V. Rajapriya, International Journal of Computer Science and Mobile Applications,
Vol.2 Issue. 5, May- 2014, pg. 36-44 **ISSN: 2321-8363**

A LITERATURE SURVEY ON WEB CRAWLERS

V. Rajapriya

School of Computer Science and Engineering, Bharathidasan University, Trichy, India
rajpriyavaradharajan@gmail.com

ABSTRACT: The web contains large data and it contains innumerable websites that is monitored by a tool or a program known as Crawler. Forum Crawler Under Supervision is a supervised web-scale forum crawler. The goal is to crawl relevant forum content from the web with minimal overhead. Forums have different layouts or styles and are powered by different forum software packages. They have similar implicit navigation paths connected by specific URL types to lead users from entry pages to thread pages. It reduces the web forum crawling problem to a URL type recognition problem. It also shows how to learn accurate and effective regular expression patterns of implicit navigation paths from automatically created training sets using aggregated results from weak page type classifiers. These type classifiers can be trained and applied to large set of unseen forums. It produces the best effectiveness and addresses the scalability issue and includes the concept called sentimental analysis. This paper tells about the web crawler and their challenges and I produced survey of four papers.

Keywords: Web crawlers, Architecture, Challenges, Uses

1. INTRODUCTION:

A *web crawler* (also known as a *robot* or a *spider*) is a system for the bulk downloading of web pages. Web crawlers are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries. A related use is web archiving where large sets of web pages are periodically collected and archived for posterity. A third use is web data mining, where web pages are analyzed for statistical properties, or where data analytics is performed on them. Finally, web monitoring services allow their clients to submit standing queries, or *triggers*, and they continuously crawl the web and notify clients of pages that match those queries. The web crawlers lies in the fact that the web is not a centrally managed repository of information, but rather consists of hundreds of millions of independent web content providers, each one providing their own services, and many competing with one another. In other words, the web can be viewed as a federated information repository, held together by a set of agreed-upon protocols and data formats, such as the Transmission Control Protocol (TCP), the Domain Name Service (DNS), the Hypertext Transfer Protocol (HTTP), the Hypertext Markup Language (HTML) and the robots exclusion protocol. Crawlers can also be used for specific type of information and then checking links or validating HTML code.

Web Forum Structure:

A web forum [1] is a tree like or hierarchical structure. A forum can be divided into categories for the relevant conversations and then the posted messages. Under the categories are sub-forums and these sub-forums can be further divided into more sub-forums.

User groups:

A user of the forum can automatically be access to a more privileged user group based on conditions set by the administrator. An anonymous user of the site is commonly known as visitors. Visitors are to granted access to all

functions that do not require breach privacy. A guest can usually view the contents and then the posted messages of the forum.

Moderators

The moderators are called visitors or users of the forum who are granted access to the posted messages and threads of all members for the purpose of moderating conversations and also keeping the forum clean Moderators also answer users' concerns about the forum, general questions, as well as take action to specific complaints in the conversations.

Posts:

A post is a user will hold conversations to submit a message enclosed into a block containing the user's details and the date and time it was submitted. Posts have a limit usually measured in the characters. To have a message of minimum length of 10 characters. There is always an upper limit most boards have it at either 10,000, or 20,000 characters.

Thread

A thread is a collection of posts, conversations of the messages usually displayed from oldest to latest, A thread can contain any number of posts, including multiple posts from the same members, even if they are one after the other.

