Finding Similar Tweets and Similar Users by Applying Document Similarity to Twitter Streaming Data

by

Iwao FUJINO*1 and Yuko HOSHINO*2

( received on September 28, 2013 & accepted on January 27, 2014 )

Abstract

Twitter has grown so rapidly that its users are suffering from an information overload. In order to help people to find interesting tweets and users from the enormous Twitter space, by applying the document similarity of Gerard Salton and Christopher Buckley to Twitter streaming data, we attempt to devise a content-based scheme that compares similarities between Twitter users by matching their tweets against each other, especially in terms of the new key-value type database environment. Considering that each tweet is a very short message, we performed data processing of individual tweet level and accumulated tweet level according to author name in our study. At the individual tweet level, finding similar tweets is functional for finding retweets of tweeted messages, which is helpful for estimating the information propagation in Twitter. Also in the meaning of security, it is also effective for finding spam tweets and dishonest copies of tweeted messages. At the accumulated tweet level, finding users who use similar words or expressions is functional for finding users who post similar contents or topics, which is helpful for finding friends who have similar preferences and interests. As for the concrete procedures of both levels, we first use Japanese morphological analysis to pick up terms from Twitter data. Then we calculate tfidf to provide a weight parameter for each term. Finally we calculate the document similarity from weight parameter vector between any two documents, which shows how much the tweets or the authors are similar to each other. As a confirmation work, we build a computer system to search tweets by keywords query and to show user similarity between any two users. Distribution graphs of similarity in both tweet level and author level are also achieved.

Keywords: Twitter, Morphological Analysis, Key-value Database, Tfidf, Document Similarity

1. Introduction

Twitter is an online social networking service and microblogging service started from July 2006. Unlike many other social network services, Twitter not only maintains social links among users but also broadcasts information to crowds like the mass media [1]. Furthermore because these two roles work interactively, users may have massive chances to access information which they have interest in as well as interact with the users who posted the information on Twitter. According to Wikipedia [2], Twitter has over 500 million registered users as of 2012 and generating over 340 million tweets every day. Also according to the report of Semio cast [3], Japan remains the second most active country in terms of posted tweets: in June 2012, 10.6% of all public tweets were posted from Japan, while Japanese users represent 6.7% of all Twitter users. However nowadays Twitter users are suffering from an information overload as it has grown so rapidly. In order to overcome this problem, a computer system is introduced to help Twitter users to find what they want. By means of this computer system, Twitter users can search the information relevant to given keywords or check if any tweets similar to the tweets of themselves or find the users who generally post tweets with similar words or expressions similar to themselves. When dealing with an individual tweet, finding similar tweets is functional for finding retweets of tweeted messages, which is helpful for estimating the information propagation in Twitter. Also in the meaning of security, it is also effective for finding spam tweets and dishonest copies of tweeted messages. When dealing with tweets accumulated according to author, finding users who use similar words or expressions is functional for finding users who post similar contents or topics, which is helpful for finding friends who have similar preferences and interests. From these standpoints the document similarity of tweets and Twitter users should be placed as a fundamental theoretical problem, and implementing the document similarity on a computer system efficiently is an important problem as well.

Methodology for Twitter data analysis is classified into two types: the relation-based type and the
content-based type. In the field of content-based type, many previous works focus on ranking and clustering Twitter users according to the similarity of Twitter contents. As leading approaches in this filed, two interesting examples are introduced in the following. S. Petrovic, M. Osborne and V. Lavrenko reported their work on “Streaming First Story Detection with application to Twitter” [4]. Their approach is based on locality sensitive hashing method adapted to the first story detection task by introducing a backoff towards exact search. The adaptation eliminates variance in detection results and significantly improves the performance of the system. Meanwhile C. G. Akcora, M. A. Bayir, M. Demirbas and H. Ferhatosmanoglu provided their approach on “Identifying Breakpoints in Public Opinion” [5]. In this paper, based on two observations on Twitter and the streaming tfidf algorithm, they proposed efficient methods to identify break points and classify public opinions in a large stream of information. They also reported a method to detect the changes over time and find related events that caused the opinion changes from Twitter streaming timeline.

