東海大学紀要情報通信学部
vol.6,No2,2013,pp.22-30

Paper

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 twe et 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 c ontents 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 Semiocast [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

*1 Professor, Department of Telecommunication and Network Engineering, School of Information and Telecommunication Engineering, Tokai University

*2 Lecturer, Department of Information and Media Technology, School of Information and Telecommunication Engineering, Tokai University

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 $\mathrm{tf}(k,j)$ as follows,

$$\mathrm{tf}(k,j) = \frac{n_{k,j}}{\sum_l n_{l,j}} \qquad (1)$$

where the term t_k will become more important in document D_j if the value of $\mathrm{tf}(k,j)$ becomes larger. Next we will give the definition equation of document frequency $\mathrm{df}(k)$ [6][7][8]. $\mathrm{df}(k)$ 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 $|d_k|$. Also if we denote the total number of whole document data set as $|D|$, then we can give document frequency $\mathrm{df}(k)$ as follows,

$$\mathrm{df}(k) = \frac{|d_k|}{|D|} \qquad (2)$$

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

$$\mathrm{idf}(k) = \log\frac{1}{\mathrm{df}(k)} = \log\frac{|D|}{|d_k|} \qquad (3)$$

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

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

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,

$$\overrightarrow{D_j} = (w_j^1, w_j^2, \dots w_j^m) \qquad (5)$$

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 $(\overrightarrow{D_i}, \overrightarrow{D_j})$ by cosine similarity, whose definition formula is given as follows [6][7][8].

$$\mathrm{sim}(D_i, D_j) = \frac{\overrightarrow{D_i}\cdot\overrightarrow{D_j}}{|\overrightarrow{D_i}||\overrightarrow{D_j}|} = \frac{\sum_{k=1}^{m}(w_i^k \cdot w_j^k)}{\sqrt{\sum_{k=1}^{m}(w_i^k)^2}\cdot\sqrt{\sum_{k=1}^{m}(w_j^k)^2}} \qquad (6)$$

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 Mecab 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.

