International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-1 Number-2 Issue-2 December 2011

The Study of Detecting Replicate Documents Using MD5 Hash Function

Pushpendra Singh Tomar[1], Maneesh Shreevastava[2]
M.Tech Research Scholar, LNCT Bhopal[1]
Head and Professor, IT LNCT Bhopal[2]

Abstract

A great deal of the Web is replicate or near-replicate content. Documents may be served in different formats: HTML, PDF, and Text for different audiences. Documents may get mirrored to avoid delays or to provide fault tolerance. Algorithms for detecting replicate documents are critical in applications where data is obtained from multiple sources. The removal of replicate documents is necessary, not only to reduce runtime, but also to improve search accuracy. Today, search engine crawlers are retrieving billions of unique URL's, of which hundreds of millions are replicates of some form. Thus, quickly identifying replicate detection expedites indexing and searching. One vendor's analysis of 1.2 billion URL's resulted in 400 million exact replicates found with a MD5 hash. Reducing the collection sizes by tens of percentage point's results in great savings in indexing time and a reduction in the amount of hardware required to support the system. Last and probably more significant, users benefit by eliminating replicate results. By efficiently presenting only unique documents, user satisfaction is likely to increase.

Keywords

Unique documents, detecting replicate, replication, search engine.

1. Introduction

The definition of what constitutes a replicate has somewhat different interpretations. For instance, some define a replicate as having the exact syntactic terms and sequence, whether having formatting differences or not. In effect, there are either no difference or only formatting differences and the contents of the data are exactly the same. In any case, data replication happens all the time. In large data warehouses, data replication is an inevitable phenomenon as millions of data are gathered at very short intervals. Data warehouse involves a process called ETL which stands for extract, transform and load. During the extraction phase, multitudes of data come to the data warehouse from several sources and the system behind the warehouse consolidates the data so each separate system format will be read consistently by the data consumers of the warehouse. Data portals are everywhere. The tremendous growth of the Internet has spurred the existence of data portals for nearly every topic. Some of these portals are of general interest; some are highly domain specific. Independent of the focus, the vast majority of the portals obtain data, loosely called documents, from multiple sources [1]. Obtaining data from multiple input sources typically results in replication. The detection of replicate documents within a collection has recently become an area of great interest [2] and is the focus of our described effort. Typically, inverted indexes are used to support efficient query processing in information search and retrieval engines. Storing replicate documents affects both the accuracy and efficiency of the search engine. Retrieving replicate documents in response to a user's query clearly lowers the number of valid responses provided to the user, hence lowering the accuracy of the user's response set. Furthermore, processing replicates necessitates additional computation. Replicates are abundant in short text databases. For example, popular mobile phone messages may be forwarded by millions of people, and millions of people may express their opinions on the same hot topic by mobile phone messages. In our investigation on mobile phone short messages, more than 40% short messages have at least one exact replicate. An even larger proportion of short messages are near-replicates. Detecting and eliminating these replicate short messages is of great importance for other short text processing, such as short text clustering, short text opinion mining, short text topic detection and tracking, short message community uncovering. Exact replicate short texts are easy to identify by standard hashing schemes. Informal abbreviations without introducing any additional benefit. Hence, the processing efficiency of the user's query is lowered. A problem introduced by the indexing of replicate documents is potentially skewed collection statistics. Collection statistics are often used as part of the similarity computation of a query to a document. Hence, the biasing of collection statistics may affect the overall precision of the entire

system. Simply put, not only is a given user's performance compromised by the existence of replicates, but also the overall retrieval accuracy of the engine is jeopardized. The definition of what constitutes a replicate is unclear. For instance, a replicate can be defined as the exact syntactic terms, without formatting differences. Throughout our efforts however, we adhere to the definition previously referred to as a measure of resemblance [3]. The general notion is that if a document contains roughly the same semantic content it is a replicate whether or not it is a precise syntactic match. When searching web documents, one might think that, at least, matching URL's would identify exact matches. However, many web sites use dynamic presentation wherein the content changes depending on the region or other variables. In addition, data providers often create several names for one site in an attempt to attract users with different interests or perspectives. For instance, Fox4, Onsale-Channel-9, and Real-TV all point to an advertisement for real TV.

