HIT_LTRC at TREC 2011 Microblog Track

Yun Li, Xishuang Dong, Yi Guan

{nicyun,dongxishuang}@gmail.com, guanyi@hit.edu.cn

Abstract. This paper describes our entry into the TREC 2011 Microblog track. We submitted two runs in this year's track, both in real-time fashion and without any external resources. The runs were generated through a three-step procedure, including scoring, threshold selection and re-ranking. The evaluation results of our submitted runs significantly outperform the disjunctive baseline run. We conducted additional runs to show our score decay and threshold selection strategies to be exceptionally effective.

1 Introduction

This paper describes our participation in the TREC 2011 Microblog track. We submitted two runs, *hitWIt* and *hitWId*, that all operated in a strict real-time fashion, and had no external resources involved, even for the web pages linked by URLs in the tweet's text.

We satisfied the real-time search constraints by dynamically indexing the tweets. For a specific query, the retrieval process was performed based on the repository that only indexed the tweets whose creation times were earlier than the query's timestamp, therefore the words' statistics like IDF were calculated without any future evidences.

We deployed a three-step method to meet this year's Microblog search requirement that the retrieved results should be ranked reverse-chronologically. Firstly, we gave each tweet a score according to its relevance to the assumed query. The run *hitWIt* was scored with solely language modeling approach [1], and the run *hitWId* with the case of score decay [2] considered additionally. This is the only difference between these two runs. In the second step, we calculated a score threshold for each query based on the score distribution of relevant and non-relevant tweets retrieved in the previous step, by modeling the relevant tweets' scores as the Gaussian distribution and the non-relevant tweets' scores as the Exponential distribution. Finally, the retrieved tweets with scores higher than the threshold were re-ranked according to their timestamp and returned.

Our submitted runs both outperform the disjunctive baseline run with statistically significant improvements. We presented extra experimental results to demonstrate the effectiveness of our score decay and threshold selection method. The runs with score decay factor perform better than the runs without it if the appropriate value of decay parameter is chosen, and the runs with adaptive threshold selection algorithm beat the runs with threshold fixed manually.

The next section describes the methods we employed to generate our submissions in detail. Section 3 describes the experimental methodology and results. Finally, section 4 gives conclusion and outlook of future work.

2 Method

We used a three-step process to generate our results. At first, we used a scoring function to give each tweet a relevance score with respect to one specific query. Secondly, we selected a threshold score for each query according to the distribution of the relevant and non-relevant tweets retrieved. Finally, the tweets with score higher than the threshold were re-ranked reverse chronologically according to their creation time.

Section 2.1 describes the scoring function we used to score a tweet-query pair, and section 2.2 describes the approach how the re-ranking threshold was determined. Other procedures like dynamic indexing and re-ranking are straightforward thus unnecessary to be represented.

2.1 Tweet Scoring

Our two submissions differ only in the scoring process. The run *hitWIt* used the basic language modeling scoring function, and the run *hitWId* taken the score decay circumstance into consideration. We will describe these two scoring functions in detail in this section.

Basic Score.

Our basic scoring function adopted Indri's [3] language modeling approach. In this approach, documents (or tweets) are scored by the likelihood the query was generated by the document's model. Specifically, documents are modeled by multiple-Bernoulli distribution [4] with Dirichlet prior as the Bayesian smoothing approach [5].

Given a tweet T and a query Q, the probability that the query was generated by the tweet's model is calculated as follows, according to Indri's retrieval model

$$\mathrm{P(Q|T)} = \sqrt[|Q|]{\prod_{r \in Q} \frac{\mathrm{tf_{r,T}} + \mu \mathrm{P(r|C)}}{|\mathrm{T}| + \mu}} \tag{1}$$

Where $tf_{r,T}$ is the term frequency of query's term r in tweet T, and μ is the tunable smoothing parameter.

The collection frequency of a term is calculated as $P(r|C) = cf_r/|C|$, and it is independent of the query.

