International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

DUST Removal Framework Based on Improved Multiple Sequence Alignment Technique

Pulagam Sai Nandana[1], K N Brahmaji Rao[2]

M. Tech Student, Department of CS&SE, Andhra University

Guide, Department of CS&SE, Andhra University

Abstract: ***A large number of URLs collected by web crawlers correspond to pages with duplicate or near-duplicate contents. These duplicate URLs, generically known as DUST (Different URLs with Similar Text), adversely impact search engines since crawling, storing and using such data imply waste of resources, the building of low quality rankings and poor user experiences. To deal with this problem, several studies have been proposed to detect and remove duplicate documents without fetching their contents. To accomplish this, the proposed methods learn normalization rules to transform all duplicate URLs into the same canonical form. This information can be used by crawlers to avoid fetching DUST. A challenging aspect of this strategy is to efficiently derive the minimum set of rules that achieve larger reductions with the smallest false positive rate. As most methods are based on pairwise analysis, the quality of the rules is affected by the criterion used to select the examples and the availability of representative examples in the training sets. To avoid processing large numbers of URLs, they employ techniques such as random sampling or by looking for DUST only within sites, preventing the generation of rules involving multiple DNS names. As a consequence of these issues, current methods are very susceptible to noise and, in many cases, derive rules that are very specific. In this thesis, we present a new approach to derive quality rules that take advantage of a multi-sequence alignment strategy. We demonstrate that a full multi-sequence alignment of URLs with duplicated content, before the generation of the rules, can lead to the deployment of very effective rules. Experimental results demonstrate that our approach achieved larger reductions in the number of duplicate URLs than our best baseline in two different web collections, in spite of being much faster. We also present a distributed version of our method, using the MapReduce framework, and demonstrate its scalability by evaluating it using a set of 7.37 million URLs.***

Keywords: Search engines, Crawling, De-duplication, URL Normalization, Rewrite rules.

1. Introduction

Syntactically different URLs that have similar content is a common phenomenon on the Web. Besides plagiarism, these duplicate URLs, generically known as DUST (Duplicate URLs with Similar Text [3]), occur for many reasons. For instance, in order to facilitate the user's navigation, many web sites define links or redirections as alternative paths to reach a document. In addition, webmasters usually mirror content to balance load and ensure fault tolerance. Other common reasons for the occurrence of duplicate content are the use of parameters placed in distinct positions in the URLs and the use of parameters that have no impact on the page content, such as the session_id attribute, used to identify a user connection. Detecting DUST is an extremely important task for search engines since crawling this redundant content leads to several drawbacks such as waste of resources (bandwidth and disk storage, for example); disturbance in results of link analysis algorithms; and poor user experience due to duplicate results. To overcome these problems, several authors have proposed methods for detecting and removing DUST from search engines. Whereas first efforts focused on comparing document content, more recent studies propose strategies that inspect only the URLs without fetching the corresponding page content [1],[3], [6], [9], [11]. These methods, known as URL-based de-duping, mine crawl logs and use clusters of URLs referring to (near) duplicate content1 to learn normalization rules that transform duplicate URLs into a unified canonical form. This information can be then used by a web crawler to avoid fetching DUST, including ones that are found for the first time during the crawling. The main challenge for these methods is to derive general rules with a reasonable cost from the available training sets.

Thus, in this paper, we show that a full multi-sequence alignment of duplicate URLs, performed before rules are generated, can make the learning process more robust and less susceptible to noise when compared to previous work in the literature.

Objectives

The general objective of the research described in this work is to propose a method for web-scale DUST detection to obtain a small and general set of normalization rules when compared with state-of-the-art methods. This objective translates into the following specific objectives:

- Development of a DUST detection method based on multiple sequence alignment.
- DUST detection problem to induce rules faster than (i) a method based on traditional multiple sequence alignment and (ii) other state-of-the-art DUST detection approaches.
- Development of a parallel version of the training algorithm which takes advantage of the cluster of computers normally used in the environments where large scale crawlings are performed.

