IBM at TREC 2012: Microblog Track

Myle Ott,[*] Vittorio Castelli, Hema Raghavan, Radu Florian
IBM T.J Watson Research Center, Yorktown Heights, NY 10598
`myleott@cs.cornell.edu`, `{vittorio,hraghav,raduf}@us.ibm.com`

Abstract

This paper describes IBM Research's approach to the real-time ad-hoc retrieval task of the TREC 2012 Microblog Track. As this was our first time participating in the Microblog Track, our primary goal was to build a strong baseline system on which to later improve.

In particular, our system implements a Learning-to-Rank framework on top of a full-dependence Markov Random Field (MRF) retrieval model (Metzler and Croft, 2005). We chose LambdaMART (Ganjisaffar et al., 2011) as our Learning-to-Rank learner, and trained it using over 50 features, including tweet, query and user-specific features. Our cross-validation results on the 2011 Track queries show that our system performs comparably to the top-performing systems in the TREC 2011 Microblog Track.

1 Introduction

Microblogs, such as those found on Twitter, are an increasingly popular medium for online content sharing. However, microblogs can pose numerous challenges to standard search and Information Retrieval (IR) techniques, largely due to the short and informal nature of their contents. Accordingly, there is a growing need for IR systems that can address the unique challenges associated with searching microblog collections.

[*]Work done while the author was a summer intern at IBM Research.

Unfortunately, restrictions on sharing of microblog data has made it challenging to evaluate or compare the performance of IR systems on microblogs. This changed in 2011 with the introduction of the TREC Microblog Track, where researchers were allowed to crawl and evaluate their systems on a shared microblog corpus, *Tweets11*, containing nearly 16 million tweets.

The Microblog Track has been continued for TREC 2012, and in the following sections we will describe our efforts to build a strong baseline retrieval system for this year's real-time ad-hoc retrieval task. Note that while this year's Track also includes a new real-time filtering pilot task, since this was our first year participating in the Track, we chose to focus our efforts exclusively on the ad-hoc retrieval task.

The rest of this paper is organized as follows. First, in Section 2, we describe our process for collecting and preprocessing the data. Then, we describe our ad-hoc retrieval approach in Section 3. Finally, in Section 4, we present and discuss our results.

2 Data

In this section, we describe our handling of the various data utilized by our system.

2.1 Tweets

For TREC 2012, the organizers chose to use the same *Tweets11* corpus used for TREC 2011. This corpus originally contained a random sample of approximately 16 million tweets from a two week time period spanning January 24, 2011

Table 1: Corpus statistics.

Number of tweets:	14.1m
Vocabulary size:	812k
Unique hashtags:	220k
English tweets:	62%
Retweets:	10%
Replies:	20%
Compressed size:	7GB

to February 8th, 2011. The corpus was made to be as realistic as possible, and therefore no spam removal was performed prior to release.

The dataset was then distributed to participants in one of two ways. For those participants with greater access to the Twitter API, the corpus could be downloaded in the original (and richer) JSON format. For most participants, however, a more basic version of the corpus was obtained by crawling Twitter and extracting the tweets from the HTML. This was made possible through use of a custom crawler released by the TREC organizers last year.[1]

Unfortunately, changes to Twitter's HTML in early 2012 broke the original crawler, forcing teams that joined the Microblog Track in 2012 to first fix the crawler. Fortunately, the changes to Twitter's HTML included embedding the richer JSON tweet representation into each page's HTML, making it possible for all teams to download the JSON version of the corpus.[2]

In addition to providing richer meta-data for each tweet, extracting the embedded JSON reduced the size of each block of 10,000 tweets from over 150MB to less than 5MB. More detailed corpus statistics appear in Table 1.

2.2 Queries

For the 2011 task, the TREC organizers created 50 topics (queries), representing realistic information needs that people might have on Twitter. To evaluate how well the participating systems did at returning relevant results for these queries, TREC pooled the result sets and had human judges assess tweet relevance. This assessment was done on a four-point scale, where -2 indicates `spam`, 0 indicates `not-relevant`, 1 indicates `relevant`, and 2 indicates `highly-relevant`.

For the 2012 task, a new set of 60 topics (queries) were chosen, allowing participants to use the 2011 queries and relevance judgments to improve their systems. In particular, we chose to use the relevance judgments to train a Learning-to-Rank system, which we describe further in Section 3.

2.3 Pre-processing

Once we completed downloading the corpus, we applied several pre-processing steps to the data, described here:

1. **Removal of non-English tweets:** The TREC evaluation guidelines define non-English tweets as *de facto irrelevant*. Therefore, we ran language identification on each tweet,[3] and deleted from our collection all tweets that were assigned a 0-probability of being English.