We also employed pseudo relevance feedback [13], which has been shown to be an effective strategy to improve the performance of retrieval systems. PRF assumes the top *fbDocs* documents in the original ranked list to be relevant to the query, and then uses these documents to select some meaningful terms to expand the original query. Finally, the result retrieved by the expanded query would be better than the previous one. For simplicity, we just adopted Indri's Relevance Model [6] for expansion term selection. In this model, all terms in the feedback documents are ranked according to the probability $P(r|I)$, which means how likely the term r would be generated from the query Q. Then the top ranked *fbTerms* terms will be selected to expand the original query.

$$\mathrm{P(r|Q)} = \frac{\sum_{\mathrm{D}} \mathrm{P(r|T)P(Q|T)P(T)}}{\mathrm{P(Q)}} \tag{2}$$

Score decay.

Considering tweet's real time characteristic, the longer the time after its creation, the less importance it should have. We added a decay factor with half-life h to each score calculated, the half-life means for how long time the score will be decayed with only one half of the initial score left. The decay factor was modeled exponentially.

$$\mathrm{Score(t)} = \mathrm{Score_0} \cdot \mathrm{e}^{-(\mathrm{t}-\mathrm{t_0})/\tau} \tag{3}$$

Where $Score_0$ and t_o are the initial score and time, τ is a model parameter related to half time h as equation 4

$$\mathrm{h} = \tau \cdot \ln 2 \tag{4}$$

2.2 Threshold Selection.

After giving each tweet a score, the second step is aimed at picking out those most relevant tweets whose scores are above a meticulously calculated threshold. We will describe our techniques used for

threshold adaptation in this section.

In adaptive filtering systems, the method of score distribution for dissemination threshold selection has shown exceptional effectiveness [2, 10]. This method assumed a Gaussian distribution for the scores of relevant documents and an exponential distribution for the scores of non-relevant documents. Figure 1 illustrates how the Gaussian distribution fits relevant documents' scores of Topic 24 in this year's track, and figure 2 illustrates the non-relevant documents' case.

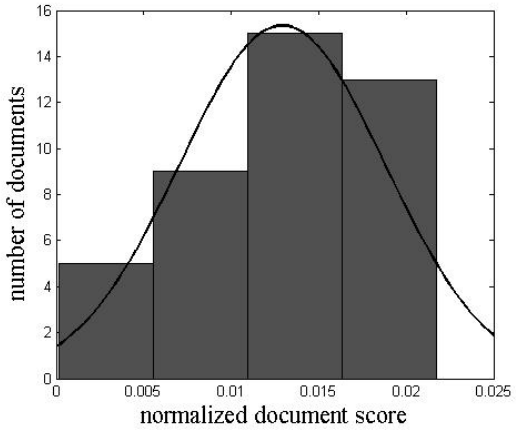

Fig. 1. Density of relevant document scores

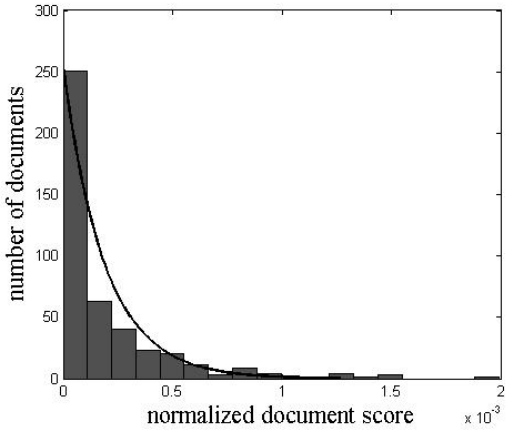

Fig. 2. Density of non-relevant document scores

We can't estimate the parameters of these two distributions directly because unlike adaptive filtering, we have no relevance judgments when generating the result of Microblog search. There is a solution for this problem in [7] in the context of distributed retrieval. The Gaussian-Exponential score density model without relevance judgments can considered as mixture model and resolved by standard Expectation Maximization (EM) [8] algorithm.

