Project Term Report

Hao Peng

December 19, 2011
1 Introduction

Twitter is one of the most popular websites that provides micro-blogging services. It provides a platform for users to post messages which are known as tweets[1]. Users can also retweet (forward) other people’s tweets for sharing. Twitter has its own ranking of tweets by their popularity in terms of the number of retweets.[2] However this ranking does not take the contents of tweets into account. It cannot not provide tweets for users according to users’ interests. This project aims at building a recommendation system based on the content of tweets.

2 Background

In general, there are two approaches for recommending systems.[3] One is based on the technique of collaborative filtering, but this approach suffers from the cold start problem. The cold start problem arises when some new subject doe not have enough collaborative information. Unfortunately, a good twitter recommendation system needs to provide new and updated tweets, because the majority of the trending topics on Twitter are news[4].

The other approach, which is used in this project, is based on content filtering. This approach requires to collect extra information about users and their behaviours. In this case, this is not a problem for a tweets recommendation system, because a lots of information about users are available on Twitter. Twitter keeps tracks of information such as posted and retweeted tweets by users, geographic information of users and follow relationships between users.

Many models have been proposed to model documents. One popular topic model is Latent Dirichlet Allocation (LDA) [5]. It can capture the characters of a collection of documents and essential statistical relations for similarity and relevance judgements.

The original LDA can discover the latent patterns in documents. But if each tweet is treated as a separate document, the average number of words per document will be too small, because Twitter the maximum number of words per tweet to 140 characters. There are several ways to adapt LDA to Twitter. One way used in this project is that all tweets posted by a user
are collected and treated as a single document.[6] The LDA in this method actually captures the latent topic distributions over users instead of messages.

The generative model works as follows. (Figure 1) Given a number $U$ of users, for each user $u$, a distribution over topics, $\theta$, is drawn from a Dirichlet prior with parameter $\alpha$. For each word $w_{u,n}$ posted by that user, a topic $z$ is drawn from a multinomial distribution with parameter $\theta$. And a word is drawn from another multinomial distribution over words $\beta$ given the topic $z$.

![Figure 1: Graph Representation of LDA model](image)

3 Data Collection and Pre-processing

The data was collected using Twitter Streaming API, which gets all public tweets posted in real time. The tweets were collected from Oct. 11, 2011 to Nov. 11, 2011 with occasionally breaks due to maintenance. The region is restricted to a rectangle with two corners at $(40.32^o N, 70.05^o W)$ and $(40.53^o N, 73.4^o W)$ respectively. A total number of 324,560 tweets by 21,731 users were collected.

In the pre-processing step, users who posted less than 10 tweets in the data collected are removed. After this step, there are 276,857 tweets and 4,714 users left. In the next step, all tweets posted by the same user are gathered together and transformed into bag-of-word representation. All tweets are first tokenized. All leading and tailing punctuations are removed. All non-English words, stop words and words with frequency less than 5 are removed. After these, there are 18,196 different words left. In other words, there are 18,196 features in total. All URL links are transformed into a key word "LINK". Some keywords such as "rt", "#" and "@" in Twitter are
not specially handled in this project. There are some proposed methods to handle these keywords such as in [7]. Then the data is divided into two sets. A training data with 3,781 users and and a testing data with 933 users. In the testing data, for each user one of tweet is randomly selected and held out for testing prediction.

4 Model Training

The model is trained using variational inference described in [5]. The author’s code is used. The number of topics is unknown and needed to be determined. The training set is further divided into two sets. 20% of users were used for validation. The perplexity [5] is computed on the held-out data to evaluate the model. A lower perplexity indicates better generalization performance. For a test set of $U$ users, the perplexity is defined as:

$$\text{perplexity}(D_{test}) = \exp \left( - \frac{\sum_{u=1}^{U} \log p(w_u)}{\sum_{u=1}^{U} N_u} \right)$$

The results of perplexities for different number of topics are plotted in Figure 2. It can be seen that when the number of topics is 30 the perplexity is the lowest. Therefore the final model is trained over all the sets of the topics with the number of topics fixed to 30.

![Figure 2: Perplexity with respect to number of topics](image)
After training the final model, the top words in each topic can be calculated. Some words in one topic are closely related and forms a meaningful sets. Examples are given in Table 1.

