A Query-Dependent Duplicate Detection Approach for Large Scale Search Engines

Shaozhi Ye*, Ruihua Song, Ji-Rong Wen, and Wei-Ying Ma

Microsoft Research Asia
5F, Sigma Center, No 49 Zhichun Rd
Beijing, China 100080

Abstract. Duplication of Web pages greatly hurts the perceived relevance of a search engine. Existing methods for detecting duplicated Web pages can be classified into two categories, i.e. offline and online methods. The offline methods target to detect all duplicates in a large set of Web pages, but none of the reported methods is capable of processing more than 30 million Web pages, which is about 1% of the pages indexed by today's commercial search engines. On the contrary, the online methods focus on removing duplicated pages in the search results at run time. Although the number of pages to be processed is smaller, these methods could heavily increase the response time of search engines. Our experiments on real query logs show that there is a significant difference between popular and unpopular queries in terms of query number and duplicate distributions. Then, we propose a hybrid query-dependent duplicate detection method which combines both advantage of offline and online methods. This hybrid method provides not only an effective but also scalable solution for duplicate detection.

1 Introduction

The World Wide Web (WWW) has been growing rapidly in the past decades. More and more information is becoming available electronically on the Web. The tremendous volume of web documents poses challenges to the performance and scalability of web search engines. Duplicate is an inherent problem that search engines have to deal with. It has been reported that about 10% hosts are mirrored to various extents in a study including 238,000 hosts [8]. Consequently, many identical or near-identical results would appear in the search results if search engines do not solve this problem effectively. Such duplicates will significantly decrease the perceived relevance of search engines. Therefore, automatic duplicate detection is a crucial technique for search engines.

* The author is also with the Department of Electronic Engineering, Tsinghua University. This work was conducted and completed when he was a visiting student at Microsoft Research Asia.

"Duplicate documents" refer to not only completely identical documents but also those nearly identical documents. The typical method of duplicate detection uses certain similarity measures, such as syntactic similarity [3, 4, 5] or semantic similarity [11], to calculate the duplicate degree of two documents. Documents with duplicate degree higher than a predefined threshold are considered duplicate documents. In [4], the concept of resemblance is defined to capture the informal notion of "roughly the same". The resemblance *r(A,B)* of two documents *A* and *B* is defined as follows. First each document is transformed into a set of k-grams (or shingles) denoted as *S(.)*. Then resemblance is computed by:

$$r(A,B) = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|} \tag{1}$$

where $|S|$ is the size of the set *S*. In [11], documents are presented by term vector and cosine measure is used to calculate the semantic similarity between two documents. In this paper, we use syntactic similarity to detect duplicate documents.

The existing duplicate detection methods can be classified into two categories, namely the offline method and the online method. The offline method calculates document similarities among a large of Web pages and detects all duplicates at the pre-processing stage. On the contrary, the online method detects duplicates in the search result at run time. The offline method seems to be more appealing since duplicate detection is done at the data preparation phase and the response time and throughput of search engines will not be affected. However, the huge scale of the Web page collection makes it nearly infeasible to detect all duplicates in practice. As of today, the offline method has reported to be capable of dealing with 30 million web pages in 10 days [4]. Considering 3 billion web pages that are currently searchable via commercial search engines, the offline methods cannot meet the performance and scalability requirements in such scenarios.

The online methods can be viewed as local methods since they detect duplicate documents in the scope of the search result of each query, while the offline methods are taken as global methods since they detect duplicates in the whole collection. For the online methods, since the number of documents is small, the duplicate detection process could be made fast enough to add only a relatively small overhead to the response time. In addition, since few users check more than the first 3 result pages (about 30 web pages) returned by search engines [10], it is usually unnecessary to detect duplicates that are out of the top *n* documents in the result list and the duplicate detection process could be further speeded up. However, as duplicate detection needs to be performed for each query, the accumulated overheads may become a significant factor to slow down the response time and decrease the throughput of a search engine.

In this paper, we propose a hybrid method for duplicate detection which takes advantages of both offline and online methods while avoiding their shortcomings. The basic idea is to divide user queries into popular and unpopular queries by mining query logs. For a popular query, we detect duplicates in its corresponding inverted list offline. For a unpopular query, duplication detection is conducted at run time. Our experiments on real query logs show that there is a significant difference between popular and unpopular queries in terms of query number and duplication distribution.

And our hybrid method could achieve good performance and scalability on duplicate detection in large scale search engines.