Equation 5 and 6 give the distribution densities of relevant and non-relevant documents' scores. Where x is the document's score and R is the judgment of the document's relevance to the query.

$$P(x|R=1) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5}$$

$$P(x|R=0) = \lambda e^{-\lambda x} \tag{6}$$

Then the E-step and M-step of the EM algorithm could be described as follows

E-step:

$$P(R=1|x_i) = \frac{P(R=1)P(x_i|R=1)}{\sum_{r=0}^{1} P(R=r)P(x_i|R=r)} \tag{7}$$

$$P(R=0|x_i) = \frac{P(R=0)P(x_i|R=0)}{\sum_{r=0}^{1} P(R=r)P(x_i|R=r)} \tag{8}$$

M-step:

$$\mu = \frac{\sum_{i=1}^{|x|} P(R=1|x_i) \cdot x_i}{\sum_{i=1}^{|x|} P(R=1|x_i)} \tag{9}$$

$$\sigma^2 = \frac{\sum_{i=1}^{|x|} P(R = 1|x_i) \cdot (x_i - \mu)^2}{\sum_{i=1}^{|x|} P(R = 1|x_i)} \tag{10}$$

$$\lambda = \frac{P(R = 0|x_i)}{\sum_{i=1}^{|x|} P(R = 0|x_i) \cdot x_i} \tag{11}$$

$$P(R = r) = \frac{\sum_{i=1}^{|x|} P(R = r|x_i)}{|x|} \tag{12}$$

We can get the posterior probability $P(R = r|x_i)$ for each document's score when the parameters of the mixture model have been estimated. Intuitively speaking, the relevant documents tend to have a score with posterior probability $P(R = 1|x_i) > P(R = 0|x_i)$, together with equations above and a series of mathematic derivation, we could get $x_i > \theta$ where

$$\theta = \begin{cases} \frac{b - \sqrt{\Delta}}{a} & \text{if } \Delta > 0 \\ +\infty & \text{if } \Delta < 0 \end{cases} \tag{13}$$

$$\Delta = b^2 - ac \tag{14}$$

$$a = \frac{1}{\sigma^2} \tag{15}$$

$$b = \frac{\mu}{\sigma^2} + \lambda \tag{16}$$

$$\rho = \frac{P(R = 1)}{P(R = 0)} \tag{17}$$

$$c = \frac{\mu^2}{\sigma^2} - 2\log(\frac{\rho}{\lambda\sqrt{2\pi}\sigma}) \tag{18}$$

Recent work shows that Gamma distribution would be a more proper fit of the non-relevant documents' scores [9]. However, there is no closed form solution for parameter estimation of Gamma distribution so numerical method is needed. We kept Exponential distribution for simplicity.

3 Experiments

In this section, we briefly describe the experimental environment we used to produce our submissions at first, and then we presented the evaluation results of our submitted runs. Finally, we will discuss more about our methods and results by conducting more experimental runs.

3.1 Experimental Setup

Data.
We downloaded about 1.16 million tweets but the number of tweets used for indexing was only about 6 million. Those types of tweets considered as noise and thus filtered are described below.

- The null tweets are tweets without any content, and will be judged as non-relevant.
- The tweets downloaded with a redirected HTTP code 302 are considered as retweets and

removed as they would be judged as non-relevant.

- There is another type of retweet with sign "RT" in their contents. Although the description before "RT" is considered as the tweet's novel information, it is so little that we consider them as noise.
- This year's track is focused on English, so all the non-English tweets are judged as noise.

Table 1 gives the number of each type of noise. Note that the number of each type of noise was calculated after the previous type of noise was filtered. For example, the number of 302-Retweets was calculated in the residual collection after the Null Tweets were removed.

Table 1. Number of different type of tweets

#Total Tweets	#Null Tweets	#302-Retweets	#RT-Retweets	#Non-English Tweets	#Indexed Tweets
16,141,812	1,204,053	1,068,257	1,528,385	6,368,274	5,952,843

We did a quick and dirty preprocessing to the collection after the noisy tweets were removed, including stem and stopword removing.

Evaluation Measure.

The standard measure in this year's track is the precision of the top 30 results for each query, i.e. P@30. We also provided Mean Average Precision (MAP) and R-Precision for comparison.

Constraints.

We didn't use any future or external resources in our experiments, even for the web pages linked by the URLs in the tweets' content.

All of our experiments satisfied the strict real-time search conditions. Tweets were dynamically indexed so all the retrieved tweets were guaranteed to be created before the incoming of the query, and the words' statistics like IDF were calculated without any future evidence.

3.2 Results and discussion

We submitted two runs in this year's track. These two runs only differed in the scoring step, with one run (*hitWIt*) used the basic language model scoring function and the other run (*hitWId*) had the score decay factor incorporated.

