Rearch on Large Scale Documents Deduplication Technique based on Simhash Algorithm

Yi Yu [a], Zijian Hu [b] and Yuzhu Zhang [c]

Chongqing University of Posts and Telecommunications, Chongqing 400065, China

[a]395066418@qq.com, [b]1006140793@qq.com, [c]1158863461@qq.com

Keywords: duplicated document detection; Simhash; Fingerprint calculation.

Abstract. On the background of the deduplication needs of repeated documents in Internet, research the deduplication technique based on Simhash algorithm on large-scale documents. On the basis of taking the Simhash algorithm as core algorithm in duplicated documents detection, improve the procedure of achieving documents features of this algorithm. It takes the meaning and length of words as a consideration factor in measuring the weight of words. Aiming at the Simhash signature of a 64-bit, provide the document service of making a similarity comparison based on the full text and paragraphs. Through test data and analysis, this technique can guarantee the stable operation, 100 million documents can be memorized in each example. The average request response time is about 20 ms. The respon.se time will increase during the peak hour, but, in general, will not go over 100 ms.

Introduction

In this era of information explosion, more and more documents on the network are duplicated, a large number of similar documents is a very common phenomenon, these duplicated documents will not only reduce the quality of products, but also for user-unfriendly, and how to avoid the emergence of a large number of duplicate or similar document is a problem. Some general hash function (e.g. SHA-1, MD5) is designed to make the hash values be uniform distribution as possible. Obviously, a small difference between different documents will get a quite different hash values. As a matter of course, we need a hash function that will get similar values when the inputs are similar.

Simhash algorithm is first applied by Google in the web deduplication system which mainly to solve mirror sites, content replication, embedded advertising, counting change, minor modifications problems and so on [1]. Simhash algorithm is a dimension reduction technique, high vectors can be mapped into a bit smaller fingerprints which is the number transformed from the feature of the information. The features make up usually a high-dimension vector and are hashed into a value by PRNG, so the fingerprint extraction is, namely, a dimensionality reduction technique. Simhash algorithm is based on a sequence of words and weights, due to a larger number of words on the page document, for computing and storage have higher requirements, and the word frequency information is merely considered into weights, the own characteristics and other information of web documents are not considered, so at the time of extracting text feature in this paper, the word frequency is not the only one index for measuring characteristics of document.

Simhash Algorithm

Moses Charka Simhash [2] is a kind of locality sensitive hash and is proven useful to detect near-duplicates by Manku's work [3]. If the difference between the documents' Simhash is much smaller, their similarity is much higher. Two same documents definitely have identical Simhash. It is same the other way around. This method has a simple and easy computation process and is widely used. Its algorithm is decomposed into three main steps as follows:

Initialize an f-dimension vector V, denoted by r1, r2… rN, and each RI is set to zero at first.

Referring to each f-bit hash value generated by feature item of a document, if i-th bit is 1, RI should plus the weight of this feature item, or minus.

Scan the vector once to change the vector V to a binary number by taking the following steps: if RI is positive, a bit of binary number should be set to 1, or set to 0. The final binary number is the document's Simhash code.

Quick Algorithm of Hamming Distance Problem

The Hamming distance [4] between two strings of equal length is the number of positions at which the corresponding symbols are different. That is to say, it measures the minimum number of substitutions required to change one string into the other.