2. Literature Survey

Our focus in this paper is on efficient and large-scale de duplication of documents on the WWW [1]. Web pages which have the same content but are referenced by different URLs, are known to cause a host of problems[1]. Crawler resources are wasted in fetching duplicate pages, indexing

requires larger storage and relevance of results are diluted for a query [1].

The dust problem is that: The web is abundant with dust, different URLs with Similar Text[2]. For example, the URLs http://google.com/news and http://news.google.-com return similar content[2]. A single web server often has multiple DNS names, and any can be typed in the URL[2]. Many are artifacts of a particular web server implementation. For example, URLs of dynamically generated pages often include parameters; which parameters impact the page"s content is up to the software that generates the pages[2]. Some sites use their own conventions; for example, a forum site we studied allows accessing story number "num" both via the URL http://domain/story?id=num and via http://domain/story num. Our study of the CNN web site has discovered that URLs of the form http://cnn.com/money/whatever get redirected to http://money.cnn.com/whatever [2].

Universal rules, such as adding http:// or removing a trailing slash are used, in order to obtain some level of canonization [2]. By knowing dust rules, one can dramatically reduce the overhead of this process. But how can one learn about site-specific dust rules? Detecting dust from a URL lis[2]t. Most of our work therefore focuses on substring substitution rules, which are similar to the "replace" function in many editors[2].The results which comes after the crawling the corresponding pages, contains the duplicate or nearduplicate contents[6]. Hence these duplicate URLs commonly known as DUST (Duplicate URLs with Similar Text)[6]. It can be effectively explained using following example, the URLs http://google.com/news and http://news.google.com return the same content[6]. That means while searching for the news using any of above URL result the same content[6]. This DUST can be created for many reasons[6]. For instance, to facilitate theuser's navigation, many web site developers define links or redirection as alternative path to find or search a document [6].

The main challenge for these methods is to derive general rules from available training sets. Some methods make use of derive rules from pairs of duplicate URLs[6]. The quality of rules is affected by the criterion used to select these pairs[6]. Current methods are less affected by noise and derive rules that are very specific[6]. Thus, an ideal method should learn general rules from few training examples, taking maximum advantage, without sacrificing the detection of DUST across different sites[6]. The DUSTER is introduced, motived by these issues. DUSTER takes advantages of multiple sequence alignment (MSA) in order to obtain a smaller and more general set of normalization rules[6]. Traditionally the multiple sequence alignment is used in molecular biology as a tool[6]. The tool, find out the similar pattern in sequences[6]. By applying these we are able to identify similarities and differences among strings[6]. As the methods find patterns involving all the available strings, the method is able to find more general rules avoiding problems related to pairwise rule generation and problem related to finding rules across sites [6].

3. Proposed Methodology

In this section, we review the problem of sequence alignment, show how to apply sequence alignment to URLs and discuss why to apply URL alignment in URL de-duplication.

Sequence Alignment
A sequence alignment is a way of arranging n sequences in order to identify similar regions between them. The alignment process aims at inserting spaces into the sequences so that similar symbols (based on some criteria) are aligned in the same position. In our specific case, the alignment of similar tokens facilitates the process of inferring the rules to transform DUST into a canonical form.

Pairwise Sequence Alignment:
The alignment of two sequences, called pair wise sequence alignment, is the basic step for aligning an arbitrary number of sequences. This problem can typically be solved using dynamic programming to calculate all the sub problems involved in the process [13].

We formally define this concept as given in Definition 5.

Given the sequences X and Y with m and n characters respectively, the alignment process can be described by using a matrix S of size (m+1)×(n+1) so that S cells are filled as follows:

where sf(Xi,Yj) is a scoring function that defines a similarity between the pairs of symbols (Xi,Yj). This function gives points for matching tokens and penalties for any gap. When the value of cell Si,j is computed, a pointer from Si,j is set to the cell (a) Si,j−1 if Si,j = Si,j−1; (b) Si−1,j if Si,j = Si−1,j; or (c) Si−1,j−1 if Si,j = Si−1,j−1 + sf(Xi,Yj). Figure 3 presents the scoring/traceback matrix resulting from the alignment of the URL strings "www.IRS.gov/foia/index.html" and www.irs.ustreas.gov/foia.