We retrieved at most 2,000 tweets for each query in the scoring step, and then the EM algorithm was executed with the scores of these tweets in order to determine the re-ranking threshold. We will discuss more about the effectiveness of the threshold automatically selected later.

The number of pseudo feedback documents *fbDocs* was 20, number of terms selected from the feedback documents for query expansion *fbTerms* was 10. The smoothing parameter μ in language model was 20. The parameter τ in the score decay model was 7 days.

Table 2 gives the evaluation results of our submissions compared with the disjunctive baseline run provided on the track's page[1].

Table 2. Result of submitted runs and the disjunctive baseline run

Measure	All Relevance Judgments			High Relevance Judgments		
Run	P@30	MAP	R-Precision	P@30	MAP	R-Precision
hitWIt	**0.3973**	**0.3189**	**0.3777**	**0.1354**	**0.2405**	**0.2551**
hitWId	0.3340	0.2727	0.3395	0.1263	0.2259	0.2392
Baseline	0.0986	0.1411	0.1486	0.0384	0.1616	0.1419

[1] http://trec.nist.gov/act_part/tracks.new11.html

Both of our submitted runs have statistical significant improvements over the disjunctive baseline run. But unexpectedly, our run with score decay factor performs worse than the run without it.

We considered the reason of the inferior result of *hitWId* as an inappropriate selection of decay parameter τ and conducted more runs to justify the effect of score decay factor. We kept other parameters constant and obtained several results based on different values of τ.

From figure 3 we can easily find out that our choice of τ in the run *hitWId* was far away from the optimal value. The performance of runs with score decay are able to outperform or at least equal to the run without score decay (*hitWIt*) when τ is greater than 40 days. The optimal performance is achieved when τ is equal to 90 days (the run *BestDecay*), and it outperforms the run *hitWIt*, as shown in table 3.

We will also investigate the effectiveness of our threshold selection method deeper. We conducted runs with threshold fixed manually that gives exact *t* documents for each query, i.e. we use a top-t manner threshold. We compare the performances of these runs with the automatically selected threshold run *hitWId*.

The comparison result is show in figure 4. The run with adaptive threshold selection method constantly outperforms other runs in both MAP and R-Precisio. Obviously, the maximal value of P@30 is achieved when *t* is set as 30, but the other two measurements (MAP and R-Precision) are not optimal in this situation.

The average number of documents returned for each query of *hitWIt* is 95. But the performance of the run *Fixedt95* with a fixed number of 95 documents returned for each query is apparently worse than *hitWIt*, as shown in table 3. It means that when the same number of documents for the query set is returned, the result generated with adaptive threshold selection algorithm is significantly better than the one with threshold set as a fixed value for all queries.

Table 3. Comparison with runs of best decay paramter τ = 90 **and fixed threshold t** = 95

Measure	All Relevance Judgments		
Run	P@30	MAP	R-Precision
BestDecay	**0.3993**	**0.3194**	**0.3783**
hitWIt	0.3973	0.3189	0.3777
Fixedt95	0.3429	0.2878	0.3538

