# ECNU at TREC 2015: Microblog Track

Qin Chen, Bo Wang, Beijing Huang, Qinmin Hu and Liang He
Shanghai Key Laboratory of Multidimensional Information Processing
East China Normal University, 500 Dongchuan Road, Shanghai, 200241, China
{qchen, bwang, bjhuang}@ica.stc.sh.cn, {qmhu, lhe}@cs.ecnu.edu.cn

**Abstract**

This paper describes our participation in TREC 2015 Microblog track, which includes two tasks related to Scenario A and Scenario B. For Scenario A, we build a real-time tweet push system, which is mainly composed by three parts: feature extraction, relevance prediction and redundancy detection. Only the highly relevant and nonredundant tweets are sent to users based on the interest profiles. For Scenario B, we apply three query expansion methods, namely the web search based, the TFIDF-PRF based and the Terrier embedded PRF based. In addition, three state-of-the-art information retrieval models as the language model, BM25 model and DFRee model are utilized. The retrieval results are combined for final delivery. The experimental results in both scenarios demonstrate that our system obtains convincing performance.

## 1 Introduction

The TREC 2015 Microblog Track still focuses on the real time filtering. The goal is to push interesting content to a user according to his/her interest profile. Specifically, two concrete tasks are given to simulate two scenarios as Scenario A and Scenario B.

Scenario A is about pushing notifications on a mobile phone. A system for this scenario is allowed to return a maximum of 10 tweets per day per interest profile. The evaluation metric will penalize the gap between the tweet time and the notification time. To fulfill this task, we build a real-time tweet push system, whose architecture is shown in Figure 1. The system mainly consists of three parts: feature extraction, relevance prediction and redundancy detection. Only the tweets which are both relevant and nonredundant are finally sent to users. The performance of our submitted runs proves the effectiveness of the system.

Regarding to Scenario B, it focuses on the periodic email digest. Specifically, a system for this scenario is required to deliver a batch of up to 100 ranked interesting tweets per day per interest profile. For this task, we leverage the Terrier search engine for indexing and retrieval, where the obtained tweets are indexed and retrieved by the hour. Three query expansion methods, namely the web search based [2], TFIDF-PRF based [2] and the Terrier embedded pseudo relevance feedback based [5], are applied for query expansion. In addition, we leverage three state-of-the-art information retrieval (IR) models, namely the language model [6], the BM25 model [7] and the DFRee model [1]. Just before the day ends, all the results are combined and merged together to send to users.

# 2 Scenario A: Push Notifications on a Mobile Phone

## 2.1 System Overview

In this section, we demonstrate the architecture of our system, which is shown in Figure 1. It shows that our system mainly consists of three parts, namely feature extraction, relevance prediction and redundancy detection. The details of each part are demonstrated in the following sections.

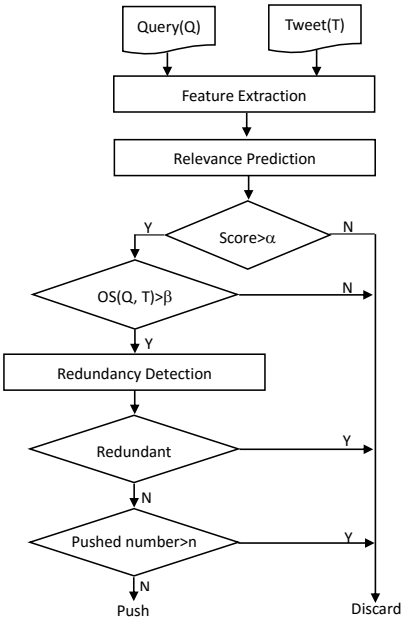


Figure 1: System Architecture

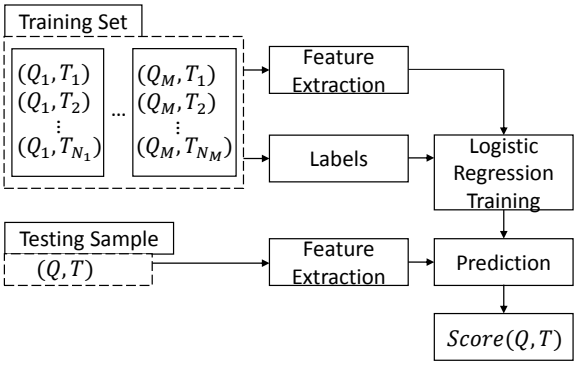


Figure 2: Relevance Prediction Framework

## 2.2 Feature Extraction

We mainly focus on the three types of features as follows: (1) **f(Q, T):** the text similarity between the original query and the tweet; (2) **f(GT, T):** the text similarity between the Google searched titles and the tweet; (3) **f(GS, T):** the text similarity between the Google searched snippets and the tweet.

