Detecting Near-Duplicates in Large-Scale Short Text Databases

Caichun Gong[1,2], Yulan Huang [1,2], Xueqi Cheng[1], and Shuo Bai[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100080, P.R.C.
[2] Graduate School of Chinese Academy of Sciences, Beijing, 100049, P.R.C.
gongcaichun@software.ict.ac.cn, huangyulan@software.ict.ac.cn, cxq@ict.ac.cn, bs_7799@hotmail.com

Abstract. Near-duplicates are abundant in short text databases. Detecting and eliminating them is of great importance. SimFinder proposed in this paper is a fast algorithm to identify all near-duplicates in large-scale short text databases. An ad hoc term weighting scheme is employed to measure each term's discriminative ability. A certain number of terms with higher weights are seletect as features for each short text. SimFinder generates several fingerprints for each text, and only texts with at least one fingerprint in common are compared with each other. An optimization procedure is employed in SimFinder to make it more efficient. Experiments indicate that SimFinder is an effective solution for short text duplicate detection with almost linear time and storage complexity. Both precision and recall of SimFinder are promising.

Keywords: duplicate detection, short text, term weighting, optimization.

1 Introduction

The rapid technological improvements in Internet and telecommunication have led to an explosion of digital data. A large proportion of such data are short texts, such as mobile phone short messages, instant messages. It is reported that more than 1.58 billion mobile phone short messages are sent each day in Mainland China [1]. Tencent QQ has attracted more than 430 million users, and billions of instant messages are sent each day [2].

Duplicates are abundant in short text databases. In our investigation, more than 40% mobile phone short messages have at least one identical duplicate, and an even larger proportion of them are near-duplicates. Detecting and eliminating these duplicate short messages is of great importance for other short text language processing, such as clustering, opinion mining, topic detection and tracking, community uncovering. Identical duplicate short texts are easy to detect by standard hashing schemes. Identification of near-duplicate short texts is much more difficult because of the following reasons: First of all, a single short text contains usually less than 200 characters, which makes it difficult to extract effective features. Second, there are usually a huge number of texts in a short text database. Third, Informal abbreviations, transliterations and network languages are prevailing in short text databases [2].

T. Washio et al. (Eds.): PAKDD 2008, LNAI 5012, pp. 877–883, 2008.

In this paper, an algorithm called SimFinder is presented to detect near-duplicates in large-scale short text databases. An ad hoc weighting scheme is employed in SimFinder to make duplicate measure more precise. Only a few terms with higher weights are extracted as features. Fingerprints are generated from these features, and only short texts with the same fingerprint will compare with each other. An optimization solution is also proposed to reduce comparisons further.

2 Related Work

A variety of techniques have been developed to identify academic plagiarism [3,4,5,6], web page duplicates [7,8,9,10], duplicate database records [11,12]. Brin et al. have proposed a prototype system called *COPS* (COpy Protection System) to safeguard intellectual property of digital documents [3]. Shivakumar et al. have developed *SCAM* (Stand Copy Analysis Mechanism) as a part of the Stanford Digital Library project [4]. Broder finds it sufficient to keep each document a "sketch" of "shingles" to compute the resemblance of two documents. Any document pair with at least one common shingle is examined whether it exceeds the threshold for resemblence. Broder's shingling method works well on duplicate detection in AltaVista search engine [8].

Lyon et al. have investigated the theoretical background to automated plagiarism detection [5]. They observe that independently written texts have a comparatively low level of matching trigrams. The Ferret plagiarism system counts matching trigrams of a pair of documents [5,6]. Shivakumar presents two approaches to compute overlap between all web document pairs simultaneously. Both of them assume that only when document d_i and d_j share more than k fingerprints can they be candidate near-duplicates, where k is a predefined threshold [7].

Manku et al. show that Charikar's *simhash* [13] is practically useful for identifying near- duplicates in large-scale web page repository [9]. *Simhash* is a fingerprint technique enjoying the property that fingerprints of near-duplicates differ only in a small number of bit positions [9,13]. If the *simhash* fingerprints of two documents are similar, they are deemed to be near-duplicates.

3 SimFinder

As for a large-scale short text database, it is imporssible to detect near-duplicates by comparing texts with each other. A certain number of fingerprints are extracted from each text in SimFinder, and only short texts sharing same fingerprints are possible to be near-duplicates.