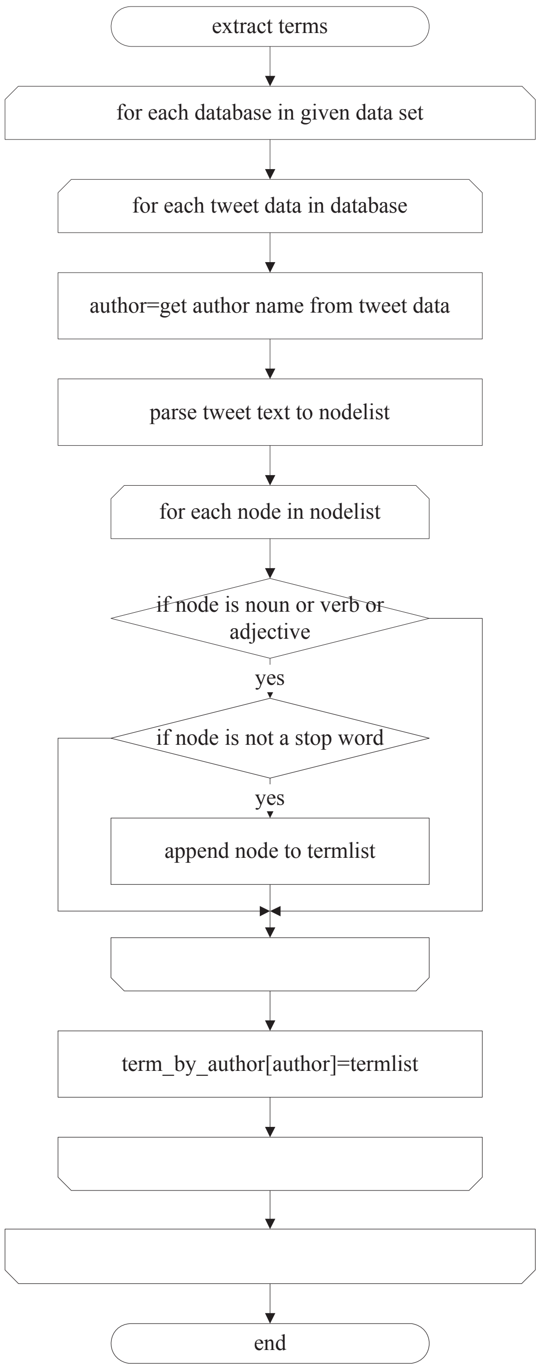

extract terms
for each database in given data set
for each tweet data in database
author=get author name from tweet data
parse tweet text to nodelist
for each node in nodelist
if node is noun or verb or adjective
yes
if node is not a stop word
yes
append node to termlist
term_by_author[author]=termlist
end

Fig. 1 Flow chart for extracting terms from Twitter streaming data set

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

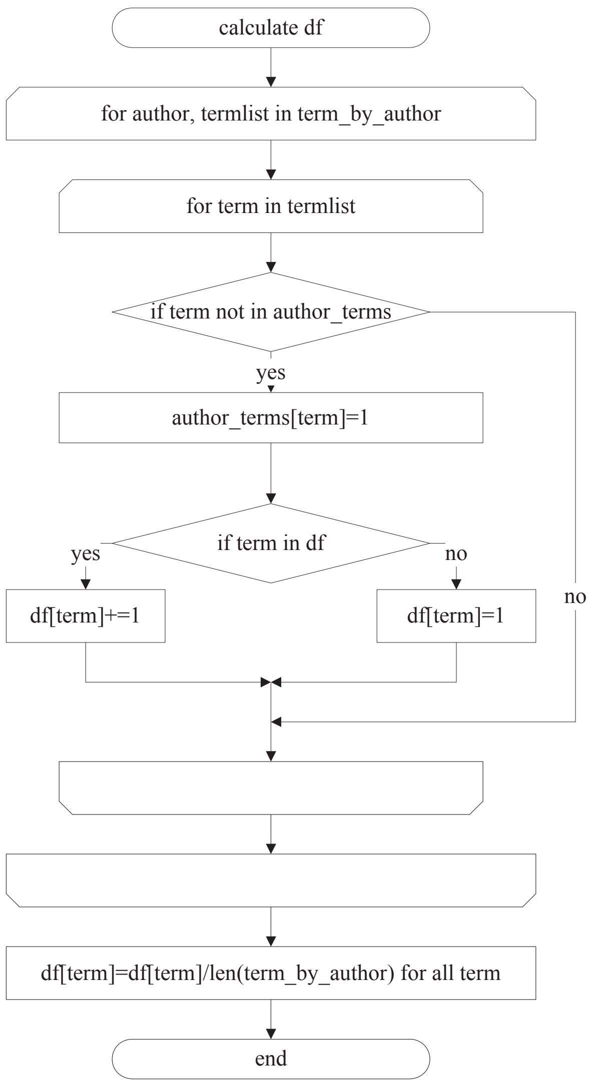

calculate df
for author, termlist in term_by_author
for term in termlist
if term not in author_terms
yes
author_terms[term]=1
if term in df
yes
no
df[term]+=1
df[term]=1
no
df[term]=df[term]/len(term_by_author) for all term
end

Fig. 2 Flow chart for calculating df parameter from extracted terms

assumption that the input data termtfidf_by_author is given by a data structure of {author, termtfidf} pair, where “author” represents author name, and the “termtfidf” represents an indefinite length {term, tfidf} pair of terms. The detail flow chart for calculating norm is shown in Figure 4 and the detail flow chart for calculating similarity is shown in Figure 5.

