Duplicate Detection for Identifying Social Spam in Microblogs

Qunyan Zhang, Haixin Ma, Weining Qian, Aoying Zhou
Center for Cloud Computing and Big Data, Software Engineering Institute,
East China Normal University, Shanghai, China
Email: {51121500043,51111500010}@ecnu.cn, {wnqian,ayzhou}@sei.ecnu.edu.cn

***Abstract*—As an important kind of social media, microblog has become an important source of opinion mining and collective behavior study. However, social spams may affect the analytical results greatly. This paper focuses on the problem of identifying potential social spammers who copy pieces of information from others. An improved locality-sensitive hashing based method is used for detecting duplicated tweets. Intensive empirical study over a real-life microblog dataset crawled from Sina Weibo, one of the most popular microblogging services, is conducted. The characteristics of potential spammers and their behaviors are analyzed.**

***Keywords*-duplicate detection; social spam; microblog; locality-sensitive hash; MapReduce**

I. Introduction

Social media has become one of the major sources for people to consume information. It has been shown that social media can be used in sensing earthquake[32], predicting financial index[3] and political election[37], and sentiment analysis[29].

The increasing importance of social media also attracts spammers to post pieces of irrelevant, false, or even harmful information. Since such kind of spammings usually take advantage of social network to spread the spams, they are often called *social spams*[11], [28], [22]. A major difference between social spam and traditional spams, such as email spam[10] and link spam[14], is that, in social media, spammers need to attract common users to *follow* spammers, so that spams are *subscribed*. A social spammer often pretends to be a common user or even an information hub, and posts spams from time to time.

We focus on the problem of identifying potential social spammers in microblogging services in this paper. Microblog is the most important and popular kind of social media. Popular microblogging services include Twitter[1], Tumblr[2], and Sina Weibo[3]. Since the following relationship in microblogs is one-way, and usually does not need to be approved by the followee, spammers are easy to live in the microblogosphere silently. A microblog spammer may pretend to be a common user and progressively attract followers before spams are posted for a long time. Identifying such users is challenging, since they are deceptive and there are only few features from spams available.

Technologies of duplicate detection is used to find users whose tweets[4] are mainly copied. These users are classified as potential spammers. This method is orthogonal and complementary to content-based[11], profile-based[22], and annotation-based[28] spam detection method. The intuition behind our approach is that, to pretend to be a common and/or informative user, the easiest way is to copy tweets from others. On the other hand, if a user posts pieces of original information along with a set of spams, then his followers are those attracted by his own sharing. Then, whether he is a spammer or not should be determined on the content he shared, which can be achieved by using existing methods.

The main contributions of this paper are summarized as follows.

- Duplicate detection is used for identifying potential social spammers.
- An improved duplicate detection algorithm, called LSH-with-filtering, is introduced. The MapReduce implementation of the algorithm is also presented. The enhanced algorithm is shown to be accurate and efficient in detecting duplicated tweets.
- The proposed method is evaluated over a real-life dataset from Sina Weibo, one of the most popular microblogging services in China. The characteristics of those potential spammers, including their behavior, and social networks are analyzed in details. It is shown that they have different characteristics compared to those common users.

The rest part of this paper is organized as follows. In Section 2, the preliminaries of microblog services and duplicate detection methods are introduced. Our improved LSH-based duplicate detection algorithm along with its MapReduce implementation is introduced in Section 3. Section 4 is devoted to the detailed report of intensive study of the Sina Weibo dataset. After the introduction to related work in Section 5, the last section is for concluding remarks and discussion.

It should be noted that the classification of spams, es-

[1]http://twitter.com/
[2]http://tumblr.com/
[3]http://weibo.com/

[4]The terms of Twitter, the most popular microblogging service, is used in this paper.

pecially commercial spams, and advertisement is subjective. We do not distinguish them, and use the term *spams* to represent all pieces of irrelevant, false, or malicious information.

II. Preliminaries

A. Microblogs

Terms used by Twitter is used in this paper. A *tweet* is a triple $< t_i, u_j, c >$, in which c is a piece of infomration shared by user u_j at timestamp t_i. A *followship* relationship is a binary (u, v), which means user u is *following* user v. Here, u is called the *follower*, while v is called the *followee*. Thus, all tweets shared by v would appear in u's *timeline*, which is a reversely chronologically ordered list of tweets shared by any of u's followees.