In our study as a basic consideration, we think about the concept of document similarity proposed by Gerard Salton and Christopher Buckley [6] for information retrieval may be applied to Twitter streaming data. We attempt to devise a content-based scheme that compares similarities between Twitter users by matching their tweets against each other, especially in terms of the new key-value type database environment. The key-value type database stores data in a collection of (key, value) pairs, such that each possible key appears at most once in the collection. As Twitter data is dispatched in the style of attribute and its value pairs, the key-value type database becomes the most appropriate database for processing Twitter data. Considering that each tweet is a very short message, we performed two levels of data processing in our study. At the first level, called a tweet level, we intend to deal with individual tweet data just as it is. At the second level, called an author level, we intend to process accumulated tweets data which is divided according to authors. For both of these levels, we first use Japanese morphological analysis to pick up terms from Twitter data. Then we calculate tfidf to give a weight parameter for each term. Finally we calculate the document similarity from weight parameter vector between any two documents. As a confirmation work, we will build a simple search engine to discover similar tweets for a given query or to calculate document similarity between any two users of Twitter. Calculation results of document similarity distribution at both tweet level and author level are plotted in graphs also.

The remainder of this paper consists of as follows. In section 2 we will describe basic processing procedure and calculation of weight parameter tfidf and document similarity. In section 3 we will give the details for implementing the procedure and the calculation of various parameters. In section 4 we will deal with individual tweet to build a similar tweets search system and also to find similar tweets and to figure the distribution of similarity according to tweet id. In section 5 we will deal with accumulated tweet data according to author to find similar authors and also to figure the distribution of similarity according to author name. Finally in section 6, we will conclude our works and describe some problems for our future research.

2. Basic Procedure of Proposed Scheme and Calculation of Document Similarity

2.1 Basic procedure of proposed scheme

Regarding the documents in Japanese, the document similarity is performed by the following procedures.

1. Extract terms from Japanese documents by using morphological analysis.
2. Calculate the tf (term frequency) parameter for each term and each document.
3. Calculate the idf (inverse document frequency) parameter for each term.
4. Multiply the tf parameter and the idf parameter to get the tfidf (term frequency and inverse document frequency) weight parameter for each term and each document.
5. Calculate document similarity between any two documents using the tfidf weight parameter vector.

2.2 Weight parameter of terms

In order to numeralize each extracted term, we utilize tfidf as the weight parameter in our study. There are several kinds of calculation expression for tfidf [6][7][8]. The expression we use here will be described in the following. At first we will give the definition expression of term frequency tf(k,j) [6][7][8]. tf(k,j) is the appearance
frequency of term $t_k$. If we denote the appearance number of term $t_k$ in document $D_j$ as $n_{kj}$, we can give term frequency $tf(k,j)$ as follows,

$$ tf(k,j) = \frac{n_{kj}}{\sum n_{kj}} \tag{1} $$

where the term $t_k$ will become more important in document $D_j$ if the value of $tf(k,j)$ becomes larger. Next we will give the definition equation of document frequency $df(k)$ [6][7][8]. $df(k)$ is the appearance frequency of term $t_k$. When term $t_k$ appeared in document $D_j$ at least one time, we count the appearance number of document regarding term $t_k$ as one time. The total number of document where term $t_k$ appeared is called the appearance number of document regarding the whole document in data set and we will denote it as $|d_{kj}|$. Also if we denote the total number of whole document data set as $|D|$, then we can give document frequency $df(k)$ as follows,

$$ df(k) = \frac{|d_{kj}|}{|D|} \tag{2} $$

whereas the term $t_k$ will become less important if the value of $df(k)$ becomes larger. To make it be proportional to importance of the terms, we choose the logarithmic value of inversed document frequency idf(k) [6][7][8] and the expression is given as follows,

$$ idf(k) = \log \frac{1}{df(k)} = \log \frac{|D|}{|d_{kj}|} \tag{3} $$

where the logarithm operation is used as an attenuation factor here. As a result the term $t_k$ will become more important if the value of idf(k) becomes larger. At last the weight parameter tfidf(k,j) of term $t_k$ in document $D_j$ is defined by the product of term frequency $tf(k,j)$ and document frequency $df(k)$, which is given as follows,

$$ w_j^k = tfidf(k,j) = tf(k,j) \cdot idf(k) \tag{4} $$

we will use this parameter as an importance metric for weighting terms in our study.

2.3 Document similarity

By using these methods for extracting and weighting terms in documents, we can express a document as a numerical vector given as follows,

$$ \vec{D}_j = (w_j^1, w_j^2, \ldots, w_j^m) \tag{5} $$

where $m$ indicates the dimension of vector space, which is the total number of terms appeared in the whole data set. Therefore we can calculate the similarity of two document vector $(\vec{D}_i, \vec{D}_j)$ by cosine similarity, whose definition formula is given as follows [6][7][8].

$$ \text{sim}(\vec{D}_i, \vec{D}_j) = \frac{\vec{D}_i \cdot \vec{D}_j}{|\vec{D}_i||\vec{D}_j|} = \frac{\sum_{k=1}^{m} w_i^k w_j^k}{\sqrt{\sum_{k=1}^{m} (w_i^k)^2 \cdot \sum_{k=1}^{m} (w_j^k)^2}} \tag{6} $$

3. Implementing the Calculation of Document Similarity to Twitter Streaming Data

3.1 Details of gathering and accumulating Twitter streaming data

In our study, we gathered all the tweets provided by the sample of Twitter streaming API and accumulated these data to CouchDB. The details for this step are shown as follows.

