

QCRI at TREC 2014: Applying the KISS principle for the TTG task in the Microblog Track

Walid Magdy

Qatar Computing Research Institute
Qatar Foundation
Doha, Qatar
wmagdy@qf.org.qa

Wei Gao

Qatar Computing Research Institute
Qatar Foundation
Doha, Qatar
wgao@qf.org.qa

Tarek Elganainy

Qatar Computing Research Institute
Qatar Foundation
Doha, Qatar
telganainy@qf.org.qa

Zhongyu Wei

The Chinese University of
Hong Kong
Hong Kong, China
zywei@se.cuhk.edu.hk

ABSTRACT

In this paper we present our work on the ad-hoc search and the tweet timeline generation (TTG) tasks of TREC-2014 Microblog track. Regarding the ad-hoc search task, we used our best developed system over the last year, which include hyperlink-based query expansion and re-ranking models fusion. For the new tweet timeline generation task, we applied a straightforward and simple approach, which depends on clustering retrieval results based on cosine similarities between tweets. We believe that our simple approach achieved a good result, since it was shown in scatter-plot of participants' runs that our best run in the TTG task achieved the 4th rank among 48 runs based on the f1-measure.

1. INTRODUCTION

We describe the participation of Qatar Computing Research Institute (QCRI) group in the TREC-2014 Microblog track. This year the track included two tasks; the ad-hoc search task, and the newly introduced tweets timeline generation (TTG) task. We applied what we have learned from our participation in the track in the past three years in the ad-hoc task, which include hyperlink-based query expansion methods [4, 11] and the selection and fusion of multiple re-ranking models [4, 5]. We configured our retrieval system according to the best results achieved when tested on the topics of 2013 [4, 5, 11], since it is the same collection used this year but with new topics set.

We submitted for runs for the ad-hoc task while enabling and disabling hyperlink-based pseudo relevance feedback (HPRF) and reranking. The run which applied both HPRF and reranking was then used in the TTG task by clustering the results according to similarity.

For the TTG task, since it is running for the first year, we decided to keep it simple and straightforward (KISS) by using a simple implementation of Jaccard similarity to measure the distance between tweets in the top N retrieved results and cluster those of high similarity together. Four runs was submitted for the TTG task by using different values for N , and applying two different formulas for calculating the similarity between tweets.

Details and results of our runs are described below.

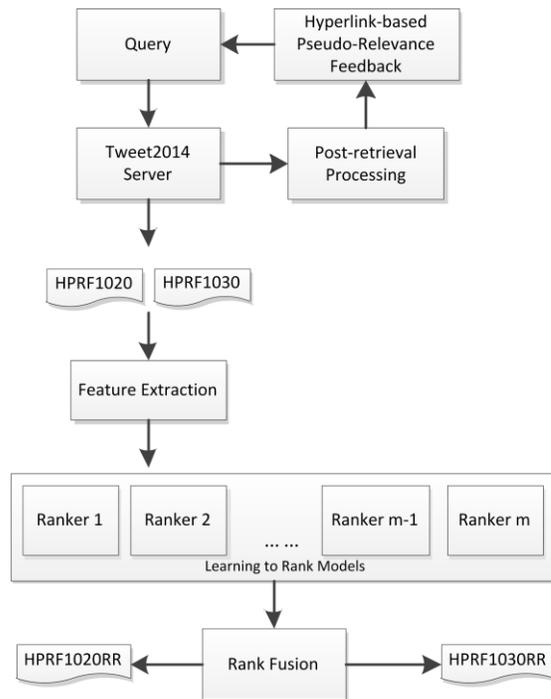


Figure 1 Ad-hoc search system

2. AD-HOC SEARCH TASK

Figure 1 presents the full architecture of our microblog ad-hoc retrieval system.

Overall, we designed our pipeline to combine query expansion and result re-ranking. For query expansion, we made use of the external documents linked by the URLs in the initial search results for query expansion. For result re-ranking, our system resorted to learning to rank by extensive engineering work for re-ranking search results given by combining the ranked lists of different rankers.