The *followship network* is a directed graph $G : (V, E)$, in which V is the set of all users, and E is the set of followship relationships. A user may *retweet* another tweet, whether it is first created by himself or other users. Thus, the *retweet* is a tweet $< t_i, u, c >$ with a pointer in c to another tweet $< t_j, v, c' >$. It should be noted that the retweeted tweet itself may be a retweet too.

Both the retweet and the retweeted one can be seen by followers of the author of the retweet. Therefore, if a user wants to share a piece of information created by another user, a more straightforwared and often easier way is just retweeting the original tweet, instead of copying and pasting the content of the original tweet and sharing the piece of information as a new tweet. Though there are chances that two users create identical tweets independantly, the possibility that most tweets of a user are coincidently the same as tweets from other users is low. Thus, duplicated content is an important clue for identifying suspicious users.

B. Duplicate detection with locality-sensitive hashing

Duplicate detection is intensively studied for its wide application in information retrieval. Existing technologies can be used to find *near* duplicate pieces of documents[31]. Note that sometimes there are minor differences between two documents, due to duplicator's intention or accidental modifications. To be self-containment, the traditional duplicate detection method used in this paper is introduced in this subsection.

Each tweet is first transformed into a word collection by using k-shingling[17]. A *shingle* in the word collection is a contiguous subsequence of tokens[5] whose length is k in the tweet.

The similarity of two word collections A and B are measured by *Jaccard similarity*, which is defined as:

$$J(A, B) = \frac{\|A \cap B\|}{\|A \cup B\|}. \quad (1)$$

[5]A token is a character if the tweet is written in Chinese, or a word if the tweet is written in English.

Minhash[4], [5] is used to cluster similar tweets, i.e. tweets whose corresponding word collection have large Jaccard similarity, together. Given a hash function h on shingles, the minhash $h_{min}(S)$ returns the minimum $h(s)$ for any $s \in S$, where S is a word collection, i.e.

$$h_{min}(S) = \arg\min_{s \in S} h(s). \quad (2)$$

An important property of minhash is that,

$$Pr[h_{min}(A) = h_{min}(B)] = J(A, B). \quad (3)$$

Thus, given a set of independent hash functions $h_1, h_2, \ldots, h_n$, Algorithm 1 is used to generate minhash signature. The signature can be used to approximate the Jaccard similarity. Given two collections A and B, if $p = J(A, B)$, then the error of using signature of n minhash functions is $\sqrt{p\,(1-p)\,/n}$, which is bounded by $1/2\sqrt{n}$.

Algorithm 1 Generate minhash signature

```
INPUT: Shingle collection S, and hash functions h_i, i = 1, 2, ..., n
ONPUT: < h_min(1), h_min(2), ..., h_min(n) >
for all i do
    h_min(i) ← ∞
end for
for all s_i ∈ S do
    for all h_j do
        if h_j(s_i) < h_min(i) then
            h_min(i) ← h_j(s_i)
        end if
    end for
end for
```

To efficiently detect near duplicated tweets, given n minhash functions, *locality-sensitive hashing* (LSH)[24], [30], [33] is used. LSH divides those n minhash functions into b bands, each of which contains r minhash functions. Each band is associated with a hash function that takes vectors with r elements and returns a number denoting a bucket. The probability that two tweets are hashed into same bucket by all bands is $1 - (1 - p^r)^b$. The probability depicts an S-curve. Therefore, documents with the same LSH result are often treated as candidates of near duplicated ones.

III. LSH with filtering

A. LSH with threshold

The probability of LSH collision has a sudden surge when Jaccard value is greater than or equal to $(1/b)^{1/r}$. Though simply treating tweets with the same LSH result as duplicated one is simple and efficient, it may bring in errors. A more definite way, which is used in this paper, is filtering those tweets further based on their minhash similarity by using the threshold $(1/b)^{1/r}$. Thus, only pairs of tweets whose minhash similarities are larger than the threshold are treated as duplicated ones.

B. MapReduce implementation

MapReduce is a programming paradigm for development of data analytics tasks that need to be executed in parallel on clusters[9]. It has the advantage that developers do not need to take care of how the program is deployed and executed in nodes of the cluster, or the fault tolerance of the system. Only functions of a *mapper* and a *reducer* are need to be implemented. A mapper returns a set of $< key, value >$ pairs, and may be deployed to be executed parallelly on several nodes in the cluster by the system, i.e. Hadoop, automatically. The system then *shuffles* all $< key, value >$ pairs emitted by the mapper executions based on their key's. All pairs with the same key are transmitted to the same node, where a *reducer* runs and processes all these pairs with the same key.