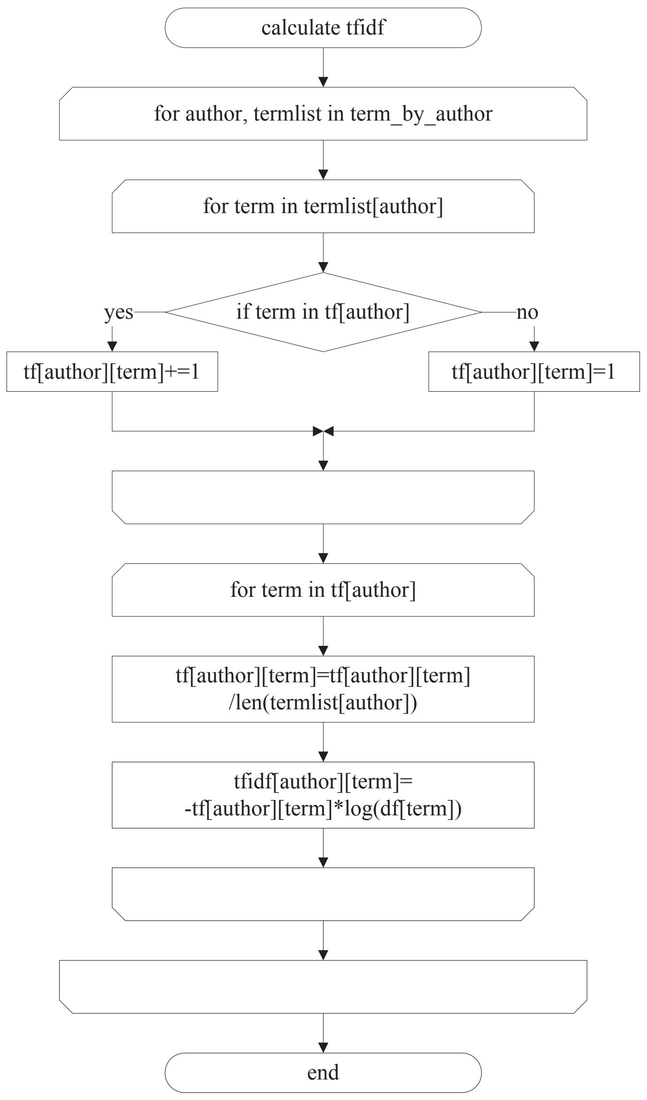


Fig. 3 Flow chart for calculating tfidf parameter from extracted terms

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.

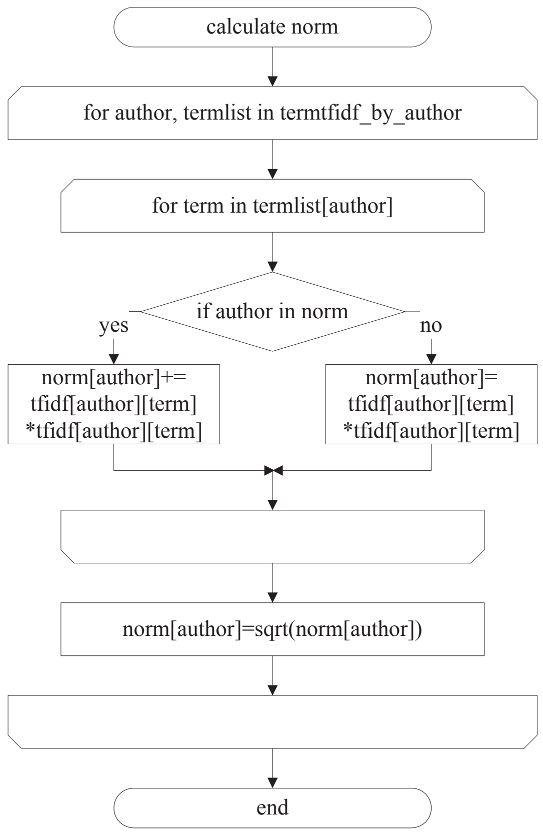


Fig. 4 Flow chart for calculating norm of author from term weight parameter saved according to author name

4.1 Summary of data set “sample”

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

Table 1 Summary of data set “sample”

Database name	sample1	sample2	sample3
Crawling date	13/08/09	13/08/11	13/08/16
Size in CouchDB	49.8MB	105.9MB	224.9MB
Number of documents	10,080	20,054	40,065
Number of authors	9,888	10,938	37,923
Number of terms	14,410	22,540	34,123

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.