To calculate the text similarities, two widely used measures as the Jensen-Shannon Divergence (JSD) [3] and the Overlap Similarity(OS) are applied. Therefore, a total of six features can be extracted based on the text similarities. For the sake of simplicity, we only demonstrate how to extract f(Q, T). The features as f(GT, T) and f(GS, T) can be obtained in a similar way.

For the JSD measure, both the queries and tweets are required to be probability vectors, where all the entries sum up to 1. In our experiment, we obtain the vectors by normalized TF weighting with smoothing. Specifically, given a query vector as $V_Q$ and a tweet vector as $V_T$, the JSD similarity is calculated as follows:

$$JSD(Q,T) = \frac{1}{2}KL(V_Q||M) + \frac{1}{2}KL(V_T||M) \tag{1}$$

where $M$ is the average vector of the query and tweet. $KL(V_Q||M)$ denotes the Kullback-Leibler divergence [4] between the distributions of $V_Q$ and $M$.

To calculate the Overlap Similarity (OS) between a query and a tweet, we first obtain the corresponding word sets as $S(Q)$ and $S(T)$. Then, $OS(Q,T)$ is formulated as:

$$OS(Q,T) = \frac{|S(Q) \cap S(T)|}{|S(Q)|} or \frac{|S(Q) \cap S(T)|}{|S(T)|} \tag{2}$$

where $S(Q) \cap S(T)$ represents the intersection of $S(Q)$ and $S(T)$, and $|.|$ denotes the size of the set.

### 2.3 Relevance Prediction Framework

Figure 2 shows the framework of the relevance prediction part. A Logistic Regression model is trained offline for its simplicity and effectiveness. We use the official results of Microblog Track 2013 and 2014 as the training data set. The relevance judgement degree is ignored, which means both the relevant (judged as 1) and highly relevant (judged as 2) are assigned with a label 1. The features applied are demonstrated in Section 2.2. For a coming tweet in the tweet stream, we can quickly extract the features, and predict the relevance score with the trained model.

### 2.4 Redundancy Detection Strategy

Since the pushed tweets are expected to be in different official clusters ideally, we perform redundant detection for the candidates. Specifically, for a candidate tweet specific to a query, we devise a redundancy detection strategy to determine whether it is redundant or not. The strategy is demonstrated in Algorithm 1.

**Algorithm 1** Redundancy Detection

**Input:** a candidate pushed tweet $T$ corresponding to query $Q$, the pushed tweet set $P$, the maximum pushed number $n$, the similarity threshold determining redundancy $\gamma$, the redundancy degree threshold $\theta$

**Output:** the Boolean redundancy label $L(T)$ // True: redundant, False: nonredundant

**Initialize:** redundancy $Count = 0$, redundancy degree $RD(T) = 0$
**for** $T' \in P$ **do**
  Calculate the Overlap Similarity $OS(T, T')$
  **if** $JSD(T, T') < \gamma$ **then**
    $Count \mathrel{+}= 1$
  **end if**
**end for**
$RD(T) = Count/|P|$
**if** $RD(T) > \theta$ **then**
  $L(T) = True$
**else**
  $L(T) = False$
**end if**

## 3 Scenario B: Periodic Email Digest

### 3.1 Queries

We apply three query expansion methods, namely the web search based [2], TFIDF-PRF based [2] and the Terrier embedded pseudo relevance feedback [5] based. By various combinations of the three methods, we obtain six kinds of queries as follows: (1) **Q:** the original query; (2) **QG:** query expanded by the Google searched results; (3) **QT:** query expanded with the TFIDF-PRF based method; (4) **QP:** query expanded with the Terrier embedded pseudo relevance feedback; (5) **QGT:** the union of QG and QT; (6) **QGTP:** the union of QG, QT and QP

### 3.2 IR models

With the daily tweet stream, we leverage the Terrier search engine [5] for indexing and retrieval. Three state-of-the-art information retrieval (IR) models, namely the language model [6], the BM25

model [7] and the DFRee model [1], are utilized for this task. Specifically, with the above six kinds of queries and three IR models, we can obtain eighteen scores for a tuple as (Query, Tweet).

By assuming that different retrieval models may compensate each other by combination, we do a linear combination of the scores to obtain better performance. We use the collection of Tweet2013 for training and try out all kinds of combinations. The best combination strategies are used for this year's task.

# 4 Experimental Setup

## 4.1 Datasets

The training data sets as Tweet2013 and Tweet2014 is obtained with the official search API[1]. In this year's track, we use the officially provided GatherStatusStream tool[2] to gather a parallel sample of tweets from the Twitter public stream during the evaluation period, i.e., from July 20, 2015, 00:00:00 UTC to July 29, 2015, 23:59:59 UTC.