Since the number of tweets that need to be processed for detecting duplications is quite large, our duplicate detection method is implemented based on MapReduce, or more specifically, Hadoop, an open source implementation of the MapReduce paradigm. The pseudocode of the implementation is shown in Algorithm 2, 3, 4, and 5.

Algorithm 2 Mapper of preprocessing

1: **input:** $< key, value >$ in which:
   key: file name; $value$: tweet file F
2: **onput:** $\{< key', value' >\}$ in which:
   key': length of a tweet;
   $value'$: $< i, u(i), t, T >$
   {tweet identifier, author, time, and tokens}
3: **for all** tweet T_i in F **do**
4: $Q_J \leftarrow eliminate\_symbols(T_i)$;
5: $T \leftarrow \{Q'_J$s k-shingles$\}$;
6: $key' \leftarrow \|T\|$;
7: $value' \leftarrow < i, u(i), t, T >$;
8: $emit(< key', value' >)$;
9: **end for**

Algorithm 3 Reducer of preprocessing

1: **input:** $< key, value >$ in which:
   key: length of a tweet; $value$: $< i, u(i), t, T >$
2: **onput:** $\{< key', value' >\}$ in which:
   key': length of a tweet; $value'$: $< i, u(i), t, T >$
3: **for all** $< key, value >$ **do**
4: $emit(< key, value >)$; {Do nothing.}
5: **end for**

Algorithm 4 Mapper of LSH indexing

1: **input:** $< key, value >$ in which:
   key: length of a tweet; $value$: $< i, u(i), t, T >$
2: **onput:** $\{< key', value' >\}$ in which:
   key': bucket number; $value'$: $< i, t, key, T >$
3: $signature \leftarrow minhash(T)$;
4: $key' \leftarrow LSH(signature)$;
5: $emit(key', < i, t, key, T >)$;

A trick, which was previoused used in [1], [2], is used in Algorithm 5 line 6 for further reducing the times of Jaccard similarity computation. Intuitively, the Jaccard similarity between two sets T_i and T_j, whose size are $size_i$ and $size_j$, while $size_i \leq size_j$, is less than $size_i/size_j$. Thus, a tweet i is only possible to be a near duplication of another tweet j, when $size_j$ is larger than $size_i \times J$ and less than $size_i/J$, where J is the threshold of Jaccard similarity for detecting duplications.

Algorithm 5 Reducer of LSH indexing

1: **input:** $< key, value >$ in which:
   key: bucket number; $value$: $< i, t_i, size_i, T_i >$
2: **onput:** $\{< key', value' >\}$ in which:
   key': bucket; $value'$: duplicated tweets
   {The first tweet in it is not a duplication.}
3: $I \leftarrow \{i\}$; $value' \leftarrow \emptyset$;
4: **for all** i remained in I in ascendant order of t_i **do**
5: **for all** j remained in I, $j \neq i$ **do**
6: **if** $(size_i \times t) \leq size_j \leq size_i/t$, $\{t = (1/b)^{1/r}\}$ **then**
7: **if** $J(T_i, T_j) \geq t$ **then**
8: $value' \leftarrow value' \bigcup \{j\}$;
9: $I \leftarrow I - \{j\}$;
10: **end if**
11: **end if**
12: **end for**
13: **end for**
14: $emit(key, value')$;

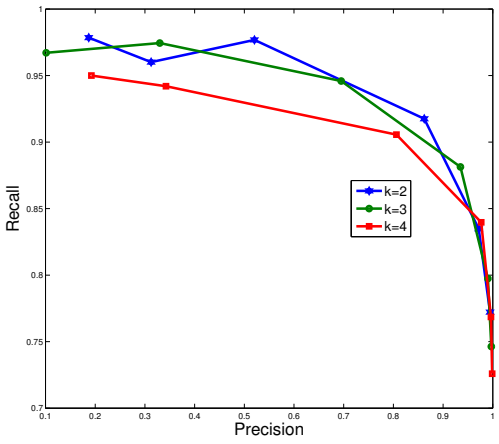


Figure 1. Precision-recall plot of the improved LSH method with different parameter settings. For each k, there are six points in the curve, each, from left to right, represents the precision and recall value with b equals to 5, 10, 20, 30, 40, and 50.