2. **Removal of retweets:** The TREC evaluation guidelines also define all retweets as *de facto irrelevant*. Therefore, per the Twitter Field Guide,[4] we deleted from our collection all tweets that contained *retweeted_status* in the status portion in their JSON. Among the remaining tweets, we additionally deleted any text that followed an `RT` token (as well as the `RT` token itself), since such text typically corresponds to quoted (retweeted) material.

3. **Conversion to ASCII:** Many tweets contain unusual or non-standard characters, which can be problematic for down-stream

[1]Available here: `https://github.com/lintool/twitter-corpus-tools`.

[2]Code to extract the embedded JSON was first made available by `spacelis` here: `https://github.com/spacelis/twitter-corpus-tools`; this code was later improved by our team and made available here: `https://github.com/myleott/twitter-corpus-tools`.

[3]We used Nakatani Shuyo's Twitter-specific language ID code, available here: `https://github.com/shuyo/ldig`.

[4]`https://dev.twitter.com/docs/platform-objects`

processing. To address these issues, we used a combination of BeautifulSoup[5] and Unidecode[6] to convert and transliterate all tweets to ASCII.

4. **Tokenization:** Next, we tokenized each tweet using a Twitter-specific tokenizer, *tweetmotif*[7], which properly handles most @-mentions, URLs and emoticons. Note that we modified the tokenizer to additionally recognize the heart (`<3`) emoticon.

5. **Removal of empty tweets:** After completing all of the other pre-processing, we deleted any empty tweets.

2.4 Indexing

After pre-processing, we used the Indri search engine [8] (Metzler and Croft, 2004) to index our corpus for fast retrieval. However, since each query has a query time associated with it, and systems are not permitted to return tweets that are broadcast after that time, we additionally chose to build one index per query instead of a single, large index.

Each query index contained all and only those tweets broadcast between the query tweet time and the chronologically previous query's tweet time. The first index contained all tweets from the beginning of the corpus until the first query's tweet time. Additionally, we utilized Indri's built-in Porter stemmer at index time, in order to increase the engine's recall.

3 Experimental Setup

Given an information need, q, issued at a specific time, t, the TREC 2012 real-time ad-hoc retrieval task is to return an ordered list of $10,000$ tweets, ordered according to two factors: (a) relevance of the tweet to the query q; and (b) chronological closeness (*recentness*) of the tweet's broadcast time to the query time t. In particular, tweets that are more relevant to q and more recent w.r.t. t should be ordered higher in the list than tweets that are less relevant and less recent. The balance between relevance and recency is captured in the task evaluation, which includes thresholding the returned results and re-sorting the remaining results (reverse chronologically) before computing each metric. The metrics for this year's task are: P@30, MAP and AUC (ROC).

Since this was our first year participating in the Microblog Track, we largely ignored these task-specific considerations, and instead focused on building a strong baseline retrieval system on which to later improve. To this end, we implemented a standard Learning-to-Rank framework. In particular, for each query, q, issued at time, t, we retrieved a result set using the following process:

1. Retrieve a large, high-recall result set, `baseline`, using the Indri search engine[9] (Metzler and Croft, 2004), which was a popular choice at TREC 2011. In particular, following Metzler and Cai (2011), we utilize the full-dependence variant of Indri's MRF retrieval model (Metzler and Croft, 2005) to retrieve 20,000 results for each query, and additionally use Indri's built-in Porter stemming to increase recall.

2. If q is a training query, i.e., a 2011 query, train a ranking model using the query, the `baseline` result set and associated TREC 2011 judgments, and all chronologically previous training data. We use LambdaMART[10] (Ganjisaffar et al., 2011) to train our ranking models, and tune parameters as outlined in Section 3.2.

3. If q is a test query, i.e., a 2012 query, use the chronologically previous ranking model to re-rank the `baseline` result set, and output the top k re-ranked results. Following the TREC evaluation guidelines, we set $k = 10,000$. Note that the final output ranking is actually interpolated with the original `baseline` ranking, as outlined in Section 3.2.

[5]`http://www.crummy.com/software/BeautifulSoup/`

[6]`http://pypi.python.org/pypi/Unidecode`

[7]`https://github.com/brendano/tweetmotif`

[8]`http://www.lemurproject.org/indri/`

[9]`http://www.lemurproject.org/indri/`

[10]`http://code.google.com/p/jforests/`

ends_in_hashtag	ends_in_URL
en_prob	has_coord
has_exclamation	has_happy_emoticon
has_heart_emoticon	has_other_emoticon
has_question	has_RT
has_sad_emoticon	has_tongue_emoticon
has_wink_emoticon	is_reply
num_hashtags	num_mentions
num_URLs	num_URLs_RE
tweet_length	FUTURE_num_retweets
local_day_is_{mon,tues,wed,thurs,fri,sat,sun}	
local_time_is_{morning,afternoon,evening,night}	

Figure 1: Tweet features.