Multiple sequence alignment
Given k > 2 sequences S = {S1,S2,...,Sk}, a Multiple Sequence Alignment of S can be considered a natural generalization of the pairwise alignment problem. Spaces are inserted at arbitrary positions in any of the k sequences to be aligned, so that the resulting sequences have the same size `. The sequences can be arranged in k lines and ` columns, such that element or gap of each sequence occurs in a single column.

As the Multiple Sequence Alignment problem is know to be NP-hard, several approaches have been developed to find a heuristic solution for it. In this work, in particular, we adopted a method know as Progressive Alignment [7] to align clusters of duplicate URLs (dupclusters with a size greater than two). In general lines, the method first performs the alignment between two previously selected sequences. Then a new sequence is chosen and aligned with the first alignment obtained or another pair of sequences is selected and aligned. This process is repeated until all sequences have been aligned, giving rise to the final multiple alignment.

The progressive alignment method uses a greedy policy in which once a space is inserted, it can not be removed for any subsequent alignment. Thus, all spaces are preserved until the final solution. The error rate introduced by the progressive alignment at each step tends to decrease if the most similar sequences are chosen, and increase if the most divergent sequences are chosen. Thus, determining the best order for the alignments is crucial. Ideally, the most similar sequences are aligned first, leaving the most divergent ones until the end, in order to reduce the error introduced by this heuristic solution.

URL Alignment

In order to obtain a smaller and more general set of normalization rules, our method takes advantage of multiple sequence alignment. The strategy is to create the so called consensus sequence for each dup-cluster in the training set and extract the rules from them. We perform this task by aligning the URLs in each cluster and then generating the consensus sequences as a result of this alignment. In the following subsections, we show how to align two or more URLs and how to generate a consensus sequence for these dup-clusters. Before presenting our URL alignment approach, we first show how we represent URLs.

URL Tokenization

Unlike previous works that treat URLs as strings generated according to W3C grammar4, we adopt a simpler representation. We consider a URL as a sequence of three types of tokens (URL tokens), as described by the EBNFbased5 grammar G described below:

(URL) ::= (tokeni) { (token) }
(token) ::= (alphabetic) | (number) | (punctuation)
(alphabetic) ::= (alpha) { (alpha) }
(alpha) ::= 'a'..'z' | 'A'..'Z'
(number) ::= (digit) { (digit) }
(digit) ::= '0'..'9'
(punctuation) = All remaining characters such as '/', ':', and '.'

Each URL to be aligned is initially parsed according to grammar G. This process, referred to as tokenization, decomposes the URL into a sequence of URL tokens. To facilitate URL alignment, each token extracted from a URL is represented as a singleton set. For example, URL u = http://ex.com/1.htm is represented by the following sequence of 11 token sets:

S = h{http},{:},{/},{/},{ex},{.},{com},{/},{1},{.},{htm}i

Pair-wise URL Alignment

The output of our alignment process is a sequence of sets, referred to as the consensus sequence, which is a way of representing the result of the alignment. The consensus sequence of n sequences is composed by the union of the tokens in the corresponding positions of the n aligned sequences. To help readers better understand the complete process, we illustrate it with an example. To obtain a consensus sequence for two URLs u1 = http://www.ex/ and u2 = http://www.un/home, we first obtain the token set sequences X and Y, associated with u1 and u2 respectively, with m and n tokens. X and Y are given by:

X = ({http},{:},{/},{/},{www},{.},{ex},{/})
Y = ({http},{:},{/},{/},{www},{.},{un},{/},{home})

