Query-expansion Approaches for Microblog Retrieval

Sandeep Avula
School of Information & Library Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
asandeep@live.unc.edu

Jaime Arguello
School of Information & Library Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
jarguello@unc.edu

1. INTRODUCTION

The School of Information and Library Science at the University of North Carolina at Chapel Hill submitted three runs to the "Scenario B" task of the TREC 2015 Microblog Track. The task simulates a scenario where a user specifies a topic of interest in the form of a keyword query and the system produces daily updates with at most 100 tweets about the topic of interest. Systems were responsible for monitoring a stream of tweets and making daily predictions for a set of 250 interest profiles. Each interest profile was in the form of a short keyword query. Systems were asked to produce a ranking of at most 100 tweets per interest profile at the end of each day (shortly after midnight). The evaluation period extended a 10-day period from July 20 to July 29, 2015. All tweets published between 00:00:00 to 23:59:59 UTC were valid candidates for each day of the evaluation period.

Our team submitted three runs for "Scenario B". All runs were *automatic* runs and used the interest profile title field as the input query. We explored three different ways of enriching the query representation through query expansion. In two of our runs (UNCSILS_WRM and UNCSILS_TRM), we scored tweets proportional to the negative KL-divergence between a relevance model generated from an external collection and the tweet language model. These two runs mainly differ by the external collection used to generate a relevance model for interest profile query. In our UNCSILS_WRM run, we used an external Wikipedia corpus, and in our UNCSILS_TRM run, we used a corpus of tweets collected during a three-week period prior to the evaluation period. In our third run (UNCSILS_HRM), we aimed to expand the query with related hashtags.

2. SYSTEM OVERVIEW

We developed a system to run continuously throughout the 10-day evaluation period. We used the infrastructure provided by the Track organizers to sample the public Twitter stream[1], and we used Lucene[2] to implement the indexing and retrieval components.

The system operated on a daily cycle. At the start of each day, a new index was created and tweets sampled during the day were written to this index. Then at 23:59:59 UTC, the system closed the index and sequentially executed our three runs for all 250 profiles. Finally, after completing all three runs, the system created a new daily index and began writing tweets to this new index. We did not store any of the tweets that were sampled while the system executed our three runs. However, our runs only took about 10 minutes to complete, so it is unlikely that we missed indexing many relevant tweets due to this gap. Queries and documents were stemmed using the Porter stemmer [2] and stopped using the SMART stopword list.[3]

During indexing, we used the following heuristics to filter tweets not likely to be relevant to any interest profile:

- non-English tweets based on the Twitter API's language field;
- tweets with one or more swear words;
- tweets with more than three hashtags;
- tweets with more than one URL;
- tweets with more than one user mention;
- tweets without at a hashtag or URL;
- tweets that were more than 70% stopwords; and
- tweets with fewer than four tokens that were not URLs, hashtags, and user mentions.

3. ALGORITHMS

Our three runs considered different external corpora for query expansion. We first describe the base scoring function in Section 3.1 and our three query-expansion runs in Sections 3.2-3.4.

3.1 Base Retrieval

For our "base" retrieval, we scored documents proportional the negative KL-divergence between the query and document language models:

$$\mathsf{score}(Q, D) = \prod_{w \in \mathcal{V}} P(w|\hat{\theta}_D)^{P(w|\theta_Q)}.$$

Query language models were estimated using maximum likelihood:

$$P(w|\theta_Q) = \frac{\#(w, Q)}{|Q|}.$$

Document language models were estimated using Dirichlet smoothing:

$$P(w|\hat{\theta}_D) \quad = \quad \frac{\#(w, D) + \mu P(w|\theta_C)}{|D| + \mu},$$

where $P(w|\theta_C) = \frac{\#(w,C)}{|C|}$.

[1] https://github.com/lintool/twitter-tools
[2] https://lucene.apache.org/
[3] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

3.2 Wikipedia Relevance Model

For this run (UNCSILS_WRM), we scored tweets proportional to the negative KL-divergence between a relevance model generated from Wikipedia and the tweet language model:

$$\mathsf{score}(Q,D) = \prod_{w \in \mathcal{V}} P(w|\hat{\theta}_D)^{P(w|\hat{\theta}_Q^{\text{wiki}})}. \quad (1)$$

Tweet language models were Dirichlet-smoothed as described in Section 3.1 and the Wikipedia relevance model was estimated using Lavrenko's Relevance Model RM3 [1].