1. In order to gather data from Japanese tweet as accurately as possible, we only pick out the tweets from Twitter streaming data on the conditions as follows.
   (a) The language of user account is assigned as Japanese. That is the attribute of author.lang="ja".
   (b) At least one Japanese character is included in the attribute of text.
2. All gathered data is accumulated to CouchDB. To help find the data easily, we store all Twitter data according to the gathered date with a database name like “tweets-streaming-datasetnameNO”, where NO indicates the number of database.

3.2 Details of extracting terms from Japanese tweet text

Because there is no separation symbol obviously between words in Japanese, we have to use morphological analysis to separate passages to words. In our study we used a Japanese morphological analyzer named MeCab [9] to separate words from passages and pick out all nouns, verbs and adjectives as terms. Because the original MeCeb works only on the condition of standard Japanese and there appear many non-standard transcribe in Twitter, it may not work properly sometimes for tweet texts. To improve this
weak point, we gather non-standard words, notations and expressions from Twitter and register them to users’ dictionary of MeCab so that they can be recognized. Although many efforts have been done, there still remains some improper terms in the results, so at last we remove these terms by using a stop word list. The detail flow chart for this procedure is shown in Figure 1.

3.3 Details of calculating tfidf weight parameter

This procedure consists of two steps, the df calculation step and the tfidf calculation step. In both of these two steps, the algorithm description is given on the assumption that the input data term_by_author is given by a data structure of {author, terms} pair, where “author” represents author name, and the “terms” represents an indefinite length list of terms. The detail flow chart for calculating df parameter is shown in Figure 2 and the detail flow chart for calculating tfidf weight parameter is shown in Figure 3.

3.4 Details of calculating document similarity

This procedure consists of two steps, the norm calculation step and the similarity calculation step. In both of these two steps, the algorithm description is given on the
4. Applying Document Similarity to Individual Tweet Text

In this section, we will show three confirmation experiments using the proposed procedure regarding the tweet level. The first is a simple search engine application utilizing the document similarity between given query and any tweet from Twitter streaming data. The second is going to find all similar tweets in the Twitter streaming timeline. The third is going to investigate the distribution of similarity between any two tweets. The summary of data set and the results of these experiments will be given in the following subsections.

4.1 Summary of data set “sample”

The summary of data set “sample” is shown as follows.

<table>
<thead>
<tr>
<th>Database name</th>
<th>sample1</th>
<th>sample2</th>
<th>sample3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crawling date</td>
<td>13/08/09</td>
<td>13/08/11</td>
<td>13/08/16</td>
</tr>
<tr>
<td>Size in CouchDB</td>
<td>49.8MB</td>
<td>105.9MB</td>
<td>224.9MB</td>
</tr>
<tr>
<td>Number of documents</td>
<td>10,080</td>
<td>20,054</td>
<td>40,065</td>
</tr>
<tr>
<td>Number of authors</td>
<td>9,888</td>
<td>10,938</td>
<td>37,923</td>
</tr>
<tr>
<td>Number of terms</td>
<td>14,410</td>
<td>22,540</td>
<td>34,123</td>
</tr>
</tbody>
</table>
4.2 Search tweets by weighted keywords

The first experiment in tweet level aims to build a simple search engine application for Twitter. For a query given in the form of query={keyword1: weight1, keyword2: weight2, ..., keywordn: weightn}, this application calculates the document similarity between query and each tweet in data set and outputs the results in descending order of the similarity. A screen shot of the search results for query={大学:1.0, 学祭:0.5, 文化祭:0.4} is shown Figure 6.

4.3 Finding similar tweets in Twitter streaming timeline

The second experiment aims to find the document similarity between any two tweets. In the above sample data set there are 70,199 tweets all together. But for the reason of precision and elapse time of computation, we only adopted the tweet which has more than 15 terms as an effective tweet and all the effective tweets are 5810 at this time. The results are saved into CouchDB, which can be fetched from other applications. To give an example of our calculation results, a screen shot of calculation results in CouchDB is shown in Figure 7. In this figure, the value 1 means that it is judged that the two tweets are completely similar to each other. In other words it is judged that all the six tweets in the field of similar_tweet are completely similar to the tweet in the field of tweet_id.
4.4 Distribution of document similarity between any two tweets

The third experiment aims to calculate the document similarity between any two tweets and to figure the results in a distribution graph. The result of this experiment is shown in Figure 8. The x axis indicates the document similarity between any two tweets in percentage and the unit level of similarity is 1%. The y axis, which is graduated in a logarithm scale, indicates the appearance frequency in percentage, which is normalized by the square of the number of tweets. This figure shows there are about 98% appearances for the similarity less than 1% and also shows a steep descent curve as the similarity level increases until about 96%. But for the similarity level over 96%, it shows a very sharp rise, we think that this phenomenon is caused by retweet, duplication, quotation of tweeted messages.