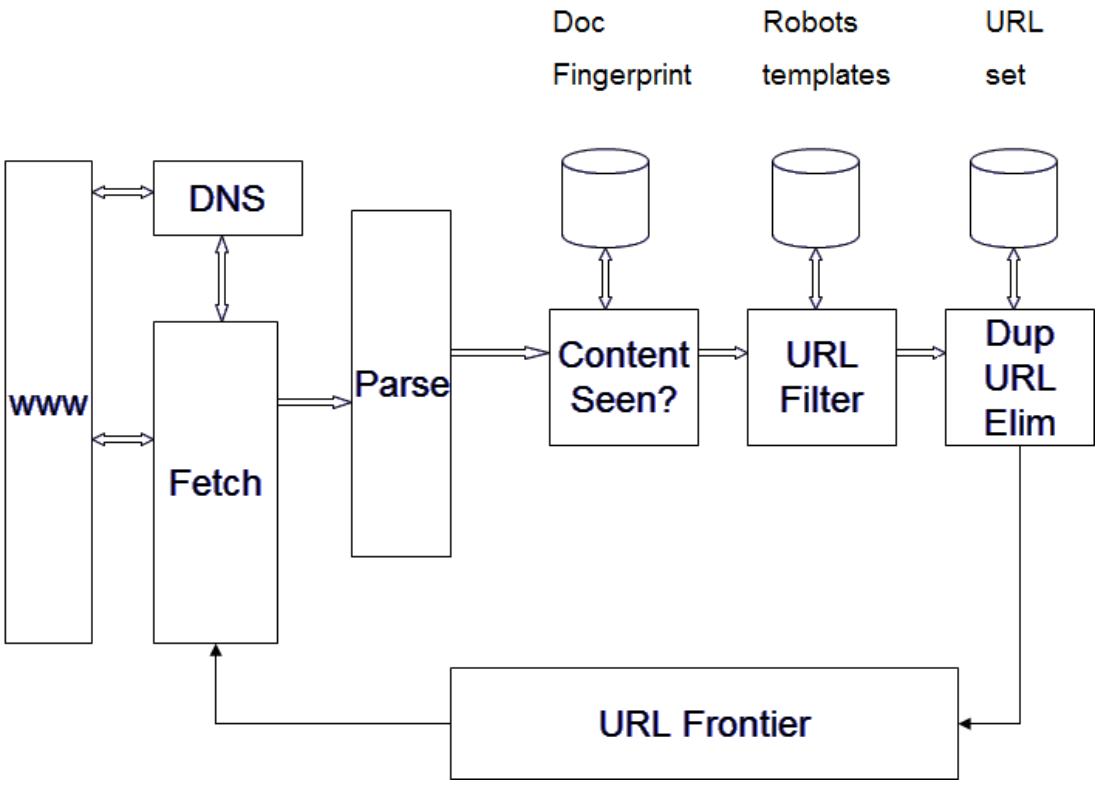

Figure 1: Web Crawler Architecture

2. CHALLENGES OF WEB CRAWLING:

The basic web crawling algorithm is simple: Given a set of seed Uniform Resource Locators (URLs), a crawler downloads all the web pages addressed by the URLs, extracts the hyperlinks contained in the pages, and iteratively downloads the web pages addressed by these hyperlinks.
Despite the apparent simplicity of this basic algorithm, web crawling has many inherent challenges:

Scale: The web is very large and continually evolving. Crawlers that seek broad coverage and good freshness must achieve extremely high throughput, which poses many difficult engineering problems. Modern search engine companies employ thousands of computers and dozens of high-speed network links.

Content selection tradeoffs: Even the highest-throughput crawlers do not purport to crawl the whole web, or keep up with all the changes. Instead, crawling is performed selectively and in a carefully controlled order. The goals are to acquire high-value content quickly, ensure eventual coverage of all reasonable content, and bypass low-quality, irrelevant, redundant, and malicious content. The crawler must balance competing objectives such as coverage and freshness, while obeying constraints such as per-site rate limitations. A balance must also be struck between exploration of potentially useful content, and exploitation of content already known to be useful.

Social obligations: Crawlers should be "good citizens" of the web, i.e., not impose too much of a burden on the web sites they crawl. In fact, without the right safety mechanisms a high-throughput crawler can inadvertently carry out a denial-of-service attack.

Adversaries: Some content providers seek to inject useless or misleading content into the orpus assembled by the crawler. Such behavior is often motivated by financial incentives, for example (mis)directing traffic to commercial web sites.