The rest of the paper is organized as follows. In Section 2 we review the previous work on duplicate detection. In Section 3 we report several important observations through mining query logs, such as the frequency distribution of queries, the difference of duplicate degree between popular and unpopular queries, etc. Based on these observations, a query-dependent duplicate detection approach is proposed in Section 4. Finally we conclude the paper and discuss future works in Section 5.

2 Prior Work

The prior work of duplicate detection can be partitioned into three categories based on the ways to calculate document similarity – shingle based, term based, and image based algorithms. We review these algorithms respectively in this section.

2.1 Shingle Based Algorithms

The algorithms, such as [1] [3] [4] [5], are based on the concept of shingle. A shingle is a set of contiguous terms in a document. Each document is divided into multiple shingles and one hash value is assigned to each shingle. By sorting these hash values, shingles with same hash values are grouped together. Then the resemblance of two documents can be calculated based on the number of matching shingles.

Several optimization techniques have been proposed to reduce the number of comparisons made. [3] selects shingles with the lowest *N* hash values and removes shingles with high frequencies. In this way, [4] processes 30M web pages in 10 days. Another more efficient alternative is also discussed in [4], which combines several shingles into one super shingle and computes hash values of the super shingles. The super shingle algorithm does not count all overlaps and thus is much faster. However, the author noted that it does not work well for short documents and no detailed results are reported. In [5], exact copies are removed in advance and then made each line a shingle.

With the help of the hash strategy, the lower bound of computation complexity of these shingle based algorithms is $O(N*logN)$. However, when N is very large and the Web page collection can not be processed by a single computer, a distribution algorithm is needed and thus the computation complexity will be close to $O(N^2)$. As the size of document set increases, more computation time and storage space will be needed, making these algorithms only feasible for a relatively small number of the Web pages.

2.2 Term Based Algorithms

Term based algorithms [11] [12] use individual terms as the basic unit, instead of using continuous k-gram shingles. They focus on semantic similarity rather than syntactic similarity by discarding the structure information of documents, such as the edit distance of terms, paragraph and sentence structures. Cosine similarity between document vectors is usually used to calculate similarity between documents. Different from the shingle based algorithms, each document in the set has to be compared with all the others, so its computation complexity is $O(N^2)$. The largest set processed by term based algorithms contains only about 500K web pages [11].

[12] describes an online algorithm for rapid determining similarity among the document set returned by an information retrieval system. It uses a phrase recognizer to obtain the most important terms in a document and computes the similarity between documents based on these terms. It works for a small IR system. But for popular search engines which need to answer over 100M queries everyday, this method is not suitable because of it is expensive to compute.

2.3 Image Based Algorithms

Image based algorithms [7, 9] target to deal with documents stored as images and their main issues are those of image processing, rather than plain text document processing. These algorithms deal with scenarios that are less relevant to our problem here, so we refer readers to [7, 9] for detail.

3 Observations of Queries and Duplicates

We investigate a log file provided by MSN[1], which contains 32,183,256 queries submitted to MSN in one day. Totally 11,609,842 unique queries are extracted from the log. Statistical analysis is conducted to get insights of these queries and duplicates in their corresponding search results. Below we report three important observations from our analysis that lead to the design of our duplicate detection algorithm.

3.1 Distribution of Query Frequencies

It is well known that the occurrence number of Web queries follows an 80-20 rule, which means that the 20% most frequent query terms occupy 80% of the number of total query occurrences [10]. Some works have shown that the double log plot of rank-frequency distribution of queries approximately follows a Zipf distribution [10, 13], which means the occurrences of the popular queries take up a major part in the whole query set. For example, on analysis of AltaVista[2]'s query log, [6] reports only 13.6% queries occur more than 3 times and 25 most common queries form 1.5% of the total number of queries, despite being only 0.00000016% of the unique 154 million queries. In [10], it was found that the top 75 terms in frequency represent only 0.05% of all unique terms, yet they account for 9% of all 1,277,763 search terms in all unique

[1] http://search.msn.com
[2] http://www.altavista.com

queries. In [13], 2.56% and 5.40% queries in the two log data sets occur more than 10 times.