3.1 Term Weighting and Duplicate Degree

Terms play different roles in texts. Generally speaking, nouns, verbs and adjectives are more discriminative than adverbs, connectives, pronouns and numerals. It is improper to assign a same weight to all terms [14]. Since few terms will occur more than one time in a single short text, the traditional *tf-idf* scheme is inappropriate for short texts.

As for each set G of terms with the same part-of-speech, an empirical weight interval $[a,b]$ is associated in SimFinder, where a and b are the minimal and maximal weight

that may be assigned to terms in G respectively. Let weight interval of G be $[a,b]$, a simple linear interpolation is used to compute the weight of each term $t \in G$ as follows:

$$W(t) = \frac{(b-a)(F(t)-F')}{F-F'} + a \tag{1}$$

Where $F(t)$ is the frequency of term t in the background database, F and F' denote the frequency of the most and least frequently used term in G respectively. The weighting scheme of Equation 1 does not take term length into account. We notice that longer terms are usually more important than shorter terms. Let $|t|$ denote the length of term t, the long-term-preferred weighting scheme can be defined as follows:

$$W(t) = \frac{(b-a)(F(t)-F')}{F-F'} |t| + a|t| \tag{2}$$

Duplicate degree is a measure of similarity between two texts. Texts with duplicate degree higher than a predefined threshold θ are considered as near-duplicates. Let A and B be two texts, the standard duplicate degree called *Jaccard similarity* is defined as follows:

$$d(A,B) = \frac{|S(A) \bigcap S(B)|}{|S(A) \bigcup S(B)|} \tag{3}$$

Where $S(A)$ and $S(B)$ are the set of terms contained in text A and B respectively. All terms are considered as equal importance in Equation 3. Let $w(t_i)$ be the weight of term t_i, the weighted variant of duplicate degree can be defined as follows:

$$d(A,B) = \frac{\sum_{ti \in S(A) \bigcap S(B)} w(t_i)}{\sum_{tj \in S(A) \bigcup S(B)} w(t_j)} \tag{4}$$

3.2 Feature Extraction and Optimization

Since there are no blanks to mark words in Chinese texts, SimFinder segments each short text into a serial of terms. Terms are then sorted in descending order of their weights. Terms with higher weights are called discriminative terms and selected as features.

Remark 1: In real short text databases, when two short texts A and B are near-duplicates, most discriminative terms occur in both A and B. Near-duplicates differ usually only in connectives, pronouns, numerals, and punctuations.

Each N contiguous features (or called N-gram) are hashed into an integer as fingerprint. If two short texts A and B have no fingerprints in common, they are impossible to be near-duplicates. As a result, numerous unnecessary comparisons can be avoided.

As for text A with m terms, no more than $\lambda=k+N$ terms are necessary to be selected as features, where k is the minimal integer satisfying the following inequality:

$$\frac{\sum_{i=1}^{k} w(t_i)}{\sum_{i=1}^{m} w(t_i)} > \theta \tag{5}$$

Where $w(t_i)$ denotes the weight of term t_i and θ is the duplicate degree threshold.

Definition 1: Let $D=\{T_1,T_2,...,T_n\}$ be the text database, and ArB denote A is duplicated to B, then $R=\{(A,B)\mid ArB,A\in D,B\in D\}$ is called a duplicate relation on D. R is called a transitive duplicate relation if and only if $\forall(A,B)\in R,(B,C)\in R\Rightarrow(A,C)\in R$.

Remark 2: In real short text databases, duplicate transitivity holds in almost all cases. In other words, if ArB and BrC, A and C are near-duplicates in almost all cases.

If ArB and BrC, A and C are called a potential duplicate pair. In traditional text databases, duplicate relation does not always observe transitivity. While almost all short text databases satisfy Remark 2. With Remark 2, potential duplicate pairs can be safely regarded as near-duplicates, so the computation of duplicate degree is unnecessary.

4 Experiments and Evaluations

Various experiments have been conducted with two short text databases. One is a short message corpus composed of 12 million mobile phone short messages (735 megabytes), the other is a BBS title corpus with 5 million BBS titles (157 megabytes).