C. Performance evaluation

The improved LSH-based duplicate detection method is evaluated over a sample of Sina Weibo dataset, which includes 483,749 tweets with labels, in which 174,998 tweets are duplicated ones.

With $n = 200$ minhash functions, the performance over different parameter value combinations are tested. The result of precision-recall is shown in Figure 1. The number of charactors in a shingle, i.e. k, is set to be 2, 3, or 4, while

the number of bands, i.e. b, is set to be 5, 10, 20, 30, 40, or 50. It is shown that the method achieves high precision and recall simultaneously when $k = 3$ and $b = 20$. Thus, this parameter setting is used in the following study of social spamming.

IV. Social spams in microblogs

A. Dataset and preprocessing

A dataset crawled from Sina Weibo via Sina Weibo API is used in this study. Our distributed crawler progressively crawls tweets, followship networks, and user profiles. The crawler begins from 32 seed users, who are opinion leaders and researchers. Then, the breadth-first search leads the crawler to 1.7 million users in the three-level followship network traversal. All tweets of these users posted before the end of February 2012 are collected. There are 649 million tweets in the dataset. We retrieve all $283,204,017$ non-retweeted tweets from $1,574,332$ users in the dataset, and use them as the source of duplicate detection. These tweets are sorted based on their published time, so that later tweets with duplicated content are considered as copied. Totally 14.7 million users are reached in four-level followship network traversal. There are $1,397,243,218$ followship relationships retreived. Meanwhile, profiles of those users are also collected.

Note that the dataset is not complete. Furthermore, items are not crawled from a static snapshot. Even though, the dataset is sufficiently large to depict the duplication phenomenon. The detail of the dataset and crawler is introduced in [25].

Tweets are preprocessed before the duplicate detection is conducted. First, symbols such as @, for mensions, and ##, for hashtags, are eliminated. Shortened URLs are also removed. Furthermore, texts that generated by microblogging clients, such as "Share Image", are removed, so that those tweets will not be considered as duplicated ones. Smileyes are also deleted. In summary, symbols and content generated by the microblogging service itself, or client softwares, are removed from tweets, so that remaining texts are all composed by users.

B. Duplicated tweets and potential spammers

The threshold of Jaccard similarity that two tweets are though to be near duplicated is set to 0.8, which means if two tweets share more than 80% of their 3-shinglings, they are treated as near duplicated ones. By using this criteria, 7.06% tweets in the dataset are duplicated ones of previous tweets.

Different users have different number and ratio of duplicated tweets. The distribution of duplicated tweets over users is shown in Table I. The users fit in the Duplicated and Severely-duplicated are treated as potential spammers, and labeled as Abnormal Users in the rest of this paper, since there are a large number of tweets from them are copied from others.

Table I
Distribution of duplicated tweets of different users.

Duplication level	Ratio of duplicated tweets	Percentage of users
Normal	$[0, 0.2)$	88.2%
Slightly duplicated	$[0.2, 0.4)$	8.4%
Duplicated	$[0.4, 0.6)$	2.1%
Severely duplicated	$[0.6, 1]$	1.3%

C. Characterizing duplicate spamming

A natural question is that whether the behaviors of those potential spammers are consistent with other normal users. The distribution of tweets over days in weeks, and hours in days, are shown in Figure 2. It is shown that the distribution of time that those tweets from abnormal users are posted is different from that of tweets from others. First, on Monday, while most normal users may be busy with work and spend less time in microblogging, those abnormal users post more tweets than other days in the week. Secondly, the percentage of tweets from abnormal users during 2:00am and 8:00am is much higher than those from normal users. It is because that some abnormal users are actually running by robots, and they do not rest at night.

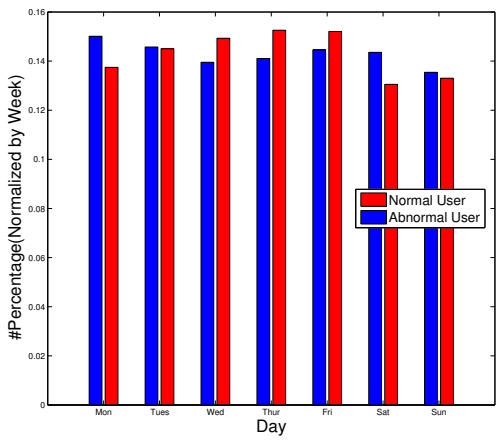
D. Characterizing duplicate spammers

The users are classified into five categories by hand. The categories are defined as follows:

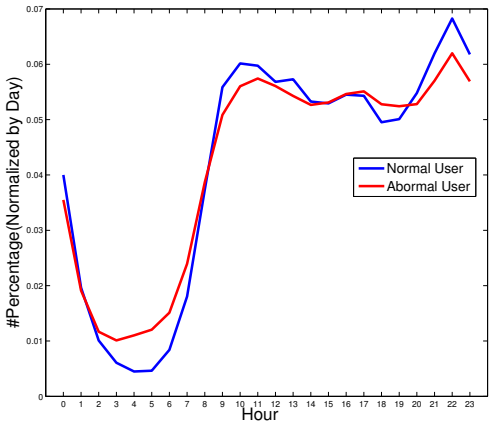
- *Certified users* These are users whose identities are certified by Sina Weibo Company. Thus, their real-life identities can be traced and obtained.
- *Robots* These are accounts whose tweets are sent from robots.
- *Information aggregators* These not-certified accounts mainly share news from media or rumors.
- *Marketing accounts* These not-certified accounts mainly share jokes or pieces of information related to commercial activities.
- *Others* All other users are classified into this category.

The distribution of users over these categories is shown in Figure 3. As expected, most robot accounts are abnormal users. It should be noted that some robots just post pieces information from other sources, such as Twitter, to Sina Weibo. Thus, they are not classified into abnormal users. Furthermore, for duplicated content, information aggregators are more welcome.

The number of tweets over users is shown in Figure 4. It is interesting that the distribution of abnormal users perfectly fits a power-law distribution, while that of normal users does not. A possible explanation to this phenomena is that some normal users begin to use the Sina Weibo service later than others. Therefore, they have less tweets than expected. It



(a)



(b)

Figure 2. Distribution of tweets over (a) days in weeks and (b) hours in days.

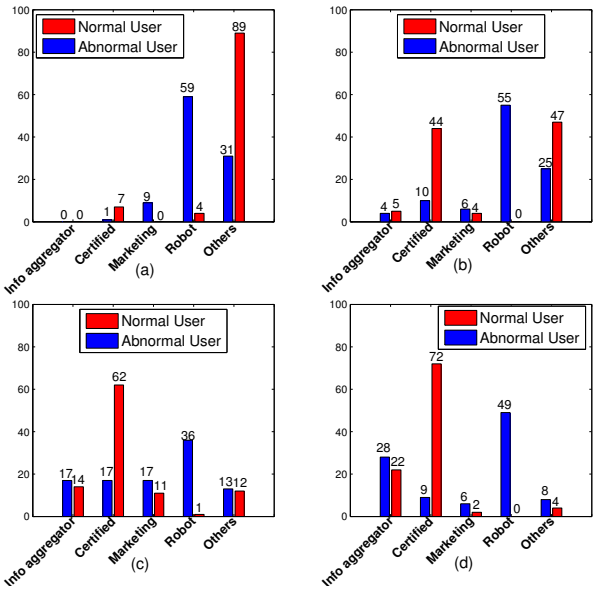


Figure 3. Distribution of users over five categories. The distribution of users who have followers in the range of $[100, 1,000)$, $[1,000, 10,000)$, $[10,000, 100,000)$, and $[100,000, 1,000,000)$, are shown in (a), (b), (c), and (d), respectively.

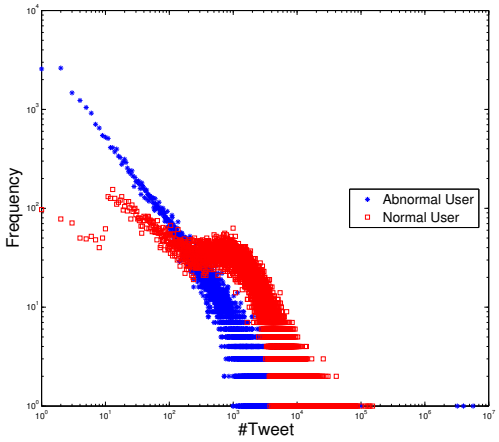


Figure 4. Distribution of number of tweets.