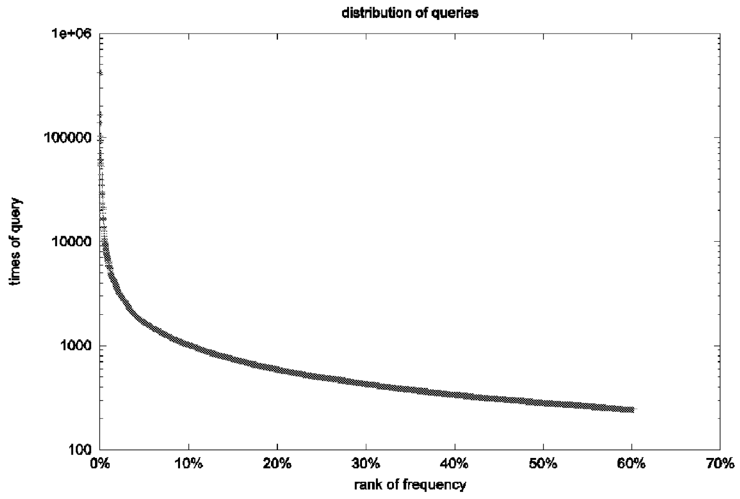

Fig. 1. Distribution of query frequency

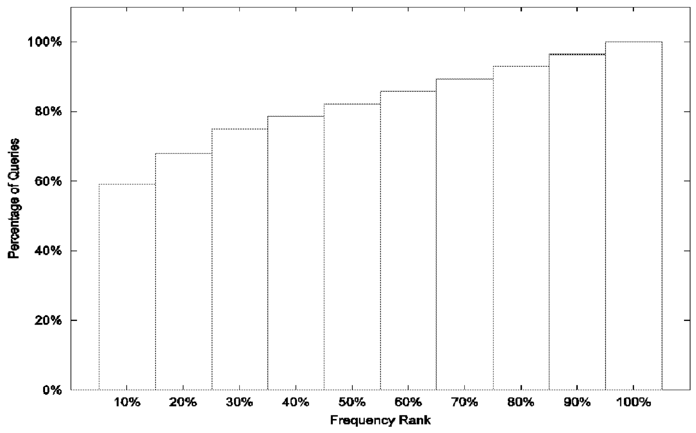

Fig. 2. Proportion of queries, ranked by query frequency

Here we revisit this phenomenon by analyzing the MSN log. Figure 1 is the distribution of query frequency in the log. X axis is the proportion of queries, which are ranked by their frequencies. Y axis is the number of occurrences of the queries (number is in log scale). It is shown that a small portion of queries is searched many times and its frequency decreases very quickly. Figure 2 is a clearer illustration of the proportions of frequent queries. The Y axis is the accumulated proportion of the top X most frequent queries. It shows that, in the MSN log, nearly 60% query occurrences are made up of 10% most frequent queries, and 70% query occurrences are made up of 20% most frequent queries.

The significance of the skewed query frequency distribution is that we can provide duplicate detection capability to most queries even if only a small portion of frequent queries are processed offline. For example, if the search results of the 10% most frequent queries are preprocessed to remove duplicates, we could directly return duplicate-free results to 60% queries submitted.

3.2 Duplicate Degrees for Popular and Unpopular Queries

The second problem we explored is to analyze if there is any difference of duplication degrees in search results between popular and unpopular queries. From the log, we randomly select 50 queries which are submitted more than 2,000 times as popular queries and 50 queries which are submitted exactly 10 times as unpopular queries. Google[3] supports the function of disabling the duplicate filter. If the option "filter=0" is appended to the search request URL, duplicate pages in the search result will not be filtered. Thus we use Google as our test bed by submitting these 100 queries to it with the duplicate filter option disabled. There are 10 web pages in every search result page returned by Google. We fetch the cached results in the first 10 result pages and get 100 results for each query.

Then we use shingle based algorithm in [4] to detect the duplicate documents. For each pair of detected duplicate documents, the one with lower rank is taken as duplicate and the other with higher rank as the source of duplicate (here we mean that rank 1 is higher than rank 2, rank 2 is higher than rank 3, and so on). We use a high threshold for similarity measure, that is, unless the resemblance is higher than 0.95, the two documents will not be judged as duplicates. Since 0.95 is a rather high (1.0 stands for exact match), resemblance here is considered transitive. So in the duplicate detection operation, we merge the duplicate list using the following rule: if document A is duplicate of document B and document C is duplicate of document A, then we treat document C as duplicate of A too. We leave out the one with highest rank in a duplicate set and treat others as duplicate documents.

The results of analysis on duplicate degrees of popular and unpopular queries are shown in Figure 3. The average duplicate degree in the search results of popular queries is about 5.5%, while that of unpopular ones is about 2.6%. It means that there are more duplicate documents in the search results of popular queries. This observation coincides with our intuition because popular queries usually are related to popular web pages and popular web pages tend to have more duplicates on the Web.

