Efficient Web Crawling By Detecting and Shunning Near Duplicate Documents

J Prasanna Kumar[1] and Dr. P Govindarajulu[2]
[1]MVSR Engineering College, Osmania University, India
prasannakumarphd@gmail.com

Abstract:

Web crawling is an integral piece of infrastructure for search engines. Large volume public comment campaigns and web portals that encourage the public to customize form letters produce many near-duplicate documents, which increases processing and storage costs, but is rarely a serious problem. With the increasing amount of data and the need to integrate data from multiple data sources, a challenging issue is to find near duplicate records efficiently. Near-duplicate detection proceeds more smoothly and efficiently when there are clues about which documents are duplicates. In this paper, we have proposed an efficient novel approach for the detection of near duplicates in web crawling. Initially the crawled web pages are stored in the repositories to detect the near duplications. Then the keywords have been extracted from the web pages and the similarity score is determined between two pages. The web page which is having a similarity score greater than a predefined threshold value is considered as a near duplicate.

Keywords: *Data mining, Web mining, Web crawling, Web documents, Stemming, Common words, Keywords, Near-duplicate document detection, Similarity Score.*

1. Introduction

The World Wide Web (WWW) is a vast resource of multiple types of information in varied formats. With the huge amount of information available online, the WWW is a fertile area for data mining research. Data mining, often called Web mining when applied to the Internet, is a process of extracting hidden predictive information and discovering meaningful patterns, profiles, and trends from large databases [2]. The purpose of Web mining is to develop methods and systems for discovering models of objects and processes on the World Wide Web and for web-based systems that show adaptive performance [1]. The web mining research is at the cross road of research from several communities, such as database, information retrieval, especially the sub-areas of machine learning and natural language processing [6]. Web Mining integrates three parent areas: Data Mining, Internet technology and World Wide Web, and for the more recent Semantic Web. [1]. Web mining is an iterative process of discovering knowledge and is proving to be a valuable strategy for understanding consumer and business activity on the Web [2].

Web mining is the application of data mining techniques to extract knowledge from Web data, i.e. Web Content [4], Web Structure [5] and Web Usage data [6]. Web content mining is the process of mining knowledge from the web pages besides other web objects. The process of mining knowledge about the link structure linking web pages and some other web objects is defined as Web structure mining. Web usage mining is defined as the process of mining the usage patterns created by the users accessing the web pages [32]. Web mining can also be used to aid a user by integrating the implicit information from multiple sources of Web data. At the simplest level, it can be a keyword oriented search. [7]. However, as the web grows, it is becoming increasingly impractical to use the whole index of a search engine to determine site crawling. Web crawling has become an important aspect of web search, as the WWW keep getting bigger and search engines strive to index the most important and up to date content. Many experimental approaches exist, but few actually try to model the current behavior of search engines, which is to crawl and refresh certain sites they

deem as important much more frequently than other sites [8]. A Web search engine crawls the Web, indexes Web pages, and builds and stores huge keyword-based indices that help locate sets of Web pages that contain specific keywords. By using a set of tightly constrained keywords and phrases, an experienced user can quickly locate relevant documents [3].

Web crawling is the process used by search engines to collect pages from the Web. From the beginning, a key motivation for designing Web crawlers has been to retrieve Web pages and add them or their representations to a local repository. Such a repository may then serve particular application needs such as those of a Web search engine. In its simplest form a crawler starts from a seed page and then uses the external links within it to attend to other pages [9]. The crawler retrieves a URL from the frontier, downloads the web resource, extracts URLs from the downloaded resource and adds the new URLs to the frontier. The crawler continues in this manner until the frontier is empty or some other condition causes it to stop [11].

With the increasing amount of data and the need to integrate data from multiple data sources, a challenging issue is to find near duplicate records efficiently [20]. Near-duplicate web documents are abundant. Two such documents differ from each other in a very small portion that displays advertisements, for example. Such differences are irrelevant for web search. So the quality of a web crawler increases if it can assess whether a newly crawled web page is a near-duplicate of a previously crawled web page or not [10]. Duplicate and near-duplicate web pages are creating large problems for web search engines: They increase the space needed to store the index, either slow down or increase the cost of serving results, and annoy the users [12].

When crawling documents, when a near duplicate is detected, we can choose to ignore the document entirely, because we know that its contents are already represented in the index. The benefits of finding near duplicates also extend to the front end of the search process. Documents which are near duplicates may appear close together in search results, but provide little benefit to the user if they are not looking for a document in that particular subject. For example, suppose consecutive versions of a document have been crawled and are stored in the index. Instead of seeing multiple versions of the same “wrong” document in the search results, they can be collapsed together in a way to present more diverse results to the user [13].