<table>
<thead>
<tr>
<th>Topic 01</th>
<th>Topic 04</th>
<th>Topic 16</th>
<th>Topic 18</th>
<th>Topic 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>rain</td>
<td>rt</td>
<td>LINK</td>
<td>game</td>
<td>ows</td>
</tr>
<tr>
<td>today</td>
<td>LINK</td>
<td>jobs</td>
<td>jets</td>
<td>LINK</td>
</tr>
<tr>
<td>wind</td>
<td>cont</td>
<td>job</td>
<td>good</td>
<td>occupywallstreet</td>
</tr>
<tr>
<td>temperature</td>
<td>space</td>
<td>tweetmyjobs</td>
<td>win</td>
<td>2</td>
</tr>
<tr>
<td>mph</td>
<td>nasa</td>
<td>ny</td>
<td>team</td>
<td>people</td>
</tr>
</tbody>
</table>

Table 1: top words within topics.

From Table 1, I also find some problems. The URL links and key word ‘rt’ may be meaningless in the topics, because they seem to appear everywhere. A potential solution is to remove them in pre-processing step, but these may also give some information.

5 Prediction

The next step is to rank new unseen tweets according to the observed tweets of some user. First, the new unseen tweets are enriched using Wikipedia articles. Second, a score is computed for each expanded tweet.

The length of a tweet is often short. After pre-processing, there may be even zero word left, which leads to poor result. In order to solve the problem of short text, articles on Wikipedia are usually used to enrich the original text.[9] In this project, Wikipedia database dump on November 16, 2011 is downloaded and indexed using Lucene. For each new tweet, I first query Lucene using the original tweet. Then the top five articles in relevance are selected to replace the original tweet. The enriched text is converted to the bag of words format as pre-processing step.

A score is computed for each new tweet to predict how much will the user like a new tweet given the observed tweets that users posted and retweeted before. A higher score indicates a larger potential that the user will like the tweet. The score of the Bayesian Sets [8] is adopted to avoid other influences on the likelihood rather than the observed tweets. It is defined as:
score($D_{new}$) = \frac{p(D_{new}|D_{observed})}{p(D_{new})}

$p(D_{new})$ is the likelihood calculated using the variational inference of LDA. By assuming all words in the new tweet and the observed tweets are independent given some $\theta$, $p(D_{new}|D_{observed})$ can be written as:

\[ p(D_{new}|D_{observed}) = \int \prod_{w_{new} \in D_{new}} \sum_{z} p(w_{new}|z)p(z|\theta)p(\theta|D_{observed}) \, d\theta. \]

However, it is difficult to compute $p(D_{new}|D_{observed})$ directly.

I try to avoid the calculation by assuming that each word $w_{new}$ in the new tweets are independent given the observed value. Therefore I can approximate $p(w_{new}|D_{observed})$ for each word and multiple them together:

\[ p(D_{new}|D_{observed}) = \prod_{w_{new} \in D_{new}} \int \sum_{z} p(w_{new}|z)p(z|\theta)p(\theta|D_{observed}) \, d\theta \]

\[ = \prod_{w_{new} \in D_{new}} \sum_{z} p(w_{new}|z) \int p(z|\theta)p(\theta|D_{observed}) \, d\theta \]

However testing result shows that the score is usually larger when a tweet is shorter. It has no clear correlation with whether the new tweet is held-out from tweets of that user. The possible reason is that my assumption on independence is not correct.

Another possible way to rank tweets is ranking by similarity. First we apply LDA for feature selection. We only use the top words in all topics as the features. And we convert the new tweets and the observed tweets into vectors. Finally, we can compute a value for every new tweet and the observed tweets of some user, using some similarity measure for vectors. But I have not finished this implementation yet.
6 Conclusions

In this project, I try to model Twitter using the LDA. Working on the data I collected, I train a LDA model. A suitable number of topics for the model is chosen. And the model appears to capture several hidden topics. To solve the short text problem, I enrich tweets in query using articles on Wikipedia. The tweets are ranked by a score calculated based on the posterior probability and likelihood. But the result is not good, the score is highly related to the length of the tweets instead the relevance based this approach. For another approach based on the similarity measures, I have not finished the implementation.

References