Our implementation of Lavrenko's Relevance Model RM3 proceeded in two steps. In the first step, we generated a relevance model from the top-t Wikipedia results produced using our base retrieval function from Section 3.1. We estimated the probability of word w in this relevance model according to:

$$P(w|\theta_Q^{\text{wiki}}) = \frac{1}{\mathcal{Z}} \sum_{D \in \mathcal{R}_t} \mathsf{score}(Q,D) \times P(w|\hat{\theta}_D), \quad (2)$$

where $\mathcal{R}_t$ denotes the top-t Wikipedia results and $\mathcal{Z}$ is a normalizing constant:

$$\mathcal{Z} = \sum_{D \in \mathcal{R}_t} \mathsf{score}(Q,D).$$

In the second step, we interpolated this relevance model (θ_Q^{wiki}) with the original query language model (θ_Q):

$$P(w|\hat{\theta}_Q^{\text{wiki}}) = \lambda P(w|\theta_Q) + (1-\lambda) P(w|\theta_Q^{\text{wiki}}). \quad (3)$$

We used the following parameter values. All document language models were smoothed using Dirichlet smoothing with $\mu = 1000$. When generating the original relevance model from Wikipedia (Equation 2), we set $t = 10$. Moreover, we used only the top-10 terms with the highest probability and re-normalized their probabilities. Finally, we interpolated the Wikipedia relevance model with the original query model (Equation 3) using $\lambda = 0.50$.

3.3 Twitter Relevance Model

For this run (UNCSILS_TRM), we build a relevance model from a corpus of tweets gathered during a three-week period before the evaluation period (from June 18 to July 12, 2015). Similar to the Wikipedia Relevance Model approach, we scored tweets proportional to the negative KL-divergence between a relevance model generated from our Twitter corpus and the tweet language model:

$$\mathsf{score}(Q,D) = \prod_{w \in \mathcal{V}} P(w|\hat{\theta}_D)^{P(w|\hat{\theta}_Q^{\text{tweet}})}. \quad (4)$$

Our implementation of Lavrenko's Relevance Model RM3 proceeded as described in Section 3.2. That is, we first estimated a relevance model from our corpus of tweets (similar to Equation 2) and then interpolated this relevance model with the original query model (similar to Equation 3).

There was only one main difference. Our corpus of tweets had a large number of duplicates, which resulted in relevance models with only a few terms. In order to discover a greater number of terms related to the original query, we removed duplicates from $\mathcal{R}_t$ by going down the ranking and filtering tweets with a Jaccard coefficient greater than or equal to 0.70 compared to a higher ranked tweet.

We used the same parameter values used in the previous run. All document language models were Dirichlet-smoothed with $\mu = 1000$. Furthermore, when generating the relevance model from our corpus of tweets, we only used the top-10 unfiltered tweets ($t = 10$) and the top-10 terms (after re-normalizing their probabilities). We interpolated the relevance model with the original query model using $\lambda = 0.50$.

3.4 Hashtag Expansion

In this approach (UNCSILS_HRM), we aimed to expand the original query with relevant hashtags. We used the same corpus of tweets used in our Twitter Relevance Model approach.

The approach proceeded as follows. Our first goal was to score hashtags based on their relevance to the original query. To this end, we constructed an index of hashtag pseudo-documents, where the text associated with each hashtag pseudo-document was gathered from all tweets containing the hashtag. Then, we built a relevance model of hashtags based on the normalized query-likelihood score from each hashtag's language model. The probability of hashtag h in this relevance model is given by:

$$P(h|\theta_Q^{\text{hash}}) \quad = \quad \frac{1}{\mathcal{Z}} \prod_{q \in Q} \frac{\#(q,H) + \mu P(q|\theta_{C_H})}{|H| + \mu}.$$

Here, H denotes the pseudo-document associated with h, C_H denotes the collection of hashtag pseudo-documents, and $\mathcal{Z}$ is a normalizing constant. In practice, we only considered the top-10 hashtags and assigned to other hashtags a zero probability.

