Online Topic Modeling for Real-time Twitter Search

Christan Grant, Clint P. George, Chris Jenneisch, and Joseph N. Wilson
University of Florida, Dept. of Computer Science
Gainesville, Florida, USA
{cgrant, cgeorge, chris, jnw} @cise.ufl.edu

October 24, 2011

Abstract

This paper discusses the work done by a team at the University of Florida for the TREC 2011 Microblog Track. To build a real-time microblog search engine we rely on topic modeling for our search. To facilicate our algorithms we bundle similar tweets together in what we call supertweet generation. We perform online inference and offline inference depending on the time frame of the topical query. In this paper we discuss our techniques, challenges, future work, but not the evaluation of our results. We offer this notebook entry to invite future discussion and collaboration.

1 Introduction

In this paper, we discuss the task of real-time microblog search. Given a topic (a time-stamped query) our goal is to find the interesting and relevant microblog entries (tweets) in the data set. This is difficult for many reasons. First, each microblog entry is short, this means we don't have as much context as in most text documents. Second, many communications do not provide any useful information or are spam. Adding the scale of the data set and real-time operation to the equation creates an amalgam of difficulties.

Our system uses a topic modeling framework for querying in the large *corpus* (document collection) of tweets. Topic models represent documents as bags of words without considering word order as being of any importance. These models have the ability to represent large document collections with lower dimensional topics, which represent clusters of similarly behaving words. In addition, the document words are assumed to be generated from topic specific multinomials and the topic for a particular word is chosen from a document topic mixture. These topics are assumed to be generated over the corpus vocabulary from a Dirichlet distribution. Blei et al. gives a detailed description of this generative process and its assumptions [2].

Topic models have a natural way to encode assumptions about observed data. Their analysis is dependent upon exploring the posterior distribution of model

parameters and hidden variables conditioned on observed words. The model parameters are corpus-level topics or concepts—sets of words with corresponding probabilities—and document-level topic mixtures.

In our approach, we use topic models to discover topics in the tweets and compare them with the estimated topics from the online query. The estimated topics are further used to rank relevant tweets in the corpus. We use both online and offline topic modeling to facilitate real-time search. If on considers the tweets as elements of a stream, we capture hourly batches of tweets and perform topic modeling on each batch independently. We perform *online* topic modeling for recent time intervals that have not yet been included in a batch.

Instead of looking at each tweet individually, we collect tweets into buckets of other similar tweet in a process called *supertweet generation*. We construct these large supertweet documents to help get around the problems associated with analysis of numerous small documents.

Given a query topic (a small set of search terms) and a time stamp we find the query topic distribution with respect to the documents before last hourly break point, and then the topic distribution for all the previous hourly tweet batches. We can then provide rankings for individual supertweets using KL divergence of the topic distributions.

In the following sections we describe the supertweet generation process, inference techniques, and ranking procedures. Then we will discuss the approaches we take.

2 Supertweet generation

One of the problems with considering tweets as documents is that they contain a very small number of words (being restricted to 140 characters). To tackle this problem we form *supertweets* (collection of tweets) from individual, similar tweets and use them for our analysis. Each supertweet can be a collection of tweets or a single tweet (when we cannot group it with other tweets) aggregated based on a group of similarity features.

2.1 Similarity features

To group tweets into supertweets, we defined several similarity measures based on tweets' textual content after removing stop words. Our stop words list includes regular Internet stop words, foreign words and swear words.

We identified a set of features found in tweets and associated each feature with a positive non-zero weight in order to prioritize them.

The weights for each feature were selected experimentally. The limited time period for this experiment precluded the development of a more principled approach to feature selection. Table 2.1 lists the features and their weights.