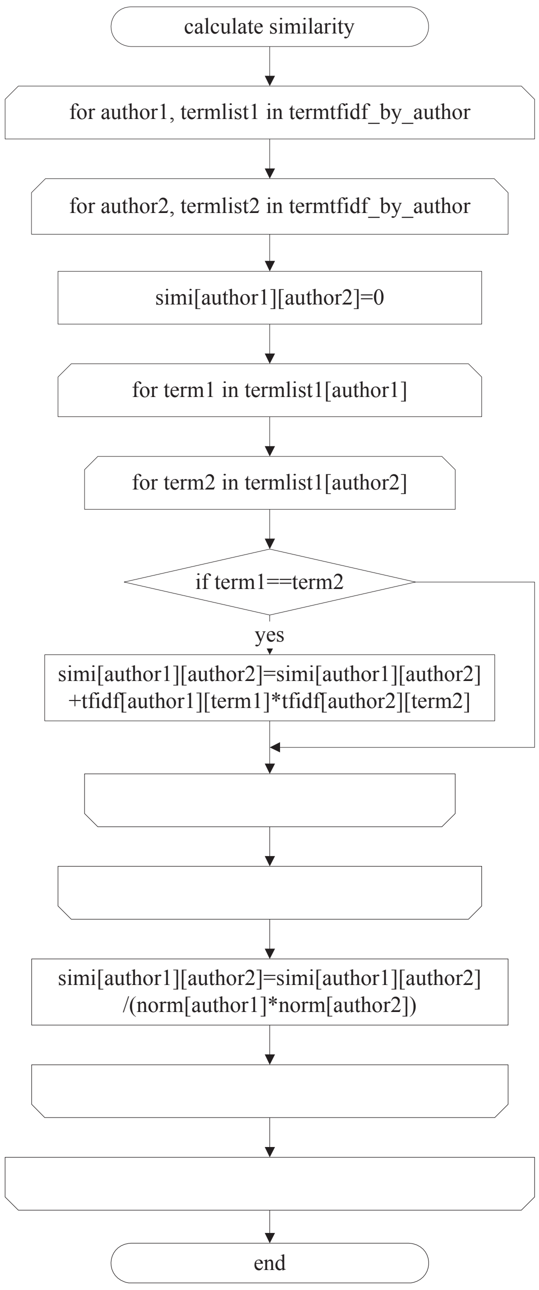
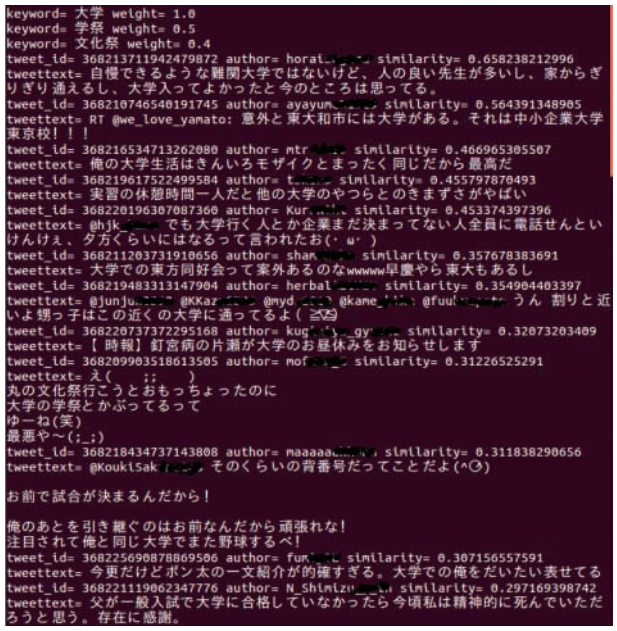

calculate similarity
for author1, termlist1 in termtfidf_by_author
for author2, termlist2 in termtfidf_by_author
simi[author1][author2]=0
for term1 in termlist1[author1]
for term2 in termlist1[author2]
if term1==term2
yes
simi[author1][author2]=simi[author1][author2] +tfidf[author1][term1]*tfidf[author2][term2]
simi[author1][author2]=simi[author1][author2] /(norm[author1]*norm[author2])
end

Fig. 5 Flow chart for calculating document similarity from term weight parameter saved according to author name

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.