Indri_rank	time_bw_tweet_and_query
{cosine,dot_product,hellinger}_sim_to_query	
{jensen_shannon,kullback_leibler}_sim_to_query	
cosine_sim_to_query_less_stopwords	
dot_product_sim_to_query_less_stopwords	
hellinger_sim_to_query_less_stopwords	
jensen_shannon_sim_to_query_less_stopwords	
kullback_leibler_sim_to_query_less_stopwords	

Figure 2: Query features.

user_default_profile	user_default_profile_image
user_geo_enabled	user_has_description
user_has_URL	user_is_translator
user_is_verified	user_lang_is_en
time_bw_account_creation_and_query	
time_bw_account_creation_and_tweet	
FUTURE_user_num_favorites	
FUTURE_user_num_followers	
FUTURE_user_num_friends	
FUTURE_user_num_statuses	
FUTURE_user_num_lists	

Figure 3: User features.

3.1 Features

We trained our ranking models using three kinds of features, given in Figures 1, 2, and 3. Note that features prefixed by "FUTURE_" rely on "future" information and are therefore excluded, per the task guidelines, except where indicated otherwise.

3.2 Cross-validation

Parameters for LambdaMART were tuned based on cross-validation experiments on the TREC 2011 queries and relevance judgments. However, in order to abide by the real-time constraint, our cross-validation procedure was performed in an online manner. In particular, for each 2012 test query, q_{2012}, issued at time t_{2012}, we tuned the parameters of a new ranking model by 5-fold cross-validation using all 2011 queries that had issue times prior to t_{2012}.

One complication of this procedure is that the learned ranking models were often built using very little training data, e.g., if q_{2012} was issued early in the corpus timeline. To address this problem, we introduced a function, chosen in an ad-hoc manner and without tuning, to weight and interpolate the LambdaMART ranking and the original Indri ranking:

$$\text{weight(LTR)} = 0.8 * \left(1 - exp\left(\frac{2 - |Q_{train}|}{20}\right)\right)$$
$$\text{weight(Indri)} = 1 - \text{weight(LTR)},$$

where $|Q_{train}|$ corresponds to the number of queries used for training the model. This function has the effect of down-weighting the LambdaMART ranking when the model is trained on only a few queries, and up-weighting its ranking when the model is trained on many queries.

4 Results and Discussion

Our team submitted three runs for evaluation, the results of which are given in Table 2. The details of these three systems are given below:

- **IBMBaseline:** The exact ranking as given by Indri, up to rank 10,000.
- **IBMLTR:** The interpolated LambdaMART and Indri ranking, as described in Section 3, using only those features not prefixed with "FUTURE_" in Figures 1, 2, and 3, i.e., only those features that do not constitute "future information" according to the task guidelines.
- **IBMLTRFuture:** The same as **IBMLTR**, but additionally includes features prefixed with "FUTURE_" in Figures 1, 2, and 3.

The results suggest that Learning-to-Rank (LTR) does improve performance over the baseline retrieval model given by Indri. Indeed, the

Table 2: Results. Bold entries in each row correspond to best performing run. "Relevant" and "Highly Relevant" refer to the relevance cutoff used for evaluation.

Metric	IBMBaseline	IBMLTR	IBMLTRFuture
P@30 (Relevant)	0.3808	**0.4136**	0.4090
P@30 (Highly Relevant)	0.2028	0.2237	**0.2254**
MAP (Relevant)	0.2620	0.2630	**0.2731**
MAP (Highly Relevant)	0.1968	0.1932	**0.2018**

LTR runs outperform the Indri baseline in all but one case (MAP, Highly Relevant, LTR w/o future features). We also observe that the "future" features are very helpful and improve LTR performance in all but one case (P@30, Relevant). We suspect that this is primarily due to the inclusion of the features `num_retweets` and `user_num_followers`, though this should be explored further in future work through a feature ablation study. Future work may additionally include features that address the vocabulary mismatch problem, potentially using topic models, as discussed in Ramage et al. (2010).

References

Yasser Ganjisaffar, Rich Caruana, and Cristina Lopes. 2011. Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, SIGIR '11, pages 85–94, New York, NY, USA. ACM.

D. Metzler and C. Cai. 2011. Usc/isi at trec 2011: Microblog track. In *Proceedings of the Text REtrieval Conference (TREC 2011)*.

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

D. Metzler and W.B. Croft. 2005. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM.

D. Ramage, S. Dumais, and D. Liebling. 2010. Characterizing microblogs with topic models. In *International AAAI Conference on Weblogs and Social Media*, volume 5, pages 130–137.