Feature	Weight
hashtag	1.5
web links	1
retweets	0.9
usernames	0.8
bigrams	0.7

Table 1: Features and weights

The highest priority is assigned to hashtags as they represent tweets that humans have deemed to have common topics. Prior work supports the use of hash tags as a basis for aggregating tweets [10]. Web links are also used for tweet aggregation on the basis that identical links should have similar content. If a tweet is a retweet, a lower weight is assigned because we deem them as not being original. *Bigrams* (consecutive word pairs) are given the lowest rank as there is a high chance of getting a matching bigram sequence as compared to other features.

Based on the above five features, we define the similarity between two tweets t_1 and t_2 as:

$$F(t_1, t_2) = \sum_{i=1}^{5} W_i * N_i(t_1, t_2)$$

where W_i is the weight for feature i, and N_i is the number of occurrences of common features between two tweets t_1 and t_2.

We can also use this measure to find the similarity between a supertweet and tweet. In this case, using the average F value for the tweet each of the tweets in the supertweet.

2.2 Supertweet Generation

The supertweet generation process aggregates tweets that are similar to the tweets currently associated with a supertweet. We use a threshold on the similarity measure to decide whether a new tweet can be added to a supertweet.

Since there are many tweets, we cannot consider the whole corpus for forming supertweets. Instead we use a *sliding window* approach in which we represent a sliding window as a priority queue Q. Q is a fixed length queue and contains supertweets in the order of their recency of formation. Whenever Q is full, we remove the oldest supertweet and insert a newly formed supertweet. The removed supertweet would be added to our final list of supertweets that we maintain in the database. Algorithm 1 describes our method for supertweet generation.

Algorithm 1 Supertweet Generation

```
for each tweet T_i in the corpus do
  if Q is empty then
    T_i is inserted into Q
  else
    Find the best supertweet ST in Q with which T_i has highest non-zero similarity measure
    greater than threshold
    if there exists a supertweet ST then
      Merge T_i in ST
    else
      Create a new supertweet with T_i
      if Q is full then
        Remove oldest supertweet from Q and add to list of supertweets
        Insert the newly created supertweet in Q
      else
        Insert in Q
      end if
    end if
  end if
end for
Copy remaining supertweets in Q to list of supertweets
```

3 Topic Modeling and Inference

3.1 Topic modeling and document analysis

Topic models such as Latent Dirichlet Allocation (LDA) [3] and Hierarchical LDA [11] are well-known for exploratory and predictive analysis of text. They define *topics* as distributions over the words in a vocabulary and *documents* as being generated by mixtures of these topics. The words of individual documents are drawn independently from document topic mixtures (mixtures of topic multinomials) [3]. Topic models represent document (message) words in a bag-of-words format without considering word order to be of any particular importance. These models support powerful methods for dimensionality reduction of large, unstructured document collections. In addition, we can use the document posteriors for information retrieval and classification.

Topic models are conventionally designed for fixed [3] or varying numbers of topics (nonparametric) [11] usually on discrete-time document collections. In the case of twitter conversations, the number of possible topics or concepts is unbounded. We can also assume that the topics associated with tweets emerge, evolve, and disappear over time. For this reason, we used the hierarchical Dirichlet process based LDA [11] to extract topics from the offline data store (which contains tweets from a two week time period).

3.2 Batch topic learning

Topic models provide a natural way to encode assumptions about observed data and their analysis is dependent on exploring the posterior distribution of model parameters and hidden variables conditioned on observed words. The model parameters are corpus-level topics or concepts, i.e., sets of words with corresponding probabilities, and document-level topic mixtures. Our twitter data analysis is largely based on these parameters. Performing maximum a-posteriori (MAP) estimation on the LDA model is intractable [3, 11]. Thus, people typically use relatively efficient sampling approaches and optimization approaches for inference. Sampling approaches are usually based on Markov Chain Monte Carlo [7] methods, in which we define a Markov chain whose stationary distribution is the posterior of interest. Most common optimization approaches for topic modeling are based on variational Bayes (VB), which optimizes a simplified parametric distribution close to the posterior on Kullback-Leibler divergence [3]. Variational Bayes estimation of the posterior usually introduces bias, where as MCMC-based Gibbs samplers generate independent samples from the posterior [8]. In this project, we used a Gibbs sampling based MCMC to estimate the parameters of the hierarchical LDA [11]. We used the hierarchical LDA package developed by Chong Wang[1] in our batch topic learning.