Similar to the previous approaches, we then interpolated our hashtag relevance model with the original query model:

$$P(w|\hat{\theta}_Q^{\text{hash}}) = \lambda P(w|\theta_Q) + (1-\lambda) P(w|\theta_Q^{\text{hash}}), \quad (5)$$

and scored tweets proportional to the negative KL divergence between this interpolated model and the tweet language model:

$$\mathsf{score}(Q,D) = \prod_{w \in \mathcal{V}} P(w|\hat{\theta}_D)^{P(w|\hat{\theta}_Q^{\text{hash}})}. \quad (6)$$

3.5 Redundancy Filtering

The Track guidelines specified that the evaluation would punish redundant tweets. To address this, for each runs' daily output, we proceeded down the ranking and filtered tweets with a Jaccard coefficient greater than or equal to 0.70 compared to a higher-ranked tweet. For each run, the system returned the top-100 unfiltered tweets.

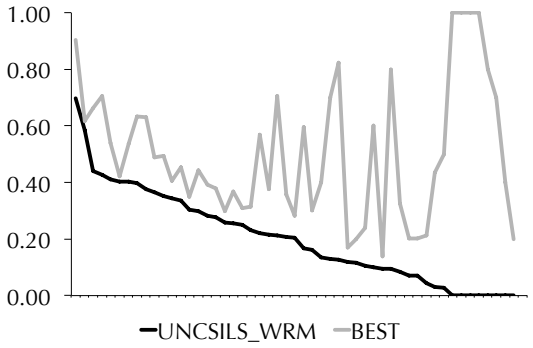
4. RESULTS

The results for our three "Scenario B" runs are presented in Table 1 in terms of NDCG. The results are averaged across the 51 topics that were included in the evaluation. We tested statistical significance using the approximation of Fisher's randomization test described in Smucker *et al.* [3]. Our Wikipedia Relevance Model approach (UNCSILS_WRM) outperformed our other two runs by a statistically significant margin. The other two runs (UNCSILS_TRM and UNCSILS_HRM) were statistically equal.

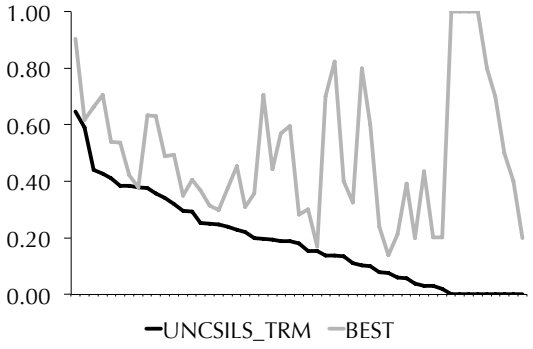
Table 1: Results in terms of NDCG. Symbols $^{\blacktriangle}$ and $^{\triangle}$ denote a statistically significant improvement over UNCSILS_TRM and UNCSILS_HRM, respectively ($p < .05$)

UNCSILS_TRM	0.189
UNCSILS_HRM	0.190
UNCSILS_WRM	$0.205^{\blacktriangle\triangle}$

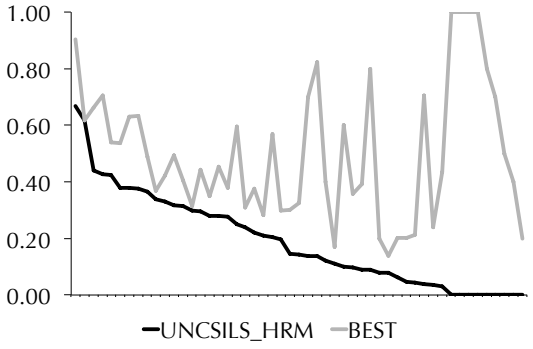
Preliminary results also show that all three runs were far from the best-performing run for most interest profiles. Figures 1.a-1.c show NDCG performance for each run compared to the best-performing run per interest profile. For each run, interest profiles are sorted in descending order of NDCG performance. As the Figures clearly show, for most interest profiles, all three runs underperformed the best run by a wide margin.



(a) UNCSILS_WRM



(b) UNCSILS_TRM



(c) UNCSILS_HRM

Figure 1: NDCG performance per interest profile compared to the best-performing run for that profile. For each run, profiles are sorted in descending order of NDCG performance.

5. REFERENCES

[1] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127. ACM, 2001.

[2] M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., 1997.

[3] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 623–632. ACM, 2007.