![Fig. 8 Distribution of document similarity between any two tweets](image)

5. Applying Document Similarity to Accumulated Tweets

In this section, we will show two confirmation experiments using the proposed procedure regarding author level. The first is going to find all similar authors from Twitter streaming timeline. The second is going to investigate the distribution of similarity between any two authors. The summary of the data set and the results of these experiments will be given in the following subsections.

5.1 Summary of data set “bigtest”

The summary of data set “bigtest” is shown as follows.

Table 2 Summary of data set “bigtest”

<table>
<thead>
<tr>
<th>Database name</th>
<th>bigtest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crawling date</td>
<td>13/03/13</td>
</tr>
<tr>
<td>Size in CouchDB</td>
<td>7.8GB</td>
</tr>
<tr>
<td>Number of documents</td>
<td>1,167,309</td>
</tr>
<tr>
<td>Number of authors</td>
<td>739,321</td>
</tr>
<tr>
<td>Number of terms</td>
<td>232,135</td>
</tr>
</tbody>
</table>

5.2 Finding similar authors in Twitter streaming timeline

The first experiment in author level aims to find the document similarity between any two Twitter users. In the above data set “bigtest” there are 1,167,309 tweets all together and 739,321 distinct authors. For the reason of precision and elapse time of computation, we only adopted effective author on the following conditions: at least 5 tweets are accumulated for an author document and the document has more than 30 terms. As a result there are 9584 effective authors all together. To give an example of our calculation result, a screen shot of the similarity results in CouchDB is shown in Figure 9. This figure shows that all the author names in the field of similar_author have similar contents to the author name in the field of author with a similarity value shown next to the author’s name, whereas we only saved the users who have similarity larger than 0.5 to CouchDB this time. For all the calculation results there are 1492 documents which have details data stored in the format of Figure 9.

5.3 Distribution of document similarity between any two authors

The second experiment aims to calculate the document similarity between any two users and to figure the results in a distribution graph. The result of this experiment is shown in Figure 10. The x axis indicates the document similarity between any two authors in percentage and the unit level of similarity is 1%. The y axis, which is graduated in a logarithm scale, indicates the appearance frequency in percentage, which is normalized by the square of the number of authors. It is observed that the curve in this
graph is very similar to that of Figure 8, but it shows a relative gentle descent compared to Figure 8.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>2bc5f4b90c7ec65b59ed6f64d068f735</td>
</tr>
<tr>
<td>_rev</td>
<td>32f5d2f60f244e12f2e83b013f52f53</td>
</tr>
<tr>
<td>author</td>
<td>8fad725b7b846a689</td>
</tr>
<tr>
<td>similar_author</td>
<td>7890ad3c54d964a87</td>
</tr>
</tbody>
</table>

Fig. 9 Calculation results of similarity between any two authors (details of document in CouchDB)

Fig. 10 Distribution of document similarity between any two authors

6. Conclusions

In this paper, according to the document similarity method for information retrieval, we proposed a content-based implementing scheme for finding similar tweets and similar users in Twitter in terms of a key-value type database environment. Based on the proposed procedures and calculation details, we produced a computer system to perform confirmation work. In consequence, the system works according to its specification and the validity of our proposed implementing scheme is confirmed at the prototype level. Although the confirmation work is achieved successfully, there are still many practical issues we have to deal with. As we use all nouns, verbs and adjectives as terms, there are too many terms we have to process in the subsequent procedures. However, many of extracted terms may not describe the content of the tweet plausibly, so that processing these redundant terms may exhaust futile computation time. Accordingly as the future work we will attempt to develop more efficient algorithm for extracting terms. Regarding the computation problem, although it is an approach to introduce more powerful computer systems or high speed database systems certainly, adopting parallel computation scheme may provide more effective solutions because our work needs to process massive irrelevant document records. As another future work we will endeavor to develop parallel algorithm for processing Twitter streaming data.

We expect that our works will be utilized to prevent spamming, to analyze the information propagation and to find friends in Twitter. Furthermore, we expect that the influences of our works may not be limited to Twitter space, but it will also contribute to the development of social relations in the substantial world.

References