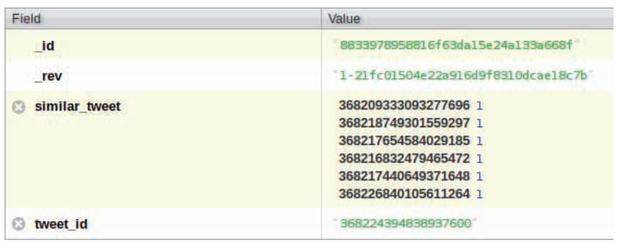


```
keyword= 大学 weight= 1.0
keyword= 学祭 weight= 0.5
keyword= 文化祭 weight= 0.4
tweet_id= 368213711942479872 author= horai[redacted] similarity= 0.658238212996
tweettext= 自慢できるような難関大学ではないけど、人の良い先生が多いし、家からぎ
りぎり通えるし、大学入ってよかったと今のところは思ってる。
tweet_id= 368210746540191745 author= ayayum[redacted] similarity= 0.564391348905
tweettext= RT @we_love_yamato: 意外と東大和市には大学がある。それは中小企業大学
東京校！！！
tweet_id= 368216534713262080 author= mtr[redacted] similarity= 0.466965305507
tweettext= 俺の大学生活はきんいろモザイクとまったく同じだから最高だ
tweet_id= 368219617522499584 author= t[redacted] similarity= 0.455797870493
tweettext= 実習の休憩時間一人だと他の大学のやつらとのきまずさがやばい
tweet_id= 368220196307087360 author= Kur[redacted] similarity= 0.453374397396
tweettext= @hjk_[redacted] でも大学行く人とか企業まだ決まってない人全員に電話せんとい
けんけぇ、夕方くらいにはなるって言われたお(･ω･)
tweet_id= 368211203731910656 author= shan[redacted] similarity= 0.357678383691
tweettext= 大学での東方同好会って案外あるのなwwwww早慶やら東大もあるし
tweet_id= 368219483313147904 author= herbal[redacted] similarity= 0.354904403397
tweettext= @junju[redacted] @KKaz[redacted] @myd[redacted] @kame_[redacted] @fuu[redacted] うん 割りと近
いよ甥っ子はこの近くの大学に通ってるよ(≧▽≦)
tweet_id= 368220737372295168 author= kug[redacted]_gy[redacted] similarity= 0.32073203409
tweettext=【 時報】釘宮病の片瀬が大学のお昼休みをお知らせします
tweet_id= 368209903518613505 author= mof[redacted] similarity= 0.31226525291
tweettext= え(    ;;    )
丸の文化祭行こうとおもっちょったのに
大学の学祭とかぶってるって
ゆーね(笑)
最悪や～(;_;)
tweet_id= 368218434737143808 author= maaaaa[redacted] similarity= 0.311838290656
tweettext= @KoukiSak[redacted] そのくらいの背番号だってことだよ(^O^)

お前で試合が決まるんだから！

俺のあとを引き継ぐのはお前なんだから頑張れな！
注目されて俺と同じ大学でまた野球するべ！
tweet_id= 368225690878869506 author= fum[redacted] similarity= 0.307156557591
tweettext= 今更だけどポン太の一文紹介が的確すぎる。大学での俺をだいたい表せてる
tweet_id= 368221119062347776 author= N_Shimizu[redacted] similarity= 0.297169398742
tweettext= 父が一般入試で大学に合格していなかったら今頃私は精神的に死んでいただ
ろうと思う。存在に感謝。
```

Fig. 6 Screen shot of search results for query={大学:1.0, 学祭:0.5, 文化祭:0.4}



Field	Value
_id	"8833978958816f63da15e24a133a668f"
_rev	"1-21fc01504e22a916d9f8310dcae18c7b"

similar_tweet	368209333093277696 1
	368218749301559297 1
	368217654584029185 1
	368216832479465472 1
	368217440649371648 1
	368226840105611264 1

tweet_id	"368224394838937600"

Fig.7 Calculation results of similarity between any two tweets (details of document in CouchDB)

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.