Some forms of replicated content, such as those appearing in publications of conference proceedings, important updates to studies, confirmation of contested results in controversial studies, and translations of important findings, may no doubt be beneficial to the scientific community. Replication is seen as unethical when the primary intent is to deceive peers, supervisors, and/or journal editors with false claims of novel data. Given the large number of papers published annually, the large diversity of journals with overlapping interests in which to publish, and the uneven access to journal publication content, it is not unreasonable to assume that the discovery of such replication is rare [4]. The recent development of algorithmic methods to systematically process published literature and identify instances of replicated/plagiarized text as accurately as possible should serve as an effective deterrent to authors considering this dubious path. Unfortunately, the methods in place now have a very limited reach, and are confined to abstracts and titles only. Replicates: where they come from. One of the main problems with the existing geospatial databases is that they are known to contain many replicate points (e.g., [6] [7], [8]). The main reason why geospatial databases contain replicates is that the databases are rarely formed completely .from scratch., and instead are built by combining measurements from numerous sources. Since some measurements are represented in the data from several of the sources, we get replicate records. Why replicates are a problem. Replicate values can corrupt the results of statistical data processing and analysis. For example, when instead of a single (actual) measurement result, we see several measurement results confirming each other, and we may get an erroneous impression that this measurement result is more reliable than it actually is. Detecting and eliminating replicates is therefore an important part of assuring and improving the quality of geospatial data, as recommended by the US Federal Standard [9]. The identification of exact replicate documents in the Reuters collection was the primary goal of Sanderson [10]. The method utilized correctly identified 320 pairs and only failing to find four, thus proving its effectiveness. In the creation of this detection method, they found a number of other replicate document types such as expanded documents, corrected documents, and template documents. The efficient computation of the overlap between all pairs of web documents was considered by Shivakumar et al. [11]. The improvement of web crawlers, web archivers the presentation of search results, among others can be aided by this information. The statistics on how common replication is on the web was reported. In addition, the statistics on the cost of computing the above information for a relatively large subset of the web about 24 million web pages which correspond to about 150 gigabytes of textual information was presented.Many organizations archiving the World Wide Web show more importance in topics dealing with documents that remain unchanged between harvesting rounds. Some of the key problems in dealing with this have been discussed by Sigurðsson [12].Subsequently, a simple, but effective way of managing at least a part of it has been summarized which the popular web crawler Heritrix [14] employed in the form of an add-on module. They discussed the limitations and some of the work necessitating improvement in handling replicates, in conclusion. Theobald et al. [13] proved that SpotSigs provide both increased robustness of signatures as well as highly efficient replication compared to various state-of-the-art approaches. It was demonstrated that simple vector-length comparisons may already yield a very good partitioning condition to circumvent the otherwise quadratic runtime behavior for this family of clustering algorithms, for a reasonable range of similarity thresholds. Additionally, the SpotSigs replication algorithm runs "right out of the box" without the need for further tuning, while remaining exact and efficient, which is dissimilar to other approaches based on hashing. Provided that there is an effective means of bounding the similarity of two documents by a single property

such as document or signature length, the SpotSigs matcher can easily be generalized toward more generic similarity search in metric spaces.