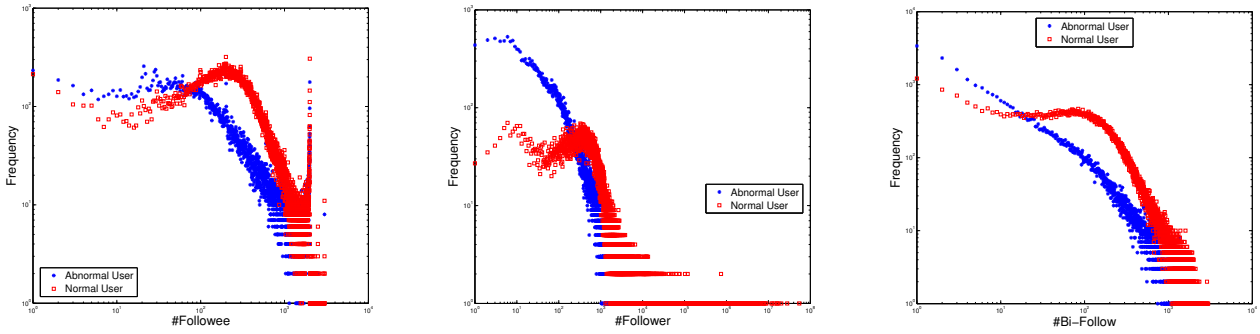
should also be noted that some abnormal users have an extremely large number of tweets. They are mainly robots.

The distribution of followees and followers are shown in Figure 5. It is shown that abnormal users tend to have less followees, followers, and bi-followships. Figure 6 shows that abnormal users have much less bi-followship rate, even for those users have many followers.

The *clustering coefficient* C_i of vertice v_i is defined as

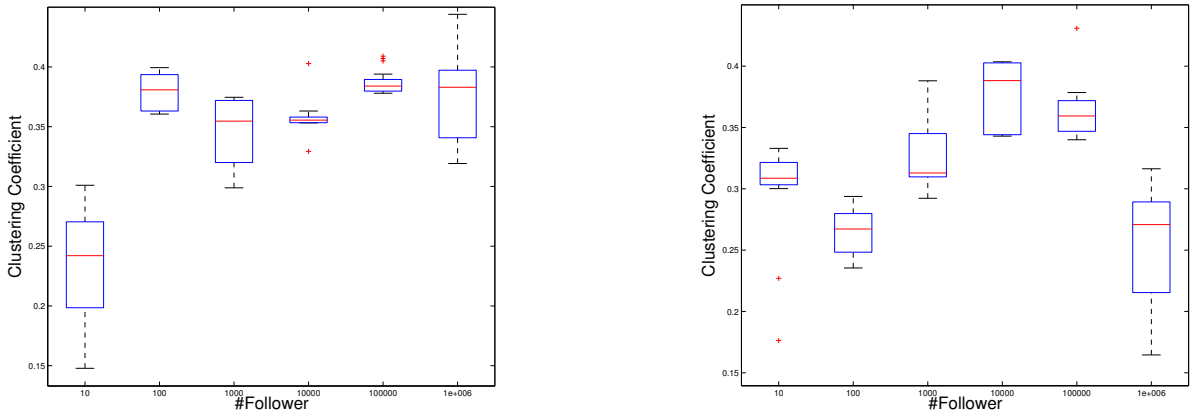
$$C_i = \frac{\|\{e_{jk}|e_{ji}, e_{ki} \in E, (e_{jk} \in E \text{ or } e_{kj} \in E)\}\|}{\|\{v_j|e_{ji} \in E\}\|}. \quad (4)$$

It can be used as a measurement of tightness of relationships of vertices in a graph. The distribution of clustering coefficient in box plot is shown in Figure 7. It is shown that abnormal users tend to have much less clustering coefficient for those who have many followers. Meanwhile, for those abnormal users who have very few followers, they have larger clustering coefficient. It is because that these users often form small communities and follow each other in the community.



(a) Distribution of followees. (b) Distribution of followers. (c) Distribution of bi-followships.

Figure 5. Distribution of followships.



(a) Normal users (b) Abnormal users

Figure 7. Distribution of clustering coefficient.

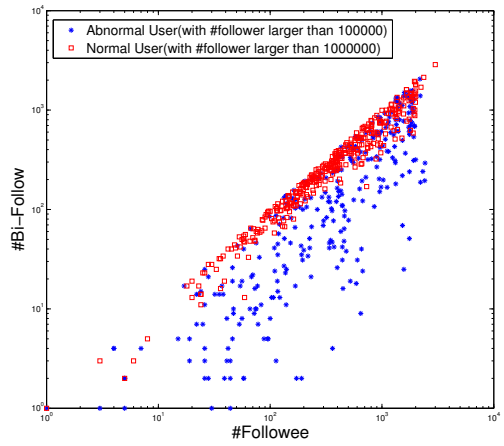


Figure 6. Distribution of bi-followships over followees for users whose followers are more than $10,000$.