Even though topic models are very useful in dimensionality reduction, clustering, and analysis of large document collections [3, 1], their topic estimation process is computationally very expensive and is currently inconceivable for huge collections [13]. In this project, we have a data set of $\sim$ 16 million tweets. It is nearly impossible to run topic inference for this entire data set in the time frame of our investigation. Our approach was to split the data set into the batches of tweets drawn from one-hour time periods and perform hierarchical LDA on the individual batches. Also, we deleted those extracted topics that did not have a minimum threshold of word associations in the whole document collection.

There are some natural limitations that prevent us from directly applying topic modeling to the twitter data available. First, twitter messages are usually small (being restricted to 140 characters), which is substantially different from conventional information text retrieval and mining problems. Second, within this short text people communicate rich meanings. For example, for long URLs we use URL shortening services, and for specific events or topics we define # tags. We found that the restricted lengths of tweets prevents us from exploiting their full potential in a topic-modeling setting. Our experiments showed that aggregating tweets (section 2) to train the topic model can obtain an improved set of topics. The research work of Hong et al. [9] discusses similar observations on a different Twitter dataset. In addition, we consider special tags (e.g. #, @) and URLs as if they are individual words, allowing them to group into different topical clusters.

[1] http://www.cs.princeton.edu/ chongw/resource.html

3.3 Online document inference

In this project, we used tweets' and super tweets' topic mixtures to find tweet similarities. By online document inference, we mean inferring topic mixtures during query time. Conventional topic learning algorithms [3, 11] are designed to run on collections of documents (i.e., in batch mode). In our recent work, we developed a system [5, 6] that can infer topic structure for newly encountered documents without retraining the estimated topic model. This model is based on a fast hybrid Metropolis search [6] and can use the learned models from any batch topic modeling algorithms.

To enable online processing for a given query, we first find out the corresponding hour batch to which it belongs by consulting its time stamp. Second, at query processing time we infer topic distributions for the tweets that belong to the query's batch and have time stamps earlier than the query time stamp, using the hierarchical LDA. Finally, we calculate the query topic mixtures based on all the batches' pre-computed topic distributions using the hybrid Metropolis topic search. Once we have the query topic mixtures, we run our query matching algorithms that are explained in the next section to display relevant tweets to the end user.

4 Ranking

Our ranking method is a straightforward filtering method. First we narrow down the set of possible tweets. We have a query and a topic distribution from the query. We take the top k supertweets from each batch query using the KL-divergence between the query topic distribution for the specific batch and each supertweet in the batch. Then for each tweet we compute a weighted distribution over several parameters. These parameters all contribute to the idea of interestingness.

Supertweet score We keep the tweet's supertweet divergence. A high score here means there is some contextual relevance between the query and this tweet neighborhood.

Recency We give an exponential back-off score so that we reward more recent tweets. The recency is given by $1000/|t_1 - t_2|$, where t_1 is the time associated with the query and t_2 is the timestamp associated with the tweet.

Word Length We say a helpful tweet is one at about 20 words. We create a Gaussian function to give the most weight to tweets around this value. The word length function is given by $\frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ where x is the number of tokens in the tweet, μ is 20, and σ is 5.

Important Words This looks at word matches. The longer the term, the larger the value we give. This function assigns higher weights to longer matching words in the candidate tweet.

Jaccard similarity coefficient We calculate the simple term matching methods.

Fuzzy Wuzzy Measure We implemented a string similarity metric developed by seat geek for matching short snippets of text [2]. We liken this algorithm to a Jaro-Winkler algorithm.