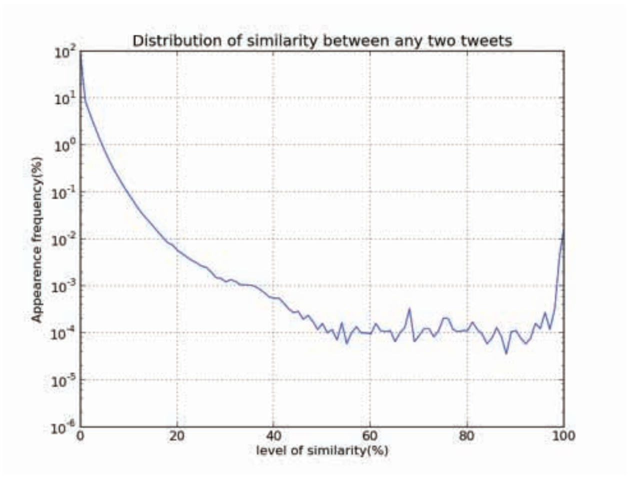


Fig. 8 Distribution of document similarity between any two tweets

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”

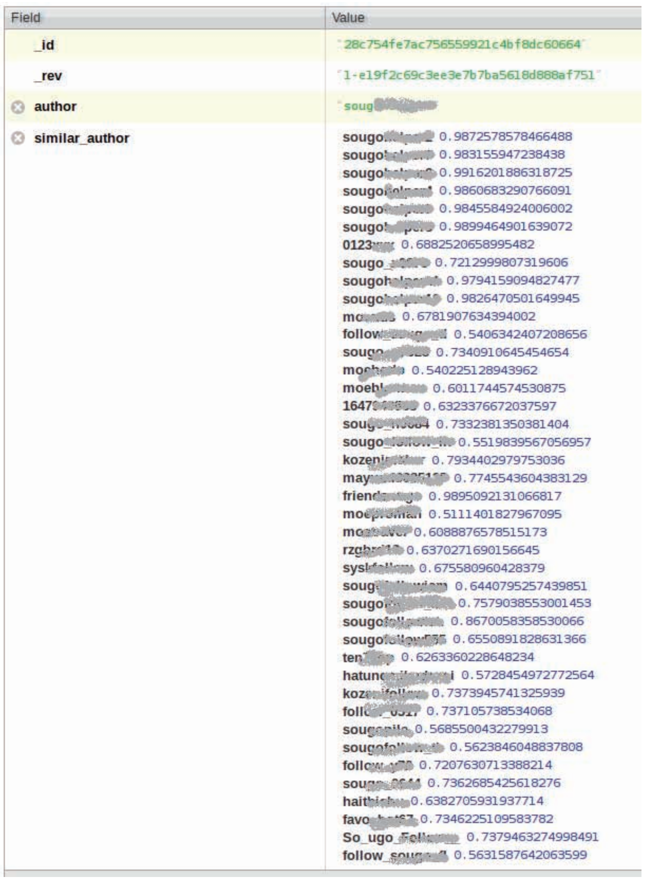
Database name	bigtest
Crawling date	13/03/13
Size in CouchDB	7.8GB
Number of documents	1,167,309
Number of authors	739,321
Number of terms	232,135

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.