Based on the Hamming distance, the detection task can come down to this definition [5]: Given a collection of *f*-bit fingerprints and a query fingerprint *F*, identify whether an existing fingerprint differs from *F* in at most *k* bits. There is a moderate approach both in time complexity and space complexity. It is based on such an intuitive assumption: in a sorted table of 2^d *f*-bit sorted fingerprints, consider *d*-bit out of each fingerprint, there are quite a few combinations, but few combinations are duplicated. So we choose *d'* who makes $|d'-d|$ a small integer. It is quick to find a collection *f'* of all fingerprint which match *F* in *d'*-bit position due to the sorted table. Since $|d'-d|$ is small, the number of fingerprint in *f'* is also expected to be small. This will shrink the size of the candidate collection and make the detection of identifying the hamming distance at most *k* with *F* more rapidly. The major procedure is as follows:

1) Firstly construct t sorted tables (T1, T2, T3 …, and Tt), each table contains two attributes: an integer pi and a permutation πi which is responsible for permuting some pi bits and top bits in a fingerprint. Once initializing, the πi will apply every existing fingerprints and the permuted fingerprints and the permuted fingerprints will be sorted.

2) Scan the tables in parallel way: we use πi (F) to denote the permuted query fingerprint F, for each table Ti, we gather the collections whose top pi bits are equal to πi (F)'s.

3) In above collections, return fingerprints that differ from πi (F) in at most k of the Hamming distance.

Take an example to demonstration. Suppose that there is a collection of 2^d fingerprints, our task is to find the fingerprints whose Hamming distance is at most *k*=3 with a given query fingerprint *F*. Now we put a 64-bit binary number into *t*=4 blocks uniformly, so each block has 16 bits. According to the pigeonhole principle, the similar Simhash codes share 1 same block at least. If any block is permuted up to top 16 bits, it will come out 4 combinations, then copy 4 tables correspondingly. For each query fingerprint *F*, take every block of *F* to match the fingerprints' top 16 bits in 4 tables respectively. Thanks to the less bits and sorted table, we take less time using binary search method. Afterwards, the rest is 64-16=48 bits, but at this moment, the size of candidate collection shrinks, we have only 2^{d-16} fingerprints.

Likewise, the rest 48 bits (i.e. 3 blocks) can also be split to 4 sub-blocks, thus there are 16 combinations that bring 16 tables, and each block occupies 12 bits. Finally, it will retrieve a much smaller collection of 2^{d-28} fingerprints.

Feature Weighting Computation for Simhash

In the above, we have discussed that the feature of a document is hashed into a value to generate the Simhash code. In Chinese environment, the feature is usually weighted by word frequency, but it doesn't reflect the sequence of the word appearance and integral structure of the document very well. Further, the rate and the efficiency of Chinese word segmentation are not comparable to the English's.

In this paper we make a little modification in feature weighting computation. We designed the weight calculation formula of a word:

$$W(w_i) = \mathrm{Score}(w_i) / \max_{w_j \in d}\{\mathrm{Score}(w_j)\} \quad (1)$$

W_i represents the word, $W(w_i)$ represents the weight of word, d represents the documents, $\text{Score}(w_i) = n_i + \lambda(\text{Len}(w_i) + \text{Pos}(w_i))$, n_i denotes word frequency, λ is a parameters, its value is related to the length of the document.

$$\text{Len}(w_i) = \begin{cases} 1, & if\ length(w_i) \geq 4 \\ 0, & otherwise \end{cases} \tag{2}$$

$$\text{Pos}(w_i) = \begin{cases} 1, & if\ w_i\ is\ a\ noun \\ 0, & otherwise \end{cases} \tag{3}$$

Length (w_i) represents the length of a word, the part of speech and the length of word are considered as important factors in measuring word weight and it is more comprehensive to characterize the feature of the document.

Simulation Results and Analysis

On the condition of 10 million test documents sorted in system, the relationship between the system request quantity and time consuming is tested, constantly increase the number of system requests per second, compare time-consuming for each request as shown in Fig.1:

As seen in the Fig.1, when the data of system are given, the relationship between the requests for system and the time consuming have the trend as following:

1) When the frequency of request for system is 2000 times per second, the time consuming is stable within 10ms, the frequency of request is not the bottleneck for system service in this condition;

2) When the frequency of request for system is 6000 times per second, the time consuming is stable within 20ms, the frequency of request is not the bottleneck for system service in this condition;

3) When the frequency of request for system is more than 6000 times per second, the time consuming increases significant, which indicates the frequency of request has become the bottleneck for system service;

Next, we test the relationship between the request quantity of system and time consuming when the frequency of requests is maintained at 4000 times per second at the beginning, then gradually increase the amount of data for document from 1 million to 100million, calculate and compare the time consuming of request, the testing results is shown as following Fig.2.