USES OF WEB CRAWLING

Web Archiving
It is a service where large sets of web pages are collected periodically and archived for posterity, provided by the Internet archive.

Web Data Mining
In Web data mining web pages are analyzed for some statistical properties or where different data analytics is performed on them. Attributor, a company that monitors the web for copyright and trademark infringements can be the example.

Web Monitoring Services
The web monitoring services allows their clients to trigger or submit standing queries, and they crawl the web continuously and notify the clients of pages that match those submitted queries.

3. LITERATURE SURVEY:

De-Duping URLs via Rewrite Rules:

In the paper "De-duping URLs via Rewrite Rules" is stated that a large fraction of the URLs on the web contain duplicate (or near-duplicate) content. De-duping URLs is an extremely important problem for search engines, since all the principal functions of a search engine, including crawling, indexing, ranking, and presentation, are adversely impacted by the presence of duplicate URLs.

Traditionally, the de-duping problem has been addressed by fetching and examining the content of the URL; their approach here is different. Given a set of URLs partitioned into equivalence classes based on the content (URLs in the same equivalence class have similar content), we address the problem of mining this set and learning URL rewrite rules that transform all URLs of an equivalence class to the same canonical form.

These rewrite rules can then be applied to eliminate duplicates among URLs that are encountered for the first time during crawling, even without fetching their content. In order to express such transformation rules, they proposed a simple framework that is general enough to capture the most common URL rewrite patterns occurring on the web; in particular, it encapsulates the DUST (Different URLs with similar text) framework [15].

They provide an efficient algorithm for mining and learning URL rewrite rules and show that under mild assumptions, it is complete, i.e., their algorithm learns every URL rewrite rule that is correct, for an appropriate notion of correctness. They demonstrate the expressiveness of their framework and the effectiveness of their algorithm by performing a variety of extensive large-scale experiments.Several previous studies have established that a large fraction of the web consists of duplicate URLs — syntactically distinct URLs having similar content. These duplicate URLs adversely affect the performance of commercial search engines in various ways.

In crawling, they waste valuable bandwidth, affect refresh times, and impact politeness constraints; in indexing, they consume unnecessary disk space; in link-based ranking, they impart disproportionate authority to undeserving URLs; in presentation, they pollute displayed search results and lead to a poor user experience. De-duping URLs is thus an extremely important problem in end-to-end web search, and enormous resources are invested by search engines for this task. The traditional approach to de-duping has been to fetch the content of the URL and then apply standard fingerprinting methods on the content to eliminate duplicates. However, it is desirable to identify duplicate URLs as early in the workflow as possible, ideally even prior to crawling.

Duplicate URLs occur on the web due to a multitude of reasons beyond blatant plagiarism. These include hosting the same set of URLs on different mirrors that are typically done for load balancing and fault tolerance, e.g., http://www-1.ibm.com and http://www-2.ibm.com. Often these are simple web-server based canonicalizations of URLs, e.g., dropping index.html from the website name, or other simple syntactic modifications such as removing the trailing slash, interchanging upper and lower cases etc.

Dynamic scripts frequently encode session-specific identifying information in the URL that is used to track the user and the session but has no impact on the content of the page. The presence of such content-neutral parts in a URL is an important reason for the proliferation of duplicates. Furthermore, even the order of dynamic parameters is mostly inconsequential with respect to the content of a URL, e.g., the URLs http://domain/show.php?a=10&b=20 and http://domain/show.php?b=20&a=10 are the same. Unlike plagiarized content, these are structured transformations on the URL string that mostly happen due to server software. With a proper understanding of these transformations, it is possible to detect whether two URLs have similar content even without explicitly examining their content.