These ranking function were implemented in Scala and are easily parallelizable.

5 Discussion

We spent about two-thirds of our time obtaining and cleaning the dataset. The size of the dataset gave us trouble when attempting to download tweets across the network. We were finally able to obtain the full data set using the twitter-corpus-tools[3]. Next, we removed all foreign language tweets from the data set. Because we used the HTML scrape of the data set, the language specified from the HTML document webpage could not be trusted. So, we created two Bloom filters to help us filter languages. We filled the first Bloom filter with with dictionaries of several different languages. We filled the second Bloom filter with terms from English and *SMS* short codes (e.g. LOL). Then we went through all tweets and removed any tweet that contained a word that was both in the foreign language bloom filter and not in the English Bloom filter (with the confidence of a particular threshold). We found this technique to be fast and accurate with a low number of false positives.

We faced several problems when testing our method on the evaluation queries. First, we found that the number of terms in each the evaluation query was too small to generate a statistically interesting topic distribution (e.g. *Toyota Recall*). This prevented us from effectively exploiting similarity based on topic distributions with some queries. Our method was more successful with longer queries containing more diverse search terms.

Second, during the preprocessing step, we removed infrequently occurring words from the dataset. Our dataset pruning can remove search terms needed to satisfy queries associated with infrequently mentioned topics. In addition, we did not perform any stemming, lemmatization, or other type of dictionary-based preprocessing on the tweets or queries. Our operating principle was that similar meaning words would be clustered together by LDA during topic modeling based retrieval. This assumption may not hold for infrequently occurring words and word forms even if synonyms or alternate forms appear frequently.

6 Conclusion and Future Work

We presented a method to perform real-time search for the TREC 2011 Microblog Track. We used topic modeling to extract and rank tweets from a large dataset.

[2] https://github.com/seatgeek/fuzzywuzzy

[3] https://github.com/lintool/twitter-corpus-tools

Our method for this problem is novel but it will require further refinement to be effective.

Our first goal for future work is evaluation of the results and techniques. Our initial results were somewhat dissappointing but have led us to a better understand of some of the limitations of topic modeling. The topic multinomials we identified appeared to be quite reasonable. The choices we were forced to make in supertweet formation and feature weighting need to be considered more carefully. Second, we will look into evolving topics across time-varying batches. Other researchers have tracked topic over time [12]. Third, our features for ranking are few. In the future we will look into adding *soft tf-idf* scores and other parameters to increase accuracy [4]. To offset our shortcomings, we believe query expansion techniques will be helpful for matching queries. Finally, we plan to create a simulator so we can test our code in a real-time environment.

References

[1] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM*, 57:7:1–7:30, February 2010.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, March 2003.

[4] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, pages 1–16, 2007.

[5] C. Fuentes, V. Gopal, G. Casella, C. P. George, T. C. Glenn, J. N. Wilson, and P. D. Gader. Product partition models for dirichlet allocation. Technical Report 519, University of Florida, September 2011. The Dept of CISE.

[6] C. P. George, T. C. Glenn, C. Fuentes, V. Gopal, G. Casella, J. N. Wilson, and P. D. Gader. Topic learning and inference using dirichlet allocation product partition models and hybrid metropolis search. Technical Report 520, University of Florida, September 2011. The Dept of CISE.

[7] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

[8] M. Hoffman, D. Blei, and F. Bach. Online learning for latent dirichlet allocation. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta,

editors, *Advances in Neural Information Processing Systems 23*, pages 856–864. 2010.

[9] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 80–88, New York, NY, USA, 2010. ACM.

[10] J. Lin, R. Snow, and W. Morgan. Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 422–429, New York, NY, USA, 2011. ACM.

[11] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[12] X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 424–433, New York, NY, USA, 2006. ACM.

[13] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on KDD*, KDD '09, pages 937–946, NY, USA, 2009. ACM.