2.1 Hyperlink-based Pseudo Relevance Feedback (HPRF)

A hyperlink in a tweet is more than a link to related content as in webpages, but actually it is considered a link to the main focus of the tweet. In fact, sometimes tweet’s text itself is totally irrelevant, and the main content lies in the embedded hyperlink, e.g. “This is really amazing, you have to check htwins.net/scale2”.

Analyzing the TREC microblog dataset over the past three years, we found more than 70% of relevant tweets contain hyperlinks. This motivates utilizing the hyperlinked documents content in an efficient way for query expansion.

The content of hyperlinked documents in the initial set of top retrieved tweets is extracted and integrated into the PRF process. Titles of hyperlinked pages usually act like heading of the document’s content, which can enrich the vocabulary in the PRF process.

We apply hyperlinked documents content extraction on two different levels:

- Tweets level (**PRF**): which represents the traditional PRF, where terms are extracted from the initial set of retrieved tweets while neglecting embedded hyperlinks.
- Hyperlinked document titles level (**HPRF**): where the page titles of the hyperlinked documents in feedback tweets are extracted and integrated to tweets for term extraction in the PRF process.

Titles and meta-description of hyperlinked documents may include unneeded text. For example, titles usually contain delimiters like ‘-’ or ‘|’ before/after page domain name, e.g., “... | CNN.com” and “... - YouTube”. We clean these fields through the following steps [4, 5]:

- Split page titles on delimiters and discard the shorter substring, which is assumed to be the domain name.
- Detect error page titles, such as “404, page not found!” and consider them broken hyperlinks.
- Remove special characters, URLs, and snippet of HTML/JavaScript/CSS codes.

This process helps in discarding terms that are potentially harmful if used in query expansion.

TFIDF and Okapi weighting [**Error! Reference source not found.**] were used for ranking the top terms were used for query expansion. We calculate TFIDF for a term x as follows:

$$TFIDF(x) = [tf_t(x) + \partial_1 tf_{h_t}(x) + \partial_2 tf_{h_d}(x)] \cdot \log \frac{N}{df(x)} \quad (1)$$

Where $tf_t(x)$ is the term frequency of term x in the top n_d initially retrieved tweet documents used in the PRF process; $tf_{h_t}(x)$ is the term frequency of term x in the titles of hyperlinks in the top n_d tweets; and $tf_{h_d}(x)$ is the term frequency of term x in the meta-description of hyperlinks in the top n_d tweets. ∂_1 and ∂_2 are binary functions that equal to 0 or 1 according to the content level of hyperlinked documents used in the expansion process. $df(x)$ is document frequency of term x in the collection; and N is the total number of documents in the collection.

k_1 and b free parameters of the Okapi weighting were selected as 2 and 0 respectively. The parameter b was set to 0 since the variation in tweets length is limited due to Twitter constraint on the number of characters used (max. 140 characters).

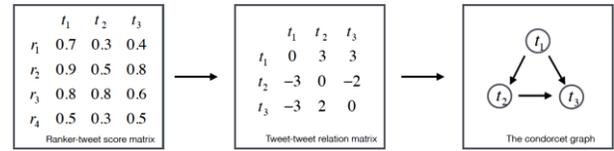


Figure 2 Demonstration for Condorcet-fuse algorithm

Terms extracted from the top n_d initially retrieved documents are ranked according to equation 1, and top n_r terms with the highest TFIDF are used to formulate Q_E for the expansion process. Weighted geometrical mean is used to calculate the final score of retrieval for a given Q query according to equation 2:

$$P(Q|d) = \sqrt{P(Q_0|d)^{1-\alpha} \cdot P(Q_E|d)^\alpha} \quad (2)$$

Where Q_0 is the original query; Q_E is the set of extracted expansion terms; $P(Q|d)$ is the probability of query Q to be relevant to document d . α is the weight given to expansion terms compared to original query (when $\alpha=0$, no expansion is applied). Language-model-based retrieval model was used to calculate the probability of relevance.