V. Related works

Duplicate detection has received much attention in the past several years. Near-duplicate detection has been an important research problem for more than a decade[5], [7], [16], [36]. There are two main issues to be considered when solving an duplication detection problem, i.e. effciency and accuracy. With the growth of web data, applications of duplication detection typically need to handle a very large collection of documents. Hence, previous studies on content duplication put more emphasis on efficiency than accuracy.

Among extensive research work, hash-based methods were received more attention due to its ability of detecting duplication or near-duplication in high dimensional space. Zhang *et al*. proposed a new signature generation algorithm, which was based on learned hash functions for words. In order to deal with tens of billions of documents, they implemented the detection approach on graphical processing units (GPUs)[40]. Another novel approach was presented

that utilizes simhash[6], [16], [27] to find near-duplicates in large collections of documents. A probabilistic search technique in the Hamming space of simhashes was proposed in [34] that can be significantly faster and more space-effcient than the previous simhash approaches.

Local-sensitive hashing (LSH) is adopted in this paper to detect duplications. LSH was proposed by P. Indyk and R.Motwani in 1998[17]. An efficient implementation was then proposed[12]. It is based on the idea that high similarity data would possibly keep similar even processed by hash function. LSH has been widely used in different applications, such as content-based similarity search engine[23], agglomerative hiearchical clustering[19], and similarity learning[15].

Other methods for efficient and accurate duplicate detection exist. Stop word n-gram method has been shown to be reliable for identifying similarity in the document level[35]. A distributed filtering mechanism based on Bloom filters was proposed by Georgia Koloniari *et al*. to address the problem of large-scale duplicate-free delivery of events produced by distributed sources[20].

Microblogging services are studied intensively in recent years[18], [21], [39], [41], [25]. Previous research includes influence analysis[13], information diffusion network analysis[13], [41], and collective behavior study[41]. This paper focuses on a different microblog analysis problem, i.e. identifying social spammers in microblogs, which can also be treated as a preprocessing for further analytics over data from microblog.

Social spamming has attracted increasing interests in research community. Features including tags, content, ads within pages, and links were used for detecting social spams in Delicious[28]. Content-based method using clustering techniques was proposed for social spam detection on the Facebook data[11]. A honeybot scheme was designed for harvesting social spams[22]. Features of spams were analyzed based on data from Twitter and MySpace. Our research is different from existing reasearch work in that, we tend to identify those potential spammers. Thus, our work focuses on the behavior of those users, instead of the content of spams.

VI. Conclusions and discussion

A duplicate detection based method is proposed for identifying potential social spammers in microblogs. It is shown that the method is effective to find users with different behaviors and social networking characteristics.

Our future work includes the in-depth study of abnormal users with many duplicated tweets, and new methods for identifying spamming tweets based on both the content of tweets and the characteristics of authors.

Acknowledgment

This work is partially supported by National Science Foundation of China under grant numbers 61170086, and 60925008, National Basic Research (973 Program) under grant number 2010CB731402, and National High-tech R&D Program (863 Program) under grant number 2012AA011003.

The authors would like to thank Ms. Dandan Shi, for her efforts on developing the first version of the duplicated detection algorithm.

References

[1] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In U. Dayal, K.-Y. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y.-K. Kim, editors, *VLDB*, pages 918–929. ACM, 2006.

[2] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In Williamson et al. [38], pages 131–140.

[3] J. Bollen, H. Mao, and X.-J. Zeng. Twitter mood predicts the stock market. *CoRR*, abs/1010.3003, 2010.

[4] A. Z. Broder. On the resemblance and containment of documents. In *In Compression and Complexity of Sequences (SEQUENCES97*, pages 21–29. IEEE Computer Society, 1997.

[5] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

[6] M. Charikar. Similarity estimation techniques from rounding algorithms. In J. H. Reif, editor, *STOC*, pages 380–388. ACM, 2002.

[7] A. Chowdhury, O. Frieder, D. A. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, 2002.

[8] F. Crestani, S. Marchand-Maillet, H.-H. Chen, E. N. Efthimiadis, and J. Savoy, editors. *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*. ACM, 2010.