Suppose we have a large collection of URLs along with their duplicate information, i.e., for every URL, which other URLs are duplicates (or near-duplicates) of this URL. Is it then possible to mine this collection and learn whether two URLs are duplicates of one another by examining only the URL strings? Equivalently, can we learn a set of rewrite rules that, given any two duplicate URLs, canonicalizes them to the same URL? If these rules are indeed learnable from an offline computation, they can be deployed in conjunction with the crawler to de-dup URLs prior to crawling them, thus ensuring that duplicate URLs are not even crawled! This is a significant departure from traditional de-duping approaches [21] that require the content of URLs in order to identify duplicates.

Considering that duplicate URLs constitute a large portion of the web, learning URL rewrite rules and deploying them in the crawler can tremendously improve the efficiency of not only the crawler but also subsequent steps in the processing pipeline. Given that URLs appear and disappear at a rapid rate, the value of mining and learning URL rewrite rules, especially in an offline manner, is not immediately apparent. Note however that the rewrite rules are

typically specific to a particular web server, and more specifically, to the software used by the web servers. As a result, these rules are likely to be more stable and have a longer life than the actual URLs themselves

Given a URL, they represent it as a function from a set of keys to a set of values. The set of key and values for the URL is defined as follows: first they split the URL into the static part comprising of the protocol, hostname, and the static path components, and the dynamic parts composed of the parameters and their values. They assumed that the set of separator tokens is known and fixed apriori; for the static part, they use the separator "/" and for the dynamic part, they use the tokens "?", "&," and ";" for identifying key-value pairs, and the token "=" for separating a key from a value. They represent the static portion with the static keys {k1, . . . , k`} corresponding to the ` components in the static part. For the dynamic part, each parameter in the URL is defined as a key.

In this paper, they present a different approach to the URL de-duping problem based on automatically generating URL rewrite rules by mining a given collection of URLs with content-similarity information. These rewrite rules can then be applied to eliminate duplicates among URLs that are en countered for the first time during crawling, even without fetching their content.

This has the huge advantage of trapping duplicates much earlier in a search-engine workflow, improving the efficiency of entire processing. Our framework is simple and has provable guarantees, and is shown to be effective in a large-scale experiment.

In their formulation, they used a fixed set of delimiters; it will be useful to study the effect of a more flexible tokenization on their algorithm. For instance, can our method be used to detect site mirrors, by segmenting the hostname using '.' as the delimiter? Extending their formalization to capture a wider set of rules, while still being efficiently learnable is also an interesting research direction.

Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms

In the paper "Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms" the author stated that Broder et al.'s [7] shingling algorithm and Charikar's [6] random projection based approach are considered \state-of-the-art" algorithms for finding near-duplicate web pages. Both algorithms were either developed at or used by popular web search engines. They compare the two algorithms on a very large scale, namely on a set of 1.6B distinct web pages.

The results show that neither of the algorithms works well for finding near-duplicate pairs on the same site, while both achieve high precision for near-duplicate pairs on different sites. Since Charikar's algorithm finds more near-duplicate pairs on different sites, it achieves a better precision overall, namely 0.50 versus 0.38 for Broder et al. 's algorithm. They present a combined algorithm which achieves precision 0.79 with 79% of the recall of the other algorithms.

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. Thus, algorithms for detecting these pages are needed. A naive solution is to compare all pairs to documents. Since this is prohibitively expensive on large datasets, Man ber [8] and Heintze [9] proposed first algorithms for detecting near-duplicate documents with a reduced number of comparisons.

Recently Hoad and Zobel [10] developed and compared methods for identifying versioned and plagiarised documents. Both Broder et al. 's and Charikar's algorithm have elegant theoretical justifications, but neither has been experimentally evaluated and it is not known which algorithm performs better in practice. In this paper we evaluate both algorithms on a very large real-world data set, namely on 1.6B distinct web pages.