Field	Value
_id	"28c754fe7ac756559921c4bf8dc60664"
_rev	"1-e19f2c69c3ee3e7b7ba5618d888af751"
author	"soug[redacted]
similar_author	sougo[redacted] 0.9872578578466488
	sougo[redacted] 0.983155947238438
	sougo[redacted] 0.9916201886318725
	sougo[redacted] 0.9860683290766091
	sougo[redacted] 0.9845584924006002
	sougo[redacted] 0.9899464901639072
	0123[redacted] 0.6882520658995482
	sougo_[redacted] 0.7212999807319606
	sougoh[redacted] 0.9794159094827477
	sougo[redacted] 0.9826470501649945
	mo[redacted] 0.6781907634394002
	follow[redacted] 0.5406342407208656
	soug[redacted] 0.7340910645454654
	mo[redacted] 0.540225128943962
	moeb[redacted] 0.6011744574530875
	1647[redacted] 0.6323376672037597
	soug[redacted] 0.7332381350381404
	sougo[redacted] 0.5519839567056957
	kozeni[redacted] 0.7934402979753036
	may[redacted] 0.7745543604383129
	frien[redacted] 0.9895092131066817
	moe[redacted] 0.5111401827967095
	mo[redacted] 0.6088876578515173
	rzg[redacted] 0.6370271690156645
	sys[redacted] 0.675580960428379
	soug[redacted] 0.6440795257439851
	sougo[redacted] 0.7579038553001453
	sougo[redacted] 0.8670058358530066
	sougo[redacted] 0.6550891828631366
	ten[redacted] 0.6263360228648234
	hatun[redacted] 0.5728454972772564
	koz[redacted] 0.7373945741325939
	foll[redacted] 0.737105738534068
	soug[redacted] 0.5685500432279913
	soug[redacted] 0.5623846048837808
	foll[redacted] 0.7207630713388214
	sou[redacted] 0.7362685425618276
	hait[redacted] 0.6382705931937714
	favo[redacted] 0.7346225109583782
	So_ugo_[redacted] 0.7379463274998491
	follow_[redacted] 0.5631587642063599

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

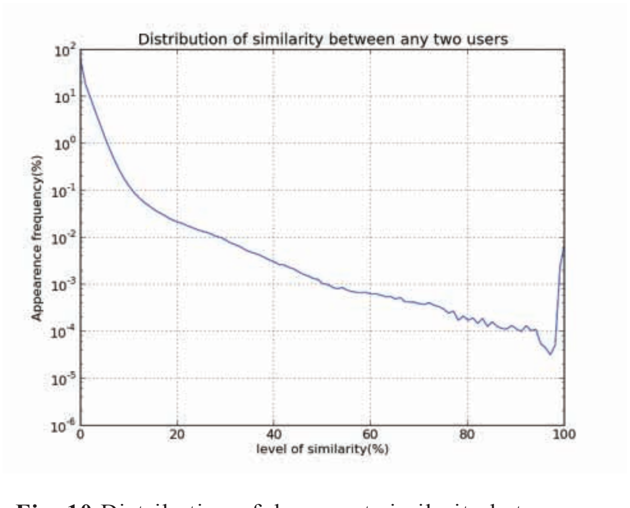


Fig. 10 Distribution of document similarity between any two authors

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

[1] H. Kwak, C. Lee, H. Park and S. Moon: "What is Twitter, a social network or a news media?" Proceedings of 19th International Conference on World Wide Web (WWW"10), pp591-600, Raleigh, North Carolina, USA (2010)

[2] "Twitter", http://en.wikipedia.org/wiki/Twitter, (Accessed on July 20, 2013)

[3] "Twitter reaches half a billion accounts More than 140 millions in the U.S.", http://semiocast.com/en/publications/2012_07_30_Twi

tter_reaches_half_a_billion_accounts_140m_in_the_US, (Accessed on July 20, 2013)

[4] S. Petrovic, M. Osborne and V. Lavrenko: "Streaming First Story Detection with application to Twitter", Human Language Technologies: The 2010 Annual Conference of the North American Chapter of ACL, pp.181-189, Los Angeles, California, USA (2010)

[5] C. G. Akcora, M. A. Bayir, M. Demirbas and H. Ferhatosmanoglu: "Identifing Breakpoints in Public Opinon", 1st workshop on Social Media Analytics (SOMA‟10), Washington, DC, USA (2010)

[6] Gerard Salton and Christopher Buckley: "Term Weighting Approaches in Automatic Text Retrieval", Information Processing and Management, 25(5), pp.513-523 (1988)

[7] W. Bruce Croft, Donald Metzler and Trevor Strohman: "Search Engines Information Retrieval in Practice", Addison Wesley, pp.237-243 (2010)

[8] Ricardo Baeza-Yates and Berthier Ribeiro-Neto: "Modern Information Retrieval the concepts and technology behind search Second edition", Addison Wesley, pp.66-79 (2011)

[9] "MeCab: Yet Another Part-of-Speech and Morphological Analyzer", http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html (Accessed on July 20, 2013)