Due to the government policy on networks, we have missed some data on the first day. To obtain the complete data set, we rent a Linode server and perform experiments on it. The total amount of tweets we obtained is about 14 million. Since the raw tweet texts contain a lot of noise, we perform data preprocessing to remove non-English tweets and specific symbols like URLs, mentions and so on. We also use Natural Language Toolkit[3] for tokenization, stemming and stop words removal.

## 4.2 Evaluation Measures

For Scenario A, two metrics are designed for evaluation. The primary metric is expected latency-discounted gain (ELG) from the temporal summarization track, which is formulated as:

$$(1/|tweets|) \times \sum Gain(tweet) \tag{3}$$

where $|tweets|$ denotes the total number of returned tweets. For a returned tweet, the gain is calculated according to the official guidelines[4]. The secondary metric is the normalized cumulative gain (nCG), which is formulated as:

$$(1/Z) \times \sum Gain(tweet) \tag{4}$$

where $Z$ is the maximum possible gain (given the 10 tweet per day limit).

In Scenario B, the list of tweets returned per day are treated as a ranked list for each topic. Then, $NDCG@k$ is computed, where $k$ is relatively small and determined based in part on the pool depth. The score of a topic is the average of the $NDCG@k$ scores across all days in the evaluation period. The score of the run is the average over all topics.

# 5 Results and Discussions

Table 1 shows the results of our submitted runs for Scenario A. All the three runs vary in the parameter settings of $\alpha$, $\beta$, $\gamma$ and $\theta$, which are demonstrated previously. The specific settings for each run is as follows: ECNURUNA1(0.17, 0.3, 0.3, 0.5), ECNURUNA2(0.18, 0.4, 0.25, 0.6), ECNURUNA3(0.20, 0.45, 0.20, 0.7). The best result for each metric is marked in bold. From this table, we observe that the third run, namely ECNURUNA3, performs the best with regard to the ELG metric. Regarding to the nCG measure, all the three runs have almost the same performance.

[1]https://github.com/lintool/twitter-tools/wiki/TREC-2013-API-Specifications
[2]https://github.com/lintool/twitter-tools/wiki/Sampling-the-public-Twitter-stream
[3]http://www.nltk.org/
[4]https://github.com/lintool/twitter-tools/wiki/TREC-2015-Track-Guidelines

For Scenario B, the performance of our submitted runs are shown in Table 2. Specifically, ECNURUNB1 is combined by (QGTP, DFRee) and (Q, LM). Here we use (QGTP, DFRee) to denote the retrieval results configured with the query QGTP and the DFRee model. ECNURUNB2 is combined by (QGTP, DFRee), (Q, LM) and (QT, BM25). ECNURUNB3 is combined by (QGTP, DFRee), (Q, LM), (QG, BM25) and (QGTP, BM25). We observe that though the first run (ECNURUNB1) is combined by fewer retrieval scores, it performs the best. Thus, it is important to know the characteristics of each IR model and apply appropriate combination strategies. Combination by more retrieval results may harm the performance.

Table 1: Performance of our submitted runs for Scenario A

| Metrics | ECNURUNA1 | ECNURUNA2 | ECNURUNA3 |
|---|---|---|---|
| ELG | 0.2193 | 0.2262 | **0.2322** |
| nCG | 0.2688 | **0.2698** | 0.2695 |

Table 2: Performance of our submitted runs for Scenario B

| Metrics | ECNURUNB1 | ECNURUNB2 | ECNURUNB3 |
|---|---|---|---|
| nDCG | **0.1610** | 0.1327 | 0.1416 |

# 6 Conclusions

In this paper, we present our work in two scenarios of the TREC 2015 Microblog Track. For Scenario A about pushing notifications on a mobile phone, we build a real-time tweet push system. It mainly performs three steps to determine whether to push a tweet or not. The maximum ELG value as 0.2322 is obtained by our system. For Scenario B which focuses on the periodic email digest, we apply three query expansion methods and three state-of-the-art IR models for search. Various retrieval results are combined for final delivery. We obtain the best nDCG performance as 0.1610 in this scenario. Noting that the combination strategy does not work very well, we are extracting more useful features and focus on the learning to rank approaches.

# Acknowledgment

This research is funded by Science and Technology Commission of Shanghai Municipality (No.15PJ1401700 and No.13511507902).

# References

[1] G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389, 2002.

[2] Q. Chen, Q. Hu, Y. Pei, Y. Yang, and L. He. ECNU at TREC 2014: Microblog track.

[3] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. In *IEEE International Symposium on Information Theory*, pages 31–31, 2004.

[4] J. M. Joyce. Kullback-leibler divergence. In *International Encyclopedia of Statistical Science*, pages 720–722. Springer, 2011.

[5] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson. Terrier information retrieval platform. In *Advances in Information Retrieval*, pages 517–519. Springer, 2005.

[6] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998.

[7] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at TREC-3. *NIST SPECIAL PUBLICATION SP*, pages 109–109, 1995.