They chose these two algorithms as both were developed at or used by successful web search engines and are considered \state-of-the-art" in finding nearduplicate web pages. They call them Algorithm B and C. They set all

parameters in Alg. B as suggested in the literature. Then the y chose the parameters in Alg. C so that it uses the same amount of space per document and returns about the same number of correct near-duplicate pairs, i.e., has about the same recall. They compared the algorithms according to three criteria: (1) precision on a random subset, (2) the distribution of the number of term differences per near-duplicate pair, and (3) the distribution of the number of near-duplicates per page.

For both algorithms every HTML page is converted into a token sequence as follows: All HTML markup in the page is replaced by white space or, in case of formatting instructtions, ignored. Then every maximal alphanumeric sequence is considered a term and is hashed using Rabin's fingerprinting scheme [11] to generate tokens, with two exceptions: (1) Every URL contained in the text of the page is broken at slashes and dots, and is treated like a sequence of individual terms. (2) In order to distinguish pages with different images the URL in an IMG-tag is considered to be a term in the page.

More specifically, if the URL points to a different host, the whole URL is considered to be a term. If it points to the host of the page itself, only the filename of the URL is used as term. Thus if a page and its images on the same host are mirrored on a different host, the URLs of the IMG-tags generate the same tokens in the original and mirrored version. Both algorithms generate a bit string from the token sequence of a page and use it to determine the near-duplicates for the page.

They performed an evaluation of two near-duplicate algorithms on 1.6B web pages. Neither performed well on pages from the same site, but a combined algorithm did without sacrificing much recall. Two changes might improve the performance of Alg. B and deserve further study: (1) A weighting of shingles by frequency and (2) using a different number k of tokens in a shingle.

However, recall that 28% of the incorrect pairs are caused by pairs of pages in two databases on the web. In these pairs the difference is formed by one consecutive sequence of tokens. Thus, reducing k would actually increase the chances that pairs of pages in these databases are incorrectly identified as near-duplicates. Note that Alg. C also could work with much less space. It would be interesting to study how this affects its performance. Additionally it would be interesting to explore whether applying Alg. C to sequences of tokens, i.e., shingles, instead of individual tokens would increase its performance.

As the results show both algorithms perform poorly on pairs from the same site, mostly due to boilerplate text. Using a boilerplate detection algorithm would probably help. Another approach would be to use a different, potentially slower algorithm for pairs on the same site and apply (one of) the presented algorithms to pairs on different sites.

Learning URL Patterns for Webpage De-duplication

In the paper “Learning URL Patterns for Webpage De-duplication” [3] the authors Hema Swetha Koppula, Krishna P. Leela and Amit Agarwal stated that Presence of duplicate documents in the World Wide Web adversely affects crawling, indexing and relevance, which are the core building blocks of web search. In this paper, they present a set of techniques to mine rules from URLs and utilize these rules for de-duplication using just URL strings without fetching the content explicitly.

Their technique is composed of mining the crawl logs and utilizing clusters of similar pages to extract transformation rules, which are used to normalize URLs belonging to each cluster. Preserving each mined rule for de-duplication is not efficient due to the large number of such rules. They present a machine learning technique to generalize the set of rules, which reduces the resource footprint to be usable at web-scale. The rule extraction techniques are robust against web-site specific URL conventions.

They compare the precision and scalability of our approach with recent efforts in using URLs for de-duplication. Experimental results demonstrate that their approach achieves 2 times more reduction in duplicates with

only half the rules compared to the most recent previous approach. Scalability of the framework is demonstrated by performing a large scale evaluation on a set of 3 Billion URLs, implemented using the MapReduce framework.

Their contributions in the paper are four fold:

1. They extend the representation of URL and Rule presented in [27]. The extensions result in better utilization of the information encoded in the URLs to generate precise Rules with higher coverage.

2. They propose a technique for extracting host specific delimiters and tokens from URLs. We extend the pairwise Rule generation to perform source and target URL selection. They also introduce a machine learning based generalization technique for better precision of Rules. Collectively, these techniques form a robust solution to the de-duplication problem.