This observation indicates that users can benefit more from duplicate removal for popular queries since there are more duplicates in their search results.

3.3 Duplicate Distribution in Search Results

The third analysis we conducted is to investigate the duplicate distributions in the search results of popular and unpopular queries. If most of the duplicates have low ranks, they would not appear in the first several result pages. Thus users may not care

[3] http://www.google.com

too much about them and detecting duplicates in search results may be less needed since most users check no more than 3 search result pages [10]. As shown in Figure 4, the duplicate distribution of either popular queries or unpopular queries is nearly random. In other word, duplicates could appear in anywhere of search results. This observation confirms the need and importance of detecting and removing duplicates in search results.

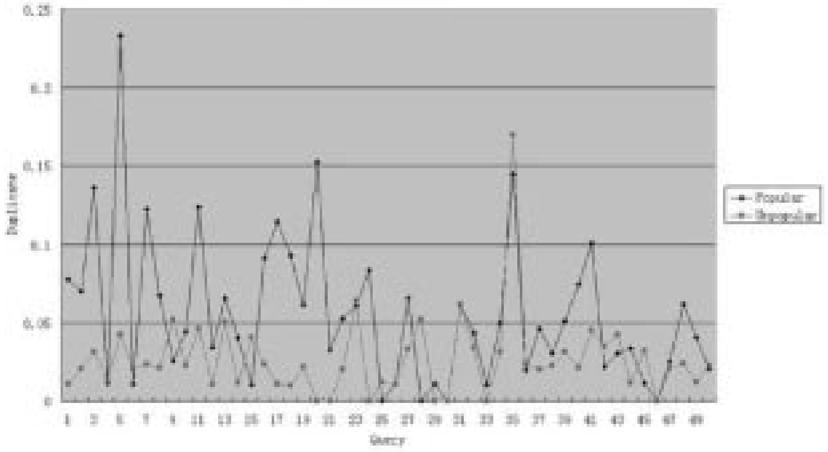

Fig. 3. Duplicate degree in retrieval result

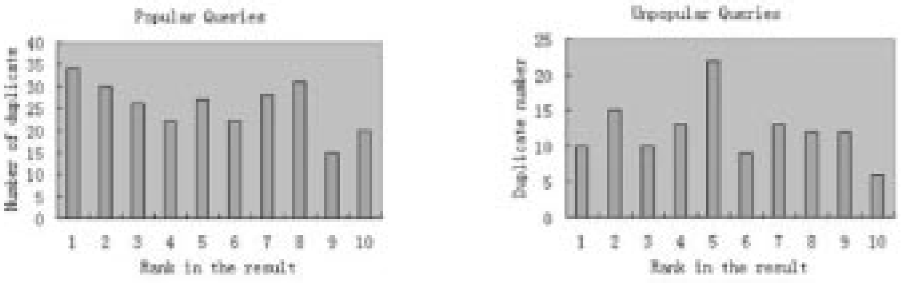

Fig. 4. Duplicate distributions in search results of popular and unpopular queries

4 Query-Dependent Duplicate Detection Algorithm

Most of the prior works use a query-independent strategy to detect duplicates in a collection of web pages. In this paper, we propose a query-dependent method for duplicate detection. Based on the three important observations in Section 3, we conclude that popular queries that occupy a major portion of the whole search requests have more duplicates in search results than unpopular queries. Also, duplicates could appear anywhere in search results. Therefore, we propose a hybrid method that intelligently takes advantage of query properties. For popular queries, duplicates are detected and removed using an offline method in the preprocess phase; for unpopular queries, we execute an online method to detect and remove duplicates at the run time.

4.1 Duplicate Detection for Popular Queries

Popular queries can be obtained from query logs through statistical analysis, as shown in Section 3.1. Most search engines use inverted file to index Web pages. An inverted index is made up of multiple inverted lists. An inverted list contains a term and the document IDs in which the term appears. For efficiency and easy implementation, we take advantage of the inverted index to conduct duplicate detection. However, standard inverted index only index separate terms and a query usually contains multiple terms. So we extend the inverted index by treating popular queries as an index unit (like “phrase”) and build inverted lists for these queries.

Duplicate detection is executed for each inverted list of popular queries. For each Web page, we only compare the shingles containing the queries to reduce the number of comparisons. We argue that this method has little impact on accuracy as in this case the goal is to detect duplicate “fragments” correlated to the query.