As shown in Fig.2, the time consuming of requests is stable within 20ms when the amount of documents are within 50 million. However, the amount of documents seriously affects the time consuming of requests, but the time consuming of requests are stable within 100ms when the amount of documents are more than 50 million. From the analysis of theory, the longer expiration time for document is set, the longer stored time is required, the more storage space is taken up, which affect the query efficiency, therefore the time consuming of request.

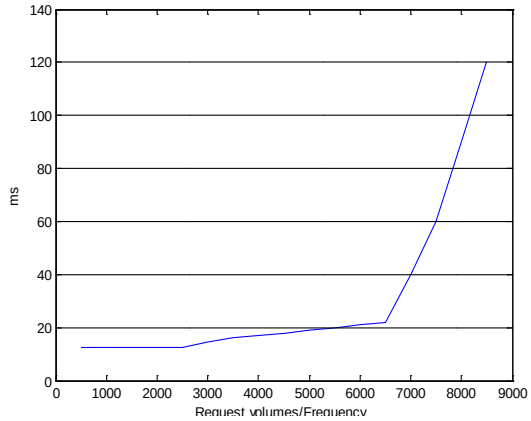

Fig.1 The relationship between request quantity and time-consuming

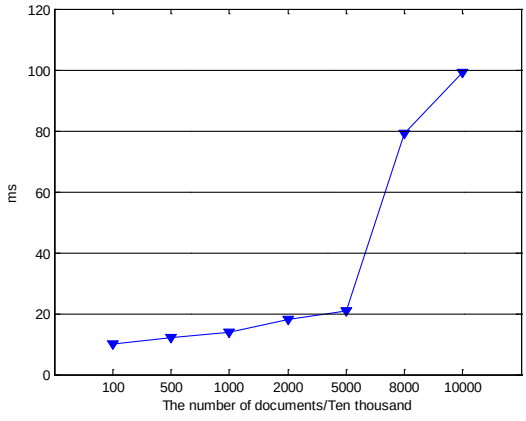

Fig.2 The relationship between request quantity and time-consuming

Summary

Detection of large-scale duplicate documents comes from a real application of data mining. We take a near duplicates detecting system which can make a real-time response to external request and improve the weight calculation formula of the feature in document based on Simhash algorithm. For Chinese document, on one hand this system can support the detection of 200 million documents, on the other hand it can provide multiple dimensions of documents detecting strategy for the fast and large scale document detection, the time-consuming of detection generally stable within 10 ms and will be close to 100 ms at peak hours of request, but generally not more than 100 ms, which realize real-time detection in large-scale deduplicated documents.

References

[1] Zhang Zu ping, Xu Xin, Long Jun, et al. Parameters correlation and optimization in text similarity measurement [J]. Journal of Chinese Computer Systems, 2011, 32(5): p. 983-988

[2] M. S. Charikar. Similarity estimation techniques from rounding algorithms [C]//ACM Symposium on the Theory of Computing, Canada, 2002, p. 380-388.

[3] G. S. Manku, A. Jain, A. Das Sarma. Detecting near duplicates for web crawling[C] // Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, 2007, p. 141-150.

[4] A. Aizawa. An information-theoretic perspective of tied measures [J]. Information processing & management, 2003, 39(1): p.45-65.

[5] Zhang Gang, Liu Ting, Zheng Shifu, et al. Fast Deletion Algorithm for Large Scale Duplicated Web Pages [C] // Chinese Information Processing Society of China 20th Anniversary Proceedings (Sequel), 2001, 11(18): p.25.