Sequences X and Y are then aligned by inserting gaps, either into or at the ends of them. To determine where gaps should be inserted, matrix S in Equation 1 has to be calculated. To accomplish this, a score function SF is defined to measure the distance between the URL token sets. The scoring function we adopt, given by Equation 2, is the Jaccard similarity coefficient [15] which is commonly used to measure the overlap between two sets. For two sets, it is denoted as the cardinality of their intersection divided by the cardinality of their union.

$$sf(X_i, Y_j) = \begin{cases} \frac{|x_i \cap y_j|}{|x_i \cap y_j|} & if \exists (x_i, y_j) \in X_i \times Y_j \left| T(x_2) = T(y_j) \right. \\ -1 & otherwise \end{cases} \quad (2)$$

where τ : T →{a,n,p}is a function which maps a token set to its token type, T is the token space and{a,n,p}are the token types (a for alphabetic, n for numeric, and p for punctuation). Suppose we have two token sets Xi = {default, index, start} and Yj = {default, index}. The union between them is Xi ∪Yj = {default,index, start} and the intersection Xi ∩Yj = {default, index}. Jaccard similarity coefficient can be computed based on the number of elements in the intersection set divided by the number of elements in the union set:

sf(Xi,Yj) = |Xi∩Yj| |Xi∪Yj| = 2 3 = 0.66

At the end of the alignment, X and Y are transformed into sequences X' and Y' given by:

X' = h{http},{:},{/},{/},{www},{.},{ex},{/},{λ}i
Y' = h{http},{:},{/},{/},{www},{.},{un},{/},{home}i

where λ indicates a gap. X' and Y' have the same length so that every token is either a unique token or a gap in the other sequence. The final consensus sequenceC12 for URLs u1 and u2 is given by uniting the token sets of X' and Y':

4. Results

In this section we present the results obtained. In particular, we compare the methods according to the number of rules they detected, the number of valid rules they selected, and their performance in DUST detection. We also study the results of applying the rules to better understand the methods that derive them.

Evaluation Metrics and Methodology

To evaluate the effectiveness of our method and the baselines, we adopted some metrics used in [9] which estimate the quality of the normalization rules generated. The metrics used in our experiments were:

- Reduction Ratio: this metric measures the reduction ratio of the number of URLs after the removal of duplicates. It is defined as jUorigj□jUnormj jUorigj , where Uorig is the original URL set and Unorm is the normalized URL set; Avg Reduction PerRule: average UR reduction achieved per rule. It is defined as jUorigj_Compression jRj where R is the set of rules;
- Cluster-Reduction Rate: this metric measures the reduction ratio of the number of clusters after the normalization process. It is defined as jCorigj□jCnormjjCorigj ,where Corig is the number of clusters before normalization and Cnorm is the number of clusters after normalization. In the experiments, we randomly divided the duplicate clusters into three sets:

10% of the clusters were retained as a training set, 10% as a validation set, and the remaining 80%, as a test set. We used the training set to generate the rules, the validation set to filter them, and the test set to evaluate them. We adopted this strategy for our method and all the baselines because it better represents a real application where only a small fraction of DUST is provided as training data6.

Parameter settings

These were the parameters used in our experiments: K = 10, minfreq = 10, Cardset = 5 and minsupp = 10.

Table 4.3: Number of candidates and valid rules generated by different methods in GOV2 and WBR10 (fprmax = 0).

Data Set	Method	£Candidates	£Valid	Rate
GOV2	$R_{fanout\ -10}$	7,0942	2,242	31.60%
	R_{tree}	2,458	718	29.21%
	DUSTER	1,685	1,332	79.05%
WBR10	$R_{fanout\ -10}$	31,565	1,985	6.29%
	R_{tree}	6,974	1,575	22.58%
	DUSTER	786	577	85.48%

Candidate Rules vs. Valid Rules

In Table 3 we present the total number of rules learned by the three methods after the training (candidate rules) and the number of the rules ready to be used in the test (valid rules). Note that a small number of valid rules is desirable since the crawler should have a small footprint.