2.2 Tweets Re-ranking

Similar to our idea in TREC2013 [4], we also explored to ensemble multiple ranking models for re-ranking the retrieved tweets. Our models were learned using Tweets2011-13 qrels and tested with Tweets2014 queries. We employed six learning to rank algorithms as the candidate rankers for search result fusion: RankNet [2], RankBoost [6], Coordinate Ascent [9], MART [7], LambdaMART [12] and RandomForests [1] using RankLib package¹. Based on these algorithms, we trained eight different rankers: (1) A Rankboost model was trained without validation set; (2) A MART model was learned using 80% training queries for training and 20% training queries for validation; (3) A RandomForest model was learned in the same way as (2); (4) A RankNet model was learned in the same way as (2); (5) Two Coordinate Ascent models were learned in the same way as (2) but one of them optimized MAP and the other optimized P@30; (6) Two LambdaMART models were learned in the same way as (5). Different from the configurations of last year, we did not use query selection methods to construct validation set since this strategy did not bring much effectiveness to our system of TREC2013 [4]. However, we used exactly the same feature list as last year which were shown useful (see [4] for detail).

Last year, we simply summated relevance scores of all learning-to-rank models for tweets re-ranking. Instead of that, we tried to combine the ranking scores of candidate rankers by *weighted Condorcet-fuse* this year. *Condorcet-fuse* is one of the state-of-the-art fusion methods in metasearch due to its effectiveness [10]. The basic idea is that tweets that can beat more tweets in a pairwise manner based on scores they received from candidate rankers should be ranked higher. Taking ranked lists generated by candidate rankers as input, we produced a Condorcet graph and output the final ranked list by computing the Hamiltonian path of that graph. The workflow of generating Condorcet graph is demonstrated in Figure 2.

¹ <http://sourceforge.net/p/lemur/wiki/RankLib/>

Table 1 QCRI results in TREC 2014 Microblog track for the ad-hoc search task

Run	MAP	P@30
PRF1030	0.4941	0.6679
HPRF1020	0.5075	0.6685
PRF1030RR	0.4998	0.6988
HPRF1020RR	0.5122	0.6982

Given four candidate rankers and three tweets, we have relevance scores for tweets assigned by rankers which form a ranker-tweets matrix shown in the first frame. (r_i, t_j) stands for the relevance score given by candidate ranker r_i to tweet t_j . We then derive the tweet-tweet relation matrix to reveal the pair-wise preference. For a pair of tweets (t_j, t_k) , we compute their relation score by counting the number of rankers giving higher score to t_j than t_k . Thirdly, we generate the Condorcet graph. For a pair of tweets t_j and t_k , there exists an edge from t_j to t_k if the value of (t_j, t_k) in tweet-tweet relation matrix is higher than or equal to 0. For the tweets that tie, there is an edge pointing in each direction. A Hamiltonian traversal of this graph will produce the final ranked list. The detail of the algorithm can be found in [10].

To reflect the different importance of candidate rankers, we implemented a weighted version of Condorcet-fuse. In this case, t_j wins t_k if the sum of the weights of those rankers that rank t_j higher than t_k is larger than the sum of the weights of those that prefer t_k to t_j . We used the mean average precision (MAP) obtained by individual candidate ranker on Tweets2011-2013 dataset as the weight of the corresponding ranking model.

2.3 Submitted Runs & Results

We had four submitted runs to the ad-hoc search task this year, as follows:

- **PRF1030**: Applied standard pseudo-relevance feedback with number of documents in feedback = 10, and number of terms in the feedback process = 30. Selection of values is based on our study to different values of feedback documents and terms in [5].
- **HPRF1020**: Applied Hyperlink-based PRF with number of document and terms used in feedback = 10 and 20 respectively.
- **PRF1030RR**: PRF1030 run after applying reranking
- **HPRF1020RR**: HPRF1020 run after applying reranking

Results achieved by our runs are presented in Table 1.

Results shows that HPRF led to slight improvement over just using PRF on both MAP and P@30. This improvement was found insignificant, which does not align with results reported on TREC-2013 dataset [5]. However, reranking led to noticeable improvement to P@30, with slight improvement to MAP. Our best achieved scores are highlighted in Table 1.

