further improved P@30 by about 7%, and this is the best result among our official submissions.

We summarize the results of our experiments as follows:

1. Different indexes resulted in different results; future statistics of word distribution had harmful effects for IR precision.
2. Simple retweet and non-English filters significantly improved retrieval performance. This result also indicates that there are many retweets and some non-English tweets in search results.
3. Focusing on highly ranked tweets resulted in a striking boost in performance, suggesting that temporal closeness between a query and them seems to play an unimportant role in the Tweets2011 corpus.
4. The learning to rank model is effective for improving the performance for real-time twitter search.

5 Conclusion and Future work

Through the real-time adhoc task of TREC 2011 Microblog track, we developed a two-step approach: reranking and filtering. Filtering step identifies and removes retweets and non-English tweets, which was found crucial for this task. After filtering, we reranked tweets based on an L2R model learned using five types of features. The reranking step further improved the search performance, achieving the best overall result. For future work, we plan to analyze effective features for learning to rank and develop a feature selection method suited for real-time search. Besides, we will use query expansion based on time-sensitive features and algorithm to improve IR performance.

Reference

[1] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the 18th annual ACM-SIAM symposium on discrete algorithms*, pages 1027–1035, 2007.

[2] C. Castillo, M. Mendoza, and B. Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World Wide Web*, pages 675–684, 2011.

[3] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[4] A. Dong, R. Zhang, P. Kolari, J. Bai, F. Diaz, Y. Chang, Z. Zheng, and H. Zha. Time is of the essence: improving recency ranking using Twitter data. In *Proceedings of the 19th international conference on World Wide Web*, pages 331–340, 2010.

[5] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H. Shum. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd; international conference on computational linguistics*, pages 295–303, 2010.

[6] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 133–142, 2002.

[7] K. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[8] D. Metzler and W. Croft. Combining the language model and inference network approaches to retrieval. *Information processing & management*, 40(5):735–750, 2004.

[9] G. Neubig, Y. Matsubayashi, M. Hagiwara, and K. Murakami. Safety information mining - what can NLP do in a disaster -. In *Proceedings of the 5th international joint conference on natural language processing (IJCNLP)*, pages 965–973, Nov 2011.

[10] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World Wide Web*, pages 851–860, 2010.

[11] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World Wide Web*, pages 1177–1178, 2010.