2. Proposed Technique

This standard specifies four secure hash algorithms, SHA-1 [5], SHA-256, SHA-384, and SHA-512. All four of the algorithms are iterative, one-way hash functions that can process a message to produce a condensed representation called a message digest. These algorithms enable the determination of a message's integrity: any change to the message wills, with a very high probability, result in a different message digests. This property is useful in the generation and verification of digital signatures and message authentication codes, and in the generation of random numbers (bits). In cryptography, SHA-1 is a cryptographic hash function designed by the National Security Agency and published by the NIST as a U.S. Federal Information Processing Standard. SHA stands for "secure hash algorithm". The three SHA algorithms are structured differently and are distinguished as SHA-0, SHA-1, and SHA-2. SHA-1 is very similar to SHA-0, but corrects an error in the original SHA hash specification that led to significant weaknesses. The SHA-0 algorithm was not adopted by many applications. SHA-2 on the other hand significantly differs from the SHA-1 hash function. Each algorithm can be described in two stages: preprocessing and hash computation. Preprocessing involves padding a message, parsing the padded message into m-bit blocks, and setting initialization values to be used in the hash computation. The hash computation generates a message schedule from the padded message and uses that schedule, along with functions, constants, and word operations to iteratively generate a series of hash values. The final hash value generated by the hash computation is used to determine the message digest.The four algorithms differ most significantly in the number of bits of security that are provided for the data being hashed – this is directly related to the message digest length. When a secure hash algorithm is used in conjunction with another algorithm, there may be requirements specified elsewhere that require the use of a secure hash algorithm with a certain number of bits of security. For example, if a message is being signed with a digital signature algorithm that provides 128 bits of security, then that signature algorithm may require the use of a secure hash algorithm that also provides 128 bits of security (e.g., SHA-256).
Additionally, the four algorithms differ in terms of the size of the blocks and words of data that are used during hashing. Table 1 presents the basic properties of all four secure hash algorithms.

Table 1. Basic properties of all four secure hash algorithms

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Security2 (bits)
SHA-1	<264	512	32	160	80
SHA-256	<264	512	32	256	128
SHA-384	<2128	1024	64	384	192
SHA-512	<2128	1024	64	512	256

The performance numbers above were for a single-threaded implementation on an Intel Core 2 1.83 GHz processor under Windows Vista in 32-bit mode, and serve only as a rough point for general comparison. This function rapidly compares large numbers of files for identical content by computing the SHA-256 hash of each file and detecting replicates. The probability of two non-identical files having the same hash, even in a hypothetical directory containing millions of files, is exceedingly remote. Thus, since hashes rather than file contents are compared, the process of detecting replicates is greatly accelerated.

3. Test Result and Analysis

It is important to mention that this process does not have to be sequential: if we have several processors, then we can eliminate records in parallel, we just need to make sure that if two record are replicates, e.g., r1 = r2, then when one processor eliminates r1 the other one does not eliminate r2. To come up with a general algorithm for detecting and eliminating replicates under uncertainty, let us _rst consider an ideal case when there is no uncertainty, i.e., when replicate records ri = (xi; yi; di) and rj = (xj ; yj ; dj) mean that the corresponding coordinates are equal: xi = xj and yi = yj . In this case, to eliminate replicates, we can do the following. We _rst sort the records in lexicographic order, so that ri goes before rj if either xi < xj , or (xi = xj and yi • yj). In this order, replicates are next to each other. So, we _rst compare r1 with r2. If coordinates in r2 are identical to coordinates in r1, we eliminate r2 as a replicate, and compare r1 with r3, etc. After the next element is no longer a replicate, we take the next record after r1 and do the same for it, etc. After each comparison,

we either eliminate a record as a replicate, or move to a next record. Since we only have n records in the original database, we can move only n steps to the right, and we can eliminate no more than n records. Thus, totally, we need no more than 2n comparison steps to complete our procedure. Since 2n is asymptotically smaller than the time O(n ¢ log(n)) needed to sort the record, the total time for sorting and deleting replicates is O(n¢log(n))+2n = O(n¢log(n)). Since we want a sorted database as a result, and sorting requires at least O(n ¢ log(n)) steps, this algorithm is asymptotically optimal.