4.2 Duplicate Detection for Unpopular Queries

According to the analysis in Section 3.1, unpopular queries occur much less frequently than popular ones and the number of distinct unpopular queries is large. So, we could only deal with them at the run time. Otherwise, we will suffer the same scalability problem in traditional methods. Since the total occurrence number of unpopular queries is small, the impact of such an online method on the search performance can be managed.

In our implementation, only a few top-ranked result pages (e.g. including 1000 web pages) need to be processed because most users check no more than the first 3 search result pages. Also, only shingles containing the query are used for comparison. With these strategies, the online processing overhead is greatly reduced.

4.3 Performance Improvement

To verify the feasibility and performance of our algorithm, we design the following simulation experiment to show the performance improvement.

The data we used in the experiment is the query log we described in Section 3. We suppose that when duplicate detection is done online, the cost for each query is 1. If the search results of a query have been processed offline, there is no online computation cost (or very little in comparison with online processing cost). Then we increase the proportion of offline processing queries, and calculate the total online processing time.

Figure 5 shows the decrease of processing time (Y-axis) for online duplicate detection in proportion to the increase of amount of offline work (X-axis). The online processing time decreases quickly when X is small. On the other hand, more online processing time is needed when the number of offline processed queries increases. Obviously we have to find a best trade-off between offline and online processes for a better performance. This could be decided by the distribution of queries and other operational conditions such as intervals of index updating and the amount of user requests.

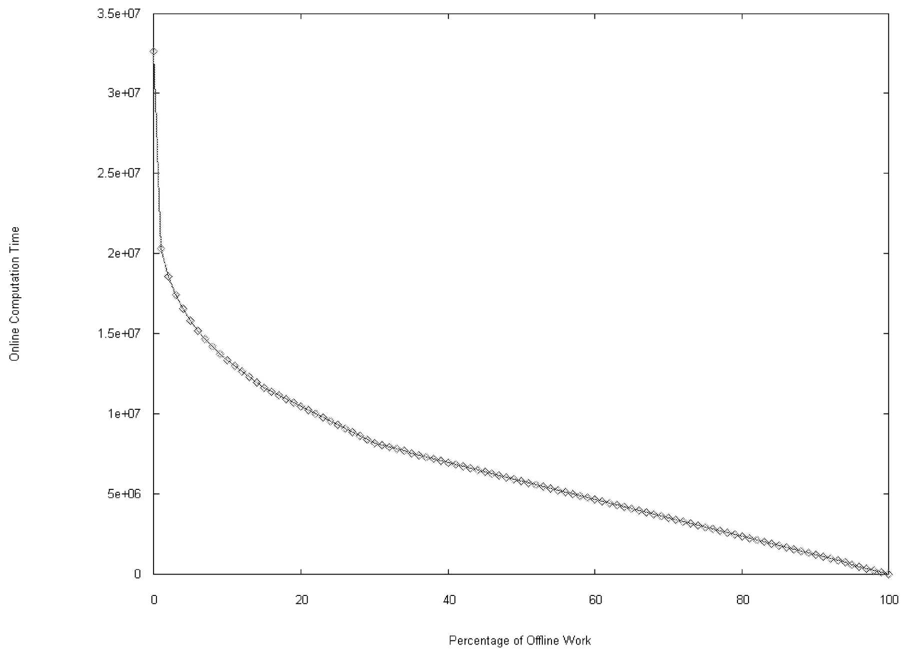

Fig. 5. Simulation experiments of performance improvement

Here we provide another analysis. The computation complexity of our proposed method is *O(N*M*logM)*, where *N* stands for the number of queries and *M* is the number of returned documents relevant to a query. According to Search Engine Watch[4], the busiest search engine serves 250M queries per day in Feb 2003. Based on the results in [6] and [10], we estimate that there are about 25% unique queries, which is 62.5M, and less than 1% queries occurring more than 100 times[5]. Assuming we process the top 10% queries and use the first 1,000 web pages returned for every query, the computation complexity of our proposed method will be $6.25*10^{10}$. Considering 3 billion web pages that are currently searchable on the Web, the computation complexity of traditional shingle based algorithms will be close to $9*10^{18}$! As can be seen, our proposed query-dependent algorithm is linear to the number of queries, and thus it is much more scalable than shingle based approaches.