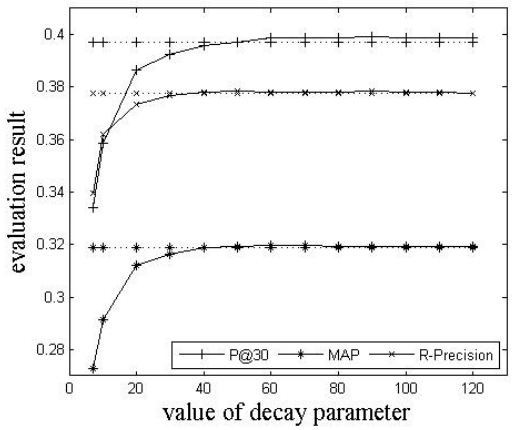

Fig. 3. Comparison with different decay parameter

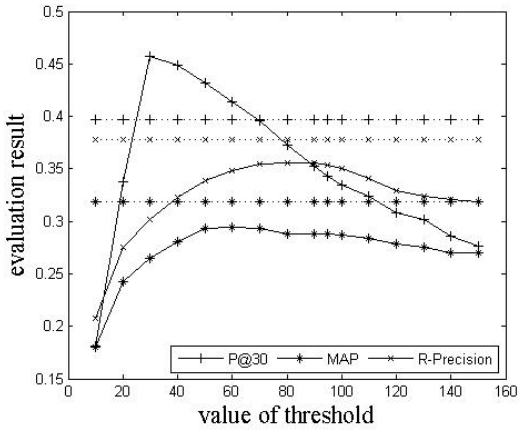

Fig.4. Comparison with fixed threshold

4 Conclusion and Outlook

We described our participation in TREC 2011 Microblog track. We submitted two runs both generated through a three-step procedure including scoring, threshold selection and re-ranking. These two runs differed only in the first step that documents in one run were scored by the basic language modeling

approach, and the other incorporated a score decay factor. We computed a re-ranking threshold for each query with the documents scored in the first step, through the score distribution method. It models the relevant documents' scores as Gaussian distribution and the non-relevant documents' scores as Exponential distribution, and then the threshold is calculated as the intersection of the curves of these two distributions. Finally, we re-rank the documents with scores higher than the threshold reverse chronologically, as required by this year's track.

Both of our submissions greatly outperform the disjunctive baseline run. With a meticulously tuned decay parameter, the performance of our run with score decay factor could be better than the basic language model run. We also demonstrated the effectiveness of our adaptive threshold selection method by comparing our automatically selected threshold run to those runs with threshold set manually as an identical value to each query. The comparison result shows that the performance of the run with the adaptive threshold selection method is constantly superior in both MAP and R-Precision to those runs with manually set threshold. It also demonstrates that the threshold selection method significantly outperforms fixing threshold manually in all three measurements when the number of documents returned for all queries is the same.

We also tried a document prior before we submitted our runs. The prior was calculated by the information in the collection [11], like the number the tweet was retweeted, the number the tweet was replied and the influence of the tweet's user [11, 12]. But unfortunately this prior brought nothing but huge noise made the performance inferior. We abandoned this run in our submission. We didn't investigate too much about the reason of this prior's poor performance, so an attempt to advance this idea may be an interesting future work.

5 References

[1] J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In Proceeding of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998.

[2] A. Arampatzis, J. Beney, C. Koster, and T. van der Weide. Incrementality, Half-life, and Threshold Optimization, for Adaptive Document Filtering. In Proceeding of the Nineth Text Retrieval Conference (TREC-9), Novemeber 12-16 2000.

[3] T. Strohman, D. Metzler, H. Turtle and W. B. Croft. Indri: A Language-Model Based Search Engine for Complex Queries (extended version). CIIR Technical Report, 2005.

[4] D. Metzler, V. Lavrenko, and W. B. Croft. Formal Multiple-Bernoulli Models for Language Modeling. In Proceeding of the 27th annual international ACM SIGIR Conference on Research and Development in information retrieval, 2004.

[5] C. Zhai and J. Lafferty. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In Proceeding of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001.

[6] V. Lavrenko and W. B. Croft. Relevance Based Language Models. In Proceeding of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001.

[7] R. Manmatha, T. Rath, and F. Feng. Modeling Score Distributions for Combining the Outputs of Search Engines. In Proceeding of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. Journal of Royal Statist. Soc. B, 39:1-38, 1977.

[9] E. Kanoulas, K. Dai, V. Pavlu, and J. A. Aslam. Score Distribution Models: Assumptions, Intuition, and Robustness to Score Manipulation. In Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2010.

[10] Y. Zhang and J. Callan. Maximum Likelihood Estimation for Filtering Thresholds. In Proceeding of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001.

[11] R. Nagmoti, A. Teredesai and M. D. Cock. Ranking Approaches for Microblog Search. In Web Intelligence, pp. 153-157, 2010.

[12] J. Weng, E. Lim, J. Jiang and Q. He. TwitterRank: Finding Topic-Sensitive Influential Twitterers. In Proceeding of the Third ACM International Conference on Web Search and Data Mining, 2010.

[13] J. Rocchio. Relevance Feedback in Information Retrieval. In The SMART Retrieval System: Experiments in Automatic Document Processing, Page 313-323. Prentice-Hall Inc., 1971.