Research on duplicate detection was initially done on databases, digital libraries, and electronic publishing. Lately duplicate detection has been extensively studied for the sake of numerous web search tasks such as web crawling, document ranking, and document archiving. A huge number of duplicate detection techniques ranging from manually coded rules to cutting edge machine learning techniques have been put forth [20, 19, 18, 15, 16, 17]. Recently few authors have projected near duplicate detection techniques [21, 14, 10]. A variety of issues such as from providing high detection rates to minimizing the computational and storage resources have been addressed by them. These techniques vary in their accuracy as well. Some of these techniques are computationally pricey to be implemented completely on huge collections. Even though some of these algorithms prove to be efficient they are fragile and so are susceptible to minute changes of the text.

The primary intent of our research is to develop a novel and efficient approach for detection of near duplicates in web documents. Initially the crawled web pages are preprocessed using document parsing which removes the HTML tags and java scripts present in the web documents. This is followed by the removal of common words or stop words from the crawled pages. Then the stemming algorithm is applied to filter the affixes (prefixes and the suffixes) of the crawled documents in order to get the keywords. Finally, the similarity score between two documents is calculated on basis of the extracted keywords. The documents with similarity scores greater than a predefined threshold value are considered as near duplicates.

The organization of the paper is as follows: In Section 2, a brief survey about the near duplicate page detection has been specified. In Section 3, the novel approach for the keywords based detection of near duplicate documents is presented. The computation of similarity scores for near duplicates detection is described in Section 4 and conclusions are summed up in Section 5.

2. Literature Review

The proposed research has been motivated by numerous existing works on and near duplicate documents detection.

Gurmeet Singh Manku et al. [10] made two research contributions while developing a near-duplicate detection system for a multi-billion page repository. Firstly they illustrated the relevance of Charikar's fingerprinting technique [28] for the ultimate goal. Secondly, they also put forth an algorithmic technique for the identification of existing f-bit fingerprints that diverge from a given fingerprint in at most k bit-positions, for small k. The technique proved beneficial for both online queries (single fingerprints) and batch queries (multiple fingerprints). Extensive experimental evaluation illustrated the practicality of the design.

An effective and efficient algorithm for the recognition and removal of duplicates in large-scale short text databases was projected by Caichun Gong et al [29]. The ad hoc term weighting technique, the discriminative-term selection technique and the optimization technique are the three techniques incorporated in SimFinder. SimFinder was found to be an effective solution for short text duplicate detection with almost linear time and storage complexity as a result of extensive experimentation.

Broder et al.'s [16] shingling algorithm and Charikar's [28] random projection based approach are regarded as the cutting edge algorithms for the identification of near-duplicate web pages. The comparison of both these algorithms on a large scale was carried out by Monika Henzinger [12] on a set of 1.6B distinct web pages. Results illustrated that both the algorithms were futile in recognizing near-duplicates on the same site nevertheless they perform exceptionally well in determining near-duplicate pairs on different sites. Owing to the fact that Charikar's algorithm was capable of identifying more near-duplicate pairs on different sites, it attains a better precision on the whole, 0.50 versus 0.38 for Broder et al.'s algorithm to be precise. A combined algorithm that attains a precision of 0.79 with 79% of the recall of the other algorithms was also presented by her.

Andrei Z. Broder [30] have presented a technique that can eradicate near-duplicate documents from a group of hundreds of millions of documents by calculating independently for each document a vector of features less than 50 bytes long and evaluating only these vectors instead of whole documents. This process consumes O(mlog m) time where m is the size of the collection. The project was implemented effectively and is presently utilized in the context of AltaVista search engine.

The utilization of simple text clustering and retrieval algorithms for the recognition of near-duplicate public comments was explored in detail by Hui Yang et al. [21]. The experiments with public comments regarding a modern regulation by the Environmental Protection Agency (EPA) illustrated the effectiveness of the algorithms.

The approximate elimination of duplicates in streaming environments given a limited space was targeted by Fan Deng et al. [18]. A data structure, Stable Bloom Filter (SBF), and a new and easy algorithm were introduced by them on basis of a renowned bitmap sketch. The fundamental idea is as follows: due to the fact that there was no way to store the whole history of the stream SBF expels stale information constantly in order to ensure that it has ample room for the more recent elements. Whenever a particular false positive rate was permitted SBF performed comparatively better than the alternative methods in terms of both accuracy and time for a fixed amount of space.