[4] http://www.searchenginewatch.com

[5] Actually, according to our statistics, there are much less than 1% queries which occurs more than 100 times

5 Conclusion and Future Work

Three important observations on the properties of queries and duplicates were reported in this paper. First, based on MSN query logs, we found that popular queries consist of a major portion of the whole search requests. Thus duplicate detection can be omitted if a small portion of frequent queries are processed offline. Second, we found that popular queries often lead to more duplicates in the search results, so the benefit of duplicate removal for popular queries is more significant. Third, duplicates are found to distribute randomly in search results.

Based on these observations, we proposed a query-dependent duplicate detection scheme that combines the advantages of both online and offline methods. That is, it first conducts offline processing for popular queries and then does additional work at run time to further improve the performance for unpopular queries. Such a strategy could effectively deal with the scalability problem of traditional offline methods while avoiding the performance problem of traditional online methods.

Although syntactic duplicates could be detected in our methods, in our experimental results there are still many pages having almost identical contents but different formats, e.g., two same pages with different site templates. For these pages, we can not simply use a fixed threshold to determine if they are duplicates. We have to compare both content and template. To deal with this kind of duplicates, one possible solution is to detect the website's template [14], partition pages into blocks [15][16], discard the template blocks, and then compute the similarity of two pages based on their content blocks. We plan to explore this direction in our future work.

We also started to explore duplicate detection in newsgroup and news search on the Web. We found that there are much more duplicates in these data than general Web pages. We think that duplicate detection will also greatly improve the performance of retrieval results in these two types of web search.

References

1. Sergey Brin, James Davis, and Hector Garcia-Molina. Copy Detection Mechanisms for Digital Documents. In Proceeding of the Special Interest Group on Management of Data (SIGMOD'95), pp.298-409, 1995
2. Peter J. Denning, Plagiarism in the Web, In Communications of the ACM, Vol.38, December 1995
3. Nevin Heintze, Scalable Document Fingerprinting, In Proceedings of the Second USENIX Electronic Commerce Worksop, pp.191-200, November 1996
4. Andrei Z. Broder, Steven C. Glassman and Mark S. Manasse, Syntactic Clastering of the Web. In Proceedings of the Sixth International World Wide Web Conference(WWW6), 1997
5. Narayanan Shivakumar and Hector Garica-Molina, Finding Near-Replicas of Documents on the Web, In International Workshop on the Web and Databases (WebDB98), 1998
6. Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz, Analysis of a Very Large AltaVista Query Log. Technical Report 1998-014, Digital System Research Center, October 1998

7. Daniel P. Lopresti, Models and Algorithms for Duplicate Document Detection, In Proceedings of the 5th International Conference on Document Analysis and Recognition, September 1999
8. Krishna Bharat and Andrei Broder, Mirror on the Web: A Study of HostPairs with Replicated Content, In Proceedings of 8th International World Wide Web Conference (WWW8'99), pp.501-512, 1999
9. Mark Turner, Yuliya Katsnelson and Jim Smith, Large-Scale Duplicate Document Detection in Operation, In Proceedings of the 2001 Symposium on Document Image Understanding Technology, 2001
10. Amanda Spink, DeitmarWolfram, Bernard Jansen and Tefko Saracevic, Searching The Web: The Public and Their Queries, In Journal of the American Society for Information Science, Vol.53, No.2, pp.226-234, 2001
11. Abdur Chowdhury, Ophir Frieder, David Grossman and Mary Catherine McCabe, Collection Statistics for Fast Duplicate Document Detection, In ACM Transactions on Information Systems, Vol.20, No.2, pp.171-191, April 2002
12. James W. Cooper, Anni R. Coden and Eric W. Brown, Detecting Similar Documents using Salient Terms, In the 11th International Conference on Information and Knowledge Management (CIKM'02), November 2002
13. Yinglian Xie and David O'Hallaron, Locality in Search Engine Queries and its Implications for Caching, In Proceedings of IEEE Infocom'2002, June 2002
14. Ziv Bar-Yossef and Sridhar Rajagopalan, Temlate Detection via Data Mining and its Applications. In Proceedings of the 11th International World Wide Web Conference (WWW'2002), 2002
15. Shipeng Yu, Deng Cai, Ji-Rong Wen and Wei-Ying Ma, Improving Pseudo-Relevance Feedback in Web Infromation Retrieval Using Web Page Segmentation, In Proceedings of the 12th International World Wide Web Conference(WWW2003), pp.11-18, May 2003
16. Deng Cai, Shipeng Yu, Ji-Rong Wen and Wei-Ying Ma, Extracting Content Structure for Web Pages Based on Visual Representation, In Proceedings of the 5th Asia Pacific Web Conference(APWeb'03), pp.406-417, 2003