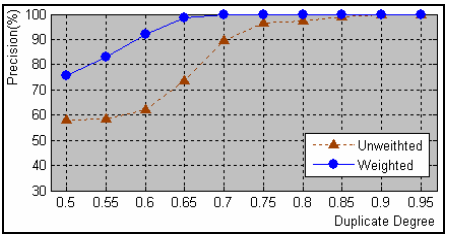

Fig. 1. The precision on short message corpus

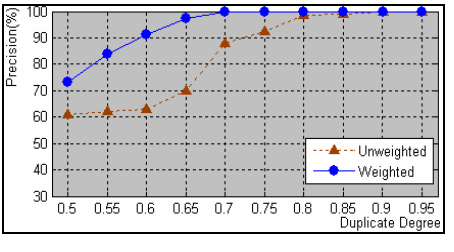

Fig. 2. The precision on BBS title corpus

Before we verify the effectiveness of SimFinder, a proper duplicate degree threshold must be determined. For each duplicate degree $d=0.50+0.05i\ (1\le i\le 10)$, 200 pairs of candidate duplicate short texts with duplicate degree in interval $[d-0.05, d]$ are

selected randomly and are checked manually whether they are near-duplicates. The precision of Equation 3 and Equation 4 are shown in Figure 1 and Figure 2. Experiments indicate that Equation 4 is more effective than Equation 3. The duplicate degree 0.65 is selected as the threshold because the precision is acceptable in both the short message corpus and the BBS title corpus.

A base-line algorithm is employed to generate all possible near-duplicate pairs. Texts with at least two continuous words in common are compared with each other. One million short messages with no identical duplicates have been used to choose gram size and feature number. The recall of algorithm *A* is defined as the ratio of the number of duplicate text detected by algorithm *A* to the number of duplicate text detected by the base-line algorithm. Figure 3 shows the effect of gram size on recall, and Figure 4 shows the effect of gram size on efficiency. *N=3* is selected in SimFinder because the recall is acceptable and the efficiency is promising.

Let $\lambda = k+N$, where *k* is defined in Ineqation 5, and *N* has been determined to be 3. Figure 5 and Figure 6 show the effect of feature number on recall and efficiency respectively. As can be seen that the feature number computed as Inequation 5 is feasible since the recall is almost 1. More features are unnecessary because the recall increases very little.

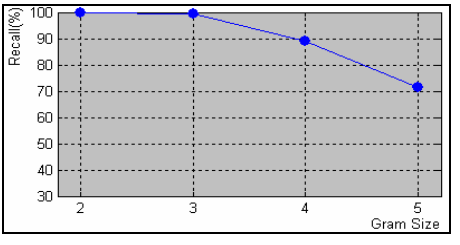

Fig. 3. The effect of gram size on recall

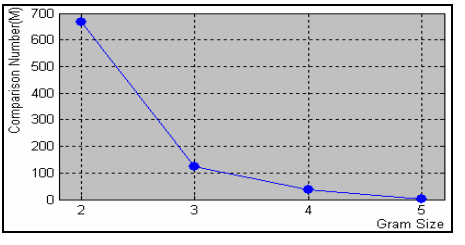

Fig. 4. The effect of gram size on comparison number

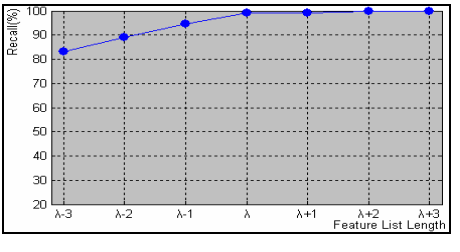

Fig. 5. The effect of feature number on recall

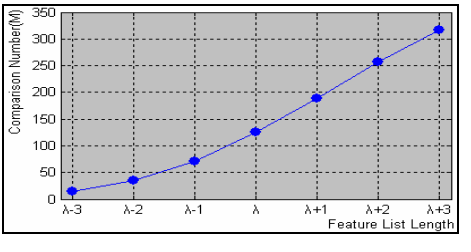

Fig. 6. The effect of feature number on comparison number

Ten thousand potential duplicate pairs are selected randomly to verify the correctness of Remark 2. For each potential duplicate pair *(A,B)*, duplicate degree *d(A,B)* is computed using Equation 4. Only 23 of them are less than 0.65, So the optimization has very little negative effect on precision. As for the short message corpus, if no