[9] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150. USENIX Association, 2004.

[10] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[11] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao. Detecting and characterizing social spam campaigns. In M. Allman, editor, *Internet Measurement Conference*, pages 35–47. ACM, 2010.

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *VLDB*, pages 518–529. Morgan Kaufmann, 1999.

[13] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *TKDD*, 5(4), 2012.

[14] Z. Gyöngyi, H. Garcia-Molina, and J. O. Pedersen. Combating web spam with trustrank. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *VLDB*, pages 576–587. Morgan Kaufmann, 2004.

[15] H. Hajishirzi, W. tau Yih, and A. Kolcz. Adaptive near-duplicate detection via similarity learning. In Crestani et al. [8], pages 419–426.

[16] M. R. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In E. N. Efthimiadis, S. T. Dumais, D. Hawking, and K. Järvelin, editors, *SIGIR*, pages 284–291. ACM, 2006.

[17] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In J. S. Vitter, editor, *STOC*, pages 604–613. ACM, 1998.

[18] A. Java, X. Song, T. Finin, and B. L. Tseng. Why we twitter: An analysis of a microblogging community. In *WebKDD/SNA-KDD*, pages 118–138, 2007.

[19] H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowl. Inf. Syst.*, 12(1):25–53, 2007.

[20] G. Koloniari, N. Ntarmos, E. Pitoura, and D. Souravlias. One is enough: distributed filtering for duplicate elimination. In Macdonald et al. [26], pages 433–442.

[21] H. Kwak, C. Lee, H. Park, and S. B. Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600, 2010.

[22] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: social honeypots + machine learning. In Crestani et al. [8], pages 435–442.

[23] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Ferret: a toolkit for content-based similarity search of feature-rich data. In Y. Berbers and W. Zwaenepoel, editors, *EuroSys*, pages 317–330. ACM, 2006.

[24] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In C. Koch, J. Gehrke, M. N. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C. Y. Chan, V. Ganti, C.-C. Kanne, W. Klas, and E. J. Neuhold, editors, *VLDB*, pages 950–961. ACM, 2007.

[25] H. Ma, W. Qian, F. Xia, X. He, J. Xu, and A. Zhou. Towards modeling popularity of microblogs. *Frontiers of Computer Science*, 7(2):171–184, 2013.

[26] C. Macdonald, I. Ounis, and I. Ruthven, editors. *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*. ACM, 2011.

[27] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In Williamson et al. [38], pages 141–150.

[28] B. Markines, C. Cattuto, and F. Menczer. Social spam detection. In D. Fetterly and Z. Gyöngyi, editors, *AIRWeb*, ACM International Conference Proceeding Series, pages 41–48, 2009.

[29] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, editors, *LREC*. European Language Resources Association, 2010.

[30] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *SODA*, pages 1186–1195. ACM Press, 2006.

[31] A. Rajaraman and J. D. Ullman. Mining of Massive Datasets.

[32] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, 2010.

[33] V. Satuluri and S. Parthasarathy. Bayesian locality sensitive hashing for fast similarity search. *PVLDB*, 5(5):430–441, 2012.

[34] S. Sood and D. Loguinov. Probabilistic near-duplicate detection using simhash. In Macdonald et al. [26], pages 1117–1126.

[35] E. Stamatatos. Plagiarism detection based on structural information. In Macdonald et al. [26], pages 1221–1230.

[36] M. Theobald, J. Siddharth, and A. Paepcke. Spotsigs: robust and efficient near duplicate detection in large web collections. In S.-H. Myaeng, D. W. Oard, F. Sebastiani, T.-S. Chua, and M.-K. Leong, editors, *SIGIR*, pages 563–570. ACM, 2008.

[37] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *ICWSM*, 2010.

[38] C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, editors. *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. ACM, 2007.

[39] L. Yu, S. Asur, and B. Huberman. What trends in chinese social media. *arXiv preprint arXiv:1107.3522*, 2011.

[40] Q. Zhang, Y. Wu, Z. Ding, and X. Huang. Learning hash codes for efficient content reuse detection. In W. R. Hersh, J. Callan, Y. Maarek, and M. Sanderson, editors, *SIGIR*, pages 405–414. ACM, 2012.

[41] A. Zhou, W. Qian, and H. Ma. Social media data analysis for revealing collective behaviors. In Q. Yang, D. Agarwal, and J. Pei, editors, *KDD*, page 1402. ACM, 2012.