3. TWEETS TIMELINE GENERATION TASK

3.1 Approach

Our expectation was that HPRF1020RR would achieve the best result; this is why we used this run for the TTG task.

For generating the timeline of tweets, we applied the following:

Table 2 QCRI results in TREC 2014 Microblog track for the TTG task

Run	P	R_{uw}	R_w	$F1_{uw}$	$F1_w$
EM50	0.4150	0.2867	0.4779	0.3391	0.4442
EM100	0.3301	0.3797	0.5650	0.3532	0.4167
SM50	0.4798	0.1688	0.3221	0.2497	0.3854
SM100	0.3881	0.2057	0.3416	0.2689	0.3634

1. Top ranked N tweets were normalized by removing name mentions, hashtags, urls, emoticons, and stopwords.
2. Porter stemmer was applied to tweets' text
3. Similarity was calculated among top N tweets in the results list.
4. INN clustering approach was applied to merge any tweets with close distance into the same cluster. Distance between two tweets was calculated as follow:

$$distance(t_i, t_j) = 1 - similarity(norm(t_i), norm(t_j))$$

Where $norm(t_i)$ is the normalized version of the tweet t_i after applying step 1 and 2.

We applied two implementations to the similarity, which are a modification to the Jaccard similarity coefficient as follows:

$$similarity_{EM}(A, B) = \frac{|A \cap B|}{\max(|A|, |B|)}$$

$$similarity_{SM}(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)}$$

$similarity_{EM}$ calculates the similarity between the text of two tweets as the number of common terms divided by the length of the longest tweet. This leads to merging two tweets in the same cluster if most of the terms in the long tweet existed in the short tweet, and the difference in the length between both tweets is not large. $similarity_{SM}$ leads to severe merging, since it focus on how many of the terms of the short tweet exist in the long tweet regardless to the difference in length. In the extreme case, if a tweet contains only one word that exists in the long tweet, $similarity_{SM}$ would equal to 1.

3.2 Submitted Runs & Results

We had four submitted runs to the ad-hoc search task this year, as follows:

- **EM50**: Top 50 retrieved results from the HPRF1020RR run were clustered using $similarity_{EM}$ as the distance function. A similarity of at least 0.6 was required to any of the tweets in a cluster to get the tweet merged to the cluster.
- **EM100**: similar to EM50, but top 100 retrieved results were used instead.
- **SM50**: similar to EM50, but $similarity_{SM}$ was used instead.
- **SM100**: similar to EM100, but $similarity_{SM}$ was used instead.

For all runs, the earliest tweet in each cluster is used to represent the cluster in the submitted run.

Results of our TTG runs are shown in Table 2. The second similarity formula led to merging most of the tweets into small number of cluster. This led to low recall but better precision.

However, the overall F1 score was lower than using $similarity_{SM}$. EM100 achieved a better unweighted F1 measure, while EM50 achieved a better weighted F1 measure, which according to the scatter plot of all submitted runs, achieved the 4th rank among 48 runs.

4. REFERENCES

1. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
2. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *ICML 2005*.
3. A. S. El-Din and W. Magdy. Web-based Pseudo Relevance Feedback for Microblog Retrieval. *TREC 2012*
4. T. El-Ganainy, Z. Wei, W. Magdy, and W. Gao. (2013). QCRI at TREC 2013 Microblog Track. *TREC 2013*
5. T. El-Ganainy, W. Magdy, and A. Rafea. Hyperlink-Extended Pseudo Relevance Feedback for Improved Microblog Retrieval. *SoMeRA 2014*
6. Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.
7. J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.
8. J. Lin and M. Efron. (2013). Overview of the TREC2013 microblog track. *TREC 2013*
9. D. Metzler and W. B. Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.
10. M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. *CIKM 2002*.
11. Z. Wei, W. Gao, T. El-Ganainy, W. Magdy, K-F. Wong. Ranking Model Selection and Fusion for Effective Microblog. *SoMeRA 2014*
12. Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.