For Rfanout-10 and DUSTER, the valid rules consist of the rules that were not discarded in the validation. The rules considered invalid are automatically removed from the 6. In experiments with larger training sets, all the methods improved their absolute performance because much more test cases were observed in the training. Due to space constraints, we do not include results for such larger training sets but it is worthy note that, when using a 50% training size, the relative performance of the methods was similar to the observed with a 10% training size.

DUST Detection

Table 5 shows a comparison between DUSTER and the baseline methods regarding the task of DUST detection for the GOV2 and WBR10 datasets. These tables show, for each fprmax level and method, the number of applied/valid rules, along with its respective reduction ratio achieved, i.e., the reduction in the amount of URL scrawled, obtained by applying these rules. The performance of DUSTER was far superior when compared to the baselines at all fprmax levels experimented. We consider fprmax = 0 level the most important one, since it includes rules that did not fail in any of the test URLs in the validation set. At this level, DUSTER was able to reduce the amount of URLs crawled in 20.73% in GOV2, while the best baseline (Rfanout10) achieved only 11.39%. In WBR10, DUSTER was able to reduce 22.87%, while the best baseline (Rfanout10) achieved only 9.50%. These results show that DUSTER obtained a gain in the process of identifying duplicate URLs of 82% in GOV2 and 140.74% in WBR10, by applying almost two times less rules than Rfanout10.

Thus, besides achieving a higher compression rate, the rules generated by DUSTER are more effective than the ones generated by Rfanout10.

We also note that Rtree presented the worst performance among the methods we implemented. Such a weak performance was due to (a) the fact that it was designed to conduct normalization within websites, it is unable to find rules involving multiple domains and (b) it needs more training examples than we were able to provide in our collections.

In general, DUSTER was quite effective and is a viable alternative for solving the DUST detection problem. When considering other false-positive levels experimented, again DUSTER was able to outperform the baselines. For instance, when considering a fprmax _ 20% on GOV2 dataset, DUSTER reduced the number of crawled URLs in 30.71% of the original set of URLs, almost one third more than the best baseline, that reduced only11.94%. In WBR10, for fprmax _ 20%, DUSTER reduced 29.75% of URLs, while the best baseline Rtree reduced only 9.67%.

5. Conclusion and Future Work

In this work, we presented DUSTER, a new method to address the DUST problem, that is, the detection of distinct URLs that correspond to pages with duplicate or near-duplicate content. DUSTER learns normalization rules that are very precise in converting distinct URLs which refer the same content to a common canonical form, making it easy to detect them. To achieve this DUSTER applies a novel strategy based on a full multi sequence alignment of training URLs with duplicate content. By analyzing the alignments obtained, accurate and general normalization rules can be generated, as demonstrated in our experiments. We evaluated the method in a set of duplicate URLs extracted from the TREC GOV2 collection and found a reduction in the number of duplicate URLs that is 82% larger than the one achieved by our best baseline. When evaluating a Brazilian web sample, we obtained a gain of 140.74% over the same baseline.

As future work, we intend to improve the scalability and precision of our method, as well as to evaluate it using other datasets. For its scalability, we intend to provide a comprehensive comparison among strategies to cope with very large dup-clusters, including (a) to better understand the impact of using split dup-clusters instead of the original ones, (b) to propose distributed algorithms for the task and (c) to use more efficient multiple sequence alignment algorithms. In particular, regarding this last item, we intend to use algorithms recently proposed for gene alignment such as the one presented in [4] which is able to align n gene sequences in time proportional to O(n log n).

References

[1] A. Agarwal, H. S. Koppula, K. P. Leela, K. P. Chitrapura, S. Garg, P. K. GM, C. Haty, A. Roy, and A. Sasturkar. Url normalization for de-duplication of web pages. In Proceedings of the 18th ACM conference on

Information and knowledge management, CIKM '09, pages 1987–1990, New York, NY, USA, 2009. ACM.
[2] B. S. Alsulami, M. F. Abulkhair, and F. E. Eassa. Near duplicate document detection survey. the Proceedings of International Journal of Computer Science and Communications Networks, 2(2):147–151,2012