DURIAN (DUplicate Removal In lArge collectioN), a refinement of a prior near-duplicate detection algorithm that utilizes conventional bag-of-words document representation, document attributes ("metadata"), and document content structure to recognize form letters and their edited copies in public comment collections, was projected by Hui Yang et al. [14]. According to the experimental results DURIAN was equally good as the human counterparts. The research was concluded with a discussion on challenges in moving near-duplicate detection into operational rulemaking environments.

A document detection algorithm known as I-Match was projected and evaluated for its performance with the aid of multiple data collections by Abdur Chowdhury et al. [31]. The chief motive behind the development of I-Match was to support the web document collections. Therefore, contrasting many of its forerunners, I-Match proficiently processes large collections and does not ignore small documents. The collections of documents utilized were varied in size, degree of expected document duplication, and document lengths. NIST and Excite@Home served as sources for data acquisition.

3. Keywords Based Near Duplicate Documents Detection

This section gives the details about the novel approach for near duplicate detection. For the task of "remove all duplicates from this collection," it is helpful to get a list of duplicate document sets so that one from each set can be retained and the rest removed. Two such documents are identical in terms of content but differ in a small portion of the document such as advertisements, counters and timestamps. These differences are irrelevant for web search [10]. Recent duplicate detection research in the Web environment has focused on issues of computational efficiency and detection effectiveness. However, for efficient large scale web indexing it is not necessary to determine the actual resemblance value: it suffices to determine whether newly encountered documents are duplicates or near-duplicates of documents already indexed.

3.1 Near Duplicate Web Documents

A more recent manifestation of the problem is efficiently finding near-duplicate Web pages, which is particularly challenging in a Web-scale because of the huge data volume and the high dimensionality of documents [24]. It is a document with a given minimal percentage of identical shingle paragraphs of another document in the collection.

3.2 Web Crawling

A Web crawler is a program, which automatically traverses the web by downloading documents and following links from page to page. They are mainly used by web search engines to gather data for indexing [9]. Crawling is the most fragile application since it involves interacting with hundreds of thousands of web servers and various name servers, which are all beyond the control of the system. Web crawling speed is governed not only by the speed of one's own Internet connection, but also by the speed of the sites that are to be crawled. Especially if one is a crawling site from multiple servers, the total crawling time can be significantly reduced, if many downloads are done in parallel [26]. The crawling loop involves picking the next URL to crawl from the frontier, fetching the page corresponding to the URL through HTTP, parsing the retrieved page to extract the URLs and application specific information, and finally adding the unvisited URLs to the frontier. Before the URLs are added to the frontier they may be assigned a score that represents the estimated benefit of visiting the page corresponding to the URL. The crawling process may be terminated when a certain number of pages have been crawled [27].

3.3 Parsing of Web Documents

Once a page has been crawled, we need to parse its content to extract information that will feed and possibly guide the future path of the crawler. Parsing might also involve steps to convert the extracted URL to a canonical form, remove stopwords from the page's content and stem the remaining words [27]. HTML Parsers are freely available for many different languages. They provide the functionality to easily identify HTML tags and associated attribute-value pairs in a given HTML document.

3.4 Common Words Removal

When parsing a Web page to extract content information or in order to score new URLs suggested by the page, it is often helpful to remove commonly used words or stopwords such as "it" and "can". This process of removing stopwords from text is called stoplisting [27].

3.5 Stemming Algorithm

Stemming algorithms, or stemmers, are used to group words based on semantic similarity. Stemming algorithms are used in many types of language processing and text analysis systems, and are also widely used in information retrieval and database search systems [25]. A stemming algorithm is an algorithm that converts a word to a related form. One of the simplest such transformations is conversion of plurals to singulars, another would be the derivation of a verb from the gerund form (the "-ing" word). A number of stemming or conflation algorithms have been developed for IR (Information Retrieval) in order to reduce morphological variants to their root form. A stemming algorithm would normally be used for document matching and classification by using it to convert all likely forms of a word in the input document to the form in a reference document [22]. Stemming is usually done by removing any attached suffixes, and prefixes from index terms before the assignment of the term. Since the stem of a term represents a broader concept than the original term, the stemming process eventually increases the number of retrieved documents [23].

3.6 Keywords Representation

We posses the distinct keywords and their counts in each of the crawled web page as a result of stemming. These keywords are then represented in a form to ease the process of near duplicates detection. Initially the keywords and their number of occurrences in a web page have been sorted in descending order based on their counts. The n numbers of keywords with highest counts are stored and used to calculate the similarity measures. In our approach the value of n is set to be 3. The similarity score between two pages can be calculated if and only the main keywords of the two pages are similar.

4. Similarity Score Computation

A quantitative way to defining that two pages are near duplicates is to use a similarity score. The similarity score measures degree of similarity between two pages. A lower similarity value indicates that the pages are more similar. Thus we can treat pairs of pages with low similarity value as near duplicates. A similarity value will find all pairs of pages whose similarities are above a given threshold.

If the main keywords of the new web page are same with a page in a repository, then we have to calculate the similarity scores of all the keywords. The similarity score between two web pages is calculated as follows:

The keywords of the web pages and their counts of the keywords are represented as follows:

$$WP_1 = \{(K_{W1}, C_1), (K_{W2}, C_2), (K_{W4}, C_4), ..., (K_{Wn}, C_n)\}$$

$$WP_2 = \{(K_{W2}, C_2), (K_{W1}, C_1), (K_{W4}, C_4), ..., (K_{Wm}, C_m)\}$$

Initially we have to find the similarity measure for all the keywords in a first page WP_1. This is calculated by taking the difference between the number of occurrences of both the keywords. If the keyword is not present in another web page then their frequency is considered as zero.

$$S_S[WP_1] = \frac{1}{N_1} \sum_{i=1}^{N_1} Abs \left[count(K_{Wi})_{P_1} - count(K_{Wi})_{P_2} \right] ; \; if \; \mathrm{K_{Wi}} \notin \mathrm{WP_2} \text{ then count} = 0$$

where $N_1 = |WP_1|$.

Then the remaining keywords (R_{KW}) have been taken and find the similarity measure for those keywords in another web page WP_2.

$$R_{KW} = WP_2 - WP_1$$

$$S_S[WP_2] = \frac{1}{N_2} \sum_{i=1}^{N_2} Abs\ [count(K_{Wi})]_{R_{KW}}$$

where $N_2 = | R_{KW} |$.

Eventually, the final similarity score is calculated as follows:

$$S_S[K_W] = S_S[WP_1] + S_S[WP_2]$$

The web documents with final similarity score greater than a predefined threshold are considered as near duplicates of documents already present in repository.

5. Conclusion

The growth of the Internet challenges Internet Search Engines as more copies of Web documents flood over search results making them less relevant to users. The rapid growth of the World Wide Web poses unprecedented scaling challenges for general-purpose crawlers and search engines. The detection of duplicate and near duplicate web documents has gained more attention in recent years amidst the web mining researchers. We have presented a novel approach to detect the near duplicate among web pages. The proposed approach has detected the duplicates and near duplicates efficiently based on the extracted keywords and their similarity scores. This approach provides better search engine quality and the reduced memory space for repositories.

References

[1] Bettina Berendt, Andreas Hotho, Dunja Mladenic, Maarten van Someren, Myra Spiliopoulou, Gerd Stumme, "A Roadmap for Web Mining:From Web to Semantic Web", Lecture Notes in Artificial Intelligence , Vol. 3209, Springer-Verlag , pp. 1-22, 2004.

[2] Jiye Ai, James Laffey, "Web Mining as a Tool for Understanding Online Learning", MERLOT Journal of Online Learning and Teaching , Vol. 3, No. 2, June 2007.

[3] Jiawei Han, Kevin, Chen-Chuan, Chang, "Data Mining for Web Intelligence", In Proceedings of IEEE Computer, pp: 64-70, 2002.

[4] Pazzani, M., Nguyen, L., and Mantik, S., “Learning from hotlists and coldlists: Towards a www information filtering and seeking agent”, Proceedings, Seventh International Conference on Tools with Artificial Intelligence, pp. 492 – 495, 1995.

[5] David, G., Jon, K., and Prabhakar R., “Inferring web communities from link topology”, Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space-structure in hypermedia systems, pp.225 – 234, 1998.

[6] Kosala, R., and Blockeel, H., “Web Mining Research: A Survey,” SIGKDD Explorations, vol. 2, Issue. 1, 2000.

[7] P. Desikan, C. DeLong, K. Beemanapalli, A. Bose and J. Srivastava, “Web Mining For Self Directed E-Learning”, Book Chapter in Data Mining for E-Learning, pp. 21 - 37, WIT Press, 2005 .

[8] Alexandros Batzios, Christos Dimou, Andreas L. Symeonidis and Pericles A. Mitkas, "BioCrawler: An Intelligent Crawler for the Semantic Web", Expert Systems with Applications, Volume 35, Issues 1-2, Pages 524-530 , 2008.

[9] S. Balamurugan, Newlin Rajkumar, and J.Preethi, "Design and Implementation of a New Model Web Crawler with Enhanced Reliability", proceedings of world academy of science, engineering and technology, Vol: 32, august 2008

[10] Gurmeet S. Manku, Arvind Jain, Anish D. Sarma, "Detecting near-duplicates for web crawling," Proceedings of the 16th international conference on World Wide Web, pp: 141 – 150, 2007.

[11] F. McCown, and M.L. Nelson. “Evaluation of Crawling Policies for a Web-Repository Crawler”, 17th ACM Conference on Hypertext and Hypermedia (HYPERTEXT 2006). pp: 157-168, August 23-25, 2006.

[12] M. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," in SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, pp. 284-291, 2006.
[13] Shreyes Seshasai, "Near Duplicate Document Detection for Google Enterprise Search", Massachusetts institute of technology, USA, 2008.
[14] Yang, H., Callan, J., Shulman, S., "Next steps in near-duplicate detection for eRulemaking", Proceedings of the 2006 international conference on Digital government research, Vol. 151, pp: 239 – 248, 2006.
[15] Brin, S., Davis, J., and Garcia-Molina, H., "Copy detection mechanisms for digital documents", In Proceedings of the Special Interest Group on Management of Data (SIGMOD 1995), ACM Press, pp.398–409, May 1995.
[16] Broder, A. Z., Glassman, S. C., Manasse, M. S. and Zweig, G., "Syntactic clustering of the web", Computer Networks, vol. 29, no. 8-13, pp.1157–1166, 1997.
[17] Conrad, J. G., Guo, X. S., and Schriber, C. P., "Online duplicate document detection: signature reliability in a dynamic retrieval environment", Proceedings of the twelfth international conference on Information and knowledge management, pp. 443 -452, 2003.
[18] Deng, F., and Rafiei, D., "Approximately detecting duplicates for streaming data using stable bloom filters," Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 25-36, 2006.
[19] Henzinger, M., "Finding near-duplicate web pages: a large-scale evaluation of algorithms," Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 284-291, 2006.
[20] Xiao, C., Wang, W., Lin, X., Xu Yu, J., "Efficient Similarity Joins for Near Duplicate Detection", Proceeding of the 17th international conference on World Wide Web, pp:131-140, 2008.
[21] Yang, H., and Callan, J., "Near-duplicate detection for eRulemaking", Proceedings of the 2005 national conference on Digital government research, pp: 78 – 86, 2005.
[22] Brijesh Shanker Singh, "Search Algorithms", Documentation Research and Training Centre. Indian Statistical Institute, 2004.
[23] Lakshmi, K. V. Developing a word-stemming program using Porter's Algorithm, NCSI minor project report, 2002.
[24] Fan Deng, Davood Rafiei, "Estimating the Number of near Duplicate Document Pairs for Massive Data Sets using Small Space", University of Alberta, Canada, 2007.
[25] Frakes, W.B., Fox, C.J, "Strength and Similarity of Affix Removal Stemming Algorithms", In proceedings of the ACM SIGIR Forum. Vol: 37, pp: 26–30, 2003.
[26]" Monica Peshave, Kamyar Dezhgosha, "How Search Engines Work and a Web Crawler Application", Department of Computer Science, University of Illinois, Springfield USA, 2007.
[27] Pant, G., Srinivasan, P., Menczer, F., "Crawling the Web". Web Dynamics: Adapting to Change in Content, Size, Topology and Use, edited by M. Levene and A. Poulovassilis, Springer- verlog, pp: 153-178, November 2004.
[28] Charikar, M., "Similarity estimation techniques from rounding algorithms". In Proc. 34th Annual Symposium on Theory of Computing (STOC), pp: 380-388, 2002.
[29] Caichun Gong, Yulan Huang, Xueqi Cheng, Shuo Bai, "Detecting Near-Duplicates in Large-Scale Short Text Databases", PAKDD, pp: 877-883, 2008.
[30] A. Z. Broder, "Identifying and filtering near-duplicate documents," in COM '00: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching. London, UK: Springer-Verlag, pp. 1-10, 2000,
[31] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine Mccabe, "Collection Statistics for Fast Duplicate Document Detection", In. ACM Transactions on Information Systems (TOIS), Volume. 20, Issue 2, 2002.
[32] Lim, E.P., and Sun, A., (2006) "Web Mining - The Ontology Approach", International Advanced Digital Library Conference (IADLC'2005), Nagoya University, Nagoya, Japan.

Article received: 2009-02-17