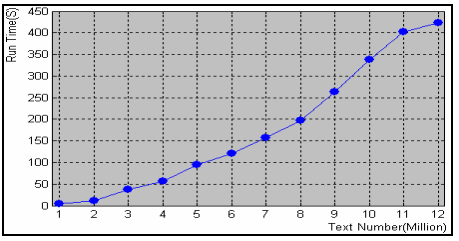

Fig. 7. Run times on short message corpus

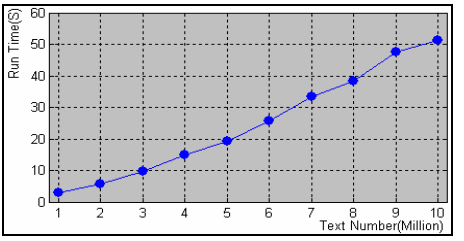

Fig. 8. Run times on BBS title corpus

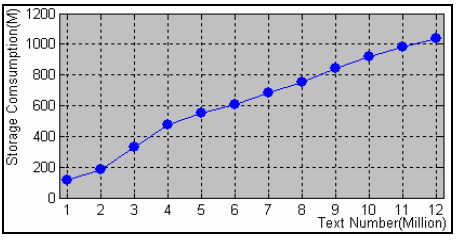

Fig. 9. Storage consumption on short message corpus

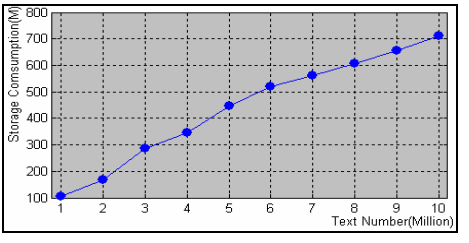

Fig. 10. Storage consumption on BBS title corpus

optimization procedure is included, the duplicate degree of 642,404,813 duplicate pairs must be computed using Equation 4. When optimization procedure is included, only 120,725,627 comparisons are needed. The optimization procedure increases the efficiency of SimFinder more than 4 times.

The SimFinder has been implemented in C++. We use a dawning server S4800A with 4 CPUs and 8G bytes of memory to test the performance of SimFinder. Figure 7 and Figure 8 show the run time of SimFinder on short message corpus and BBS title corpus respectively. Figure 9 and Figure 10 show the storage consumption of SimFinder. As can be seen that both run time and storage consumption are almost linear correlated with the size of corpus.

5 Conclusion

SimFinder is an effective and efficient algorithm to detect and eliminate duplicates in large-scale short text databases. Three techniques have been included in SimFinder: the ad hoc term weighting technique, the discriminative-term selection technique, the optimization technique. Experiments have shown that SimFinder is an encouraging solution for large-scale short text duplicate detection.

Acknowlegments. This research is supported by *The 973 National Basic Research Program of China* under the Grant NO. 2004CB318109 and 2007CB311100.

References

1. Website of Ministry of Information Industry of China, http://www.mii.gov.cn/
2. Hu, J.X.: Message text clustering based on frequent patterns (In Chinese). M.S. thesis, Institute of Computing Technology, Chinese Academy of Sciences. Beijing, China (2006)
3. Brin, S., Davis, J., Garcia-Molina, H.: Copy detection mechanisms for digital documents. In: Proceedings of the ACM SIGMOD Annual Conference, San Francisco, CA (May 1995)
4. Shivakumar, N., Garcia-Molina, H.: SCAM:A copy detection mechanism for digital documents. In: Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries, Austin, Texas (June 1995)
5. Lyon, C., Barrett, R., Malcolm, J.: A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector. In: Plagiarism: Prevention, Practice and Policies Conference (June 2004)
6. Lyon, C., Barrett, R., Malcolm, J.: Plagiarism is easy, but also easy to detect. Plagiary: Cross-Disciplinary Studies in Plagiarism, Fabrication, and Falsification 1(5), 1–10 (2006)
7. Shivakumar, N., Garnia-Molina, H.: Finding near-replicas of documents on the web. In: Proceedings of Workshop on Web Databases, Valencia, Spain (March 1998)
8. Broder, A.: Identifying and Filtering Near-Duplicate Documents. In: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, Montreal, Canada (June 2000)
9. Manku, G.S., Jain, A., Sarma, A.D.: Detecting near-duplicates for web crawling. In: Proceedings of the 16th International World Wide Web Conference, Banff, Alberta, Canada (May 2007)
10. Henzinger, M.: Finding near-duplicate web pages: A large-scale evaluation of algorithms. In: Proceedings of the 29th Annul International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, U.S.A (August 2006)
11. Tian, Z.P., Lu, H.J., Ji, W.Y., et al.: An n-gram-based approach for detecting approximately duplicate database records. International Journal on Digital Libraries 5(3), 325–331 (2001)
12. Hernandez, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, U.S.A (1995)
13. Charikar, M.: Similarity estimation techniques from rounding algorithms. In: Proceedings of 34th Annul Symposium on Theory of Computing, Montréal, Québec, Canada (May 2002)
14. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Information Processing and Management: an International Journal 24(5), 513–523 (1988)