Algorithm:

For each record, compute the indices

pi = bxi=(C ¢ ")c; : : : ; qi = byi=(C ¢ ")c:

2. Sort the records in lexicographic order • by their index vector ~pi = (pi; : : : ; qi). If several records have the same index vector, check whether some are replicates of one another, and delete the replicates. As a result, we get an index-lexicographically ordered list of records:

r(1) • : : : • r(n0), where n0 • n.

3. For i from 1 to n, we compare the record r(i) with its •-following immediate neighbors; if one of the following immediate neighbors is a replicate to r(i), then we delete this neighbor.

4. Conclusion

We proposed a new replicate document detection algorithm called DRD and evaluated its performance using multiple data collections. The document collections used varied in size, degree of expected document replication, and document lengths. In terms of human usability, no similar document detection approach is perfect. The ultimate determination of how similar a document must be to be considered a replicate relies on human judgment. Therefore, any solution must be easy to use. To support ease of use, all potential replicates should be uniquely grouped together. Therefore, any match in even single results in a potential replicate match indication. This results in the scattering of potential replicates across many groupings, and many false positive potential matches. DRD, in contrast, treats a document in its entirety and maps all potential replicate s into a single grouping. This reduces the processing demands on the user. This paper has been felt necessary when the work on developing Replicate document detection is very hopeful, and is still in promising status. This survey paper intends to aid upcoming researchers in the field of Replicate document detection in web crawling to understand the available methods and help to perform their research in further direction.

References

[1] Broder, A., Glassman, S., Manasse, S., Zweig, G. 1997. Syntactic clustering of the web. In Proceedings of the Sixth International World Wide Web Conference (WWW6'97) (Santa Clara, CA., April). 391–404.

[2] Shivakumar, N., Garica-Molina, H. 1998. Finding near-replicas of documents on the web. In Proceedings of Workshop on Web Databases (WebDB'98) (Valencia, Spain, March). 204–212.

[3] Heintze, N. 1996. Scalable document fingerprinting. In Proceedings of the Second USENIX Electronic Commerce Workshop (Oakland, CA., November). 191–200.

[4] Sanderson,M. 1997. Replicate detection in the Reuters collection. Technical Report (TR-1997-5) of the Department of Computing Science at the University of Glasgow, Glasgow G12 8QQ, UK.

[5] The SHA-1 algorithm specified in this document is identical to the SHA-1 algorithm specified in FIPS 180-1.

[6] McCain, M., and William C., 1998. Integrating Quality Assurance into the GIS Project Life Cycle, Proceedings of the 1998 ESRI Users Conference.

[7] Goodchild, M., and Gopal, S. (Eds.), 1989. Accuracy of Spatial Databases, Taylor & Francis, London.

[8] Scott, L., 1994. Identi_cation of GIS Attribute Error Using Exploratory Data Analysis, Professional Geographer 46(3), 378.386.

[9] FGDC Federal Geographic Data Committee, 1998. FGDC-STD- 001-1998. Content standard for digital geospatial metadata (revised June 1998), Federal Geographic Data Committee, Washington, D.C.

[10] Sanderson, M., 1997. "Duplicate Detection in the Reuters Collection", Technical Report (TR-1997-5), Department of Computing Science, University of Glasgow.

[11] Shivakumar, N., Garcia Molina, H., 1999. "Finding near-replicas of documents on the web",Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Vol. 1590, pp. 204-212.

[12] Sigurðsson, K., 2006. "Managing duplicates across sequential crawls", proceedings of the 6thInternational Web Archiving Workshop.

[13] Theobald, M., Siddharth, J., Paepcke, A., 2008. "SpotSigs: Robust and Efficient Near Duplicate Detection in Large Web Collections", Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, Singapore, pp. 563-570.

[14] Mohr, G., Stack, M., Ranitovic, I., Avery, D., and Kimpton, M., 2004. "An Introduction to Heritrix", 4th International Web Archiving Workshop.