3. Since scale is a necessary dimension on the Web, they present MapReduce [4] adaptation of the proposed techniques.

4. They demonstrate via experimental comparison that the proposed techniques produce 2 times more reduction in duplicates with half the number of Rules compared to [27]. Finally, via large scale experimental evaluation on a 3-billion URL corpus, we show that the techniques are robust and scalable.

They concluded that they presented a set of scalable and robust techniques for de-duplication of URLs. Their techniques scale to web due to feasible computational complexity and easy adaptability to the MapReduce paradigm. They presented basic and deep tokenization of URLs to extract all possible tokens from URLs which are mined by our Rule generation techniques for generating normalization Rules.

They presented a novel Rule generation technique which uses efficient ranking methodologies for reducing the number of pair-wise Rules. Pair-wise Rules thus generated are consumed by the decision tree algorithm to generate highly precise generalized Rules. They evaluated the effectiveness of their techniques on two data sets. They compared their results with previous approaches and showed that their approach significantly outperforms them on many key metrics.

They also evaluated their techniques on multi billion URL corpus and showed that they achieve not only high reduction ratios but also high average reduction per Rule. Their system is very practical as it is configurable for different precision levels and coverage thresholds to achieve high reduction ratios. Source selection approach which they have presented prioritizes head and torso traffic and they would like to explore the feasibility of Rule generation for the rest of tail traffic.

Dup clusters which are the ground truth for generating Rules includes false positives due to the approximate similarity measures. They would like to explore ways of handling these in a robust fashion. Generalization is performed separately for source and target and they would like to explore the feasibility of generalizing both in an iterative fashion.

Extracting and Ranking Product Features in Opinion Documents

In the paper “Extracting and Ranking Product Features in Opinion Documents” the authors stated that An important task of opinion mining is to extract people’s opinions on features of an entity. For example, the sentence, “I love the GPS function of Motorola Droid” expresses a positive opinion on the “GPS function” of the Motorola phone. “GPS function” is the feature. The paper focuses on mining features. Double propagation is a state-of-the-art technique for solving the problem. It works well for medium-size corpora.

However, for large and small corpora, it can result in low precision and low re-call. To deal with these two problems, two improvements based on part-whole and "no" patterns are introduced to increase the recall. Then feature ranking is applied to the extracted feature candidates to improve the precision of the top-ranked candidates.

They rank feature candidates by feature importance which is determined by two factors: feature relevance and feature frequency. The problem is formulated as a bipartite graph and the well-known web page ranking algorithm HITS is used to find important features and rank them high. Experiments on diverse real-life datasets show promising results.

Their proposed method deals with the problems of double propagation. So let us give a short explanation why double propagation can cause problems in large or small corpora. Double propagation assumes that features are nouns/noun phrases and opinion words are adjectives. It is shown that opinion words are usually associated with features in some ways. Thus, opinion words can be recognized by identified features, and features can be identified by known opinion words.

The extracted opinion words and features are utilized to identify new opinion words and new features, which are used again to extract more opinion words and features. This propagation or bootstrapping process ends when no more opinion words or features can be found. The biggest advantage of the method is that it requires no additional resources except an initial seed opinion lexicon.

Thus it is domain independent and unsupervised, avoiding laborious and time-consuming work of labeling data for supervised learning methods. It works well for medium–size corpora. But for large corpora, this method may extract many nouns/noun phrases which are not features. The precision of the method thus drops. The reason is that during propagation, adjectives which are not opinionated will be extracted as opinion words, e.g., "entire" and "current".

These adjectives are not opinion words but they can modify many kinds of nouns/noun phrases, thus leading to extracting wrong features. Iteratively, more and more noises may be introduced during the process. The other problem is that for certain domains, some important features do not have opinion words modifying them. For example, in reviewsof mattresses, a reviewer may say "There is a valley on my mattress", which implies a negative opinion because "valley" is undesirable for a mattress.

Obviously, "valley" is a feature, but "valley" may not be described by any opinion adjective, especially for a small corpus. Double propagation is not applicable in this situation. To deal with the problem, they propose a novel method to mine features, which consists of two steps: feature extraction and feature ranking. For feature extraction, they still adopt the double propagation idea to populate feature candidates. But two improvements based on part-whole relation patterns and a "no" pattern are made to find features which double propagation cannot find. They can solve part of the recall problem.

For feature ranking, they rank feature candidates by feature importance. A part-whole pattern indicates one object is part of another object. For the previous example "There is a valley on my mattress", they can find that it contains a part-whole relation between "valley" and "mattress". "valley" belongs to "mattress", which is indicated by the preposition "on". Note that "valley" is not actually a part of mattress, but an effect on the mattress. It is called a pseudo part-whole relation.

For simplicity, they will not distinguish it from an actual part-whole relation because for our feature mining task, they have little difference. In this case, "noun1 on noun2" is a good indicative pattern which implies noun1 is part of noun2. So if we know "mattress" is a class concept, we can infer that "valley" is a feature for "mattress".

There are many phrase or sentence patterns representing this type of semantic relation which was studied in ([8] Girju et al, 2006). Beside part-whole patterns, "no" pattern is another important and specific feature indicator in opinion documents.

They concluded that Feature extraction for entities is an important task for opinion mining. The paper proposed a new method to deal with the problems of the state-of-the-art double propagation method for feature extraction. It first uses part-whole and "no" patterns to increase recall. It then ranks the extracted feature candidates by feature importance, which is determined by two factors: feature relevance and feature frequency.

The Web page ranking algorithm HITS was applying to compute feature relevance. Experimental results using diverse real-life datasets show promising results. In the future work, apart from improving the current methods, they also plan to study the problem of extracting features that are verbs or verb phrases.

4. CONCLUSION:

There are different data mining classification techniques can be used to rank the extracted feature. This paper is to propose a new method to deal with the problem of double propogation method for feature extraction. The web page algorithm applied to compute feature relevance. Experimental results using diverse real life datasets show promising results. In future work planned to study the problem of extracting features that are verbs or verb phrases.

REFERENCES

[1] A. Dasgupta, R. Kumar, and A. Sasturkar, "De-Duping URLs via Rewrite Rules,"Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,pp. 186-194, 2008.
[2] M. Henzinger, "Finding Near-Duplicate Web Pages: A LargeScale Evaluation of Algorithms,"Proc. 29th Ann. Int'l ACM SIGIR. Conf. Research and Development in Information Retrieval,pp. 284-291, 2006.
[3] H.S. Koppula, K.P. Leela, A. Agarwal, K.P. Chitrapura, S. Garg, and A. Sasturkar, "Learning URL Patterns for Webpage DeDuplication,"Proc. Third ACM Conf. Web Search and Data Mining, pp. 381-390, 2010.
[4] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, pages 10–10, December 2004.
[5] L. Zhang, B. Liu, S.H. Lim, and E. O'Brien-Strain, "Extracting and Ranking Product Features in Opinion Documents," Proc. 23rd Int'l Conf. Computational Linguistics, pp. 1462-1470, 2010.
[6] M. S. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In 34th Annual ACM Symposium on Theory of Computing (May 2002).
[7] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic Clustering of the Web. In 6th International World Wide Web Conference (Apr. 1997), 393-404.
[8] U. Manber. Finding similar files in a large file system. In Proc. of the USENIX Winter 1994 Technical Conference (Jan. 1994).
[9] N. Heintze. Scalable Document Fingerprinting. In Proc. of the 2nd USENIX Workshop on Electronic Commerce (Nov 1996).
[10] T.C. Hoad and J. Zobel. Methods for identifying versioned and plagiarised documents. Journal of the American Society for Information Science and Technology 54(3):203-215, 2003.
[11] M. Rabin. Fingerprinting by random polynomials. Report TR-15-81, Center for Research in Computing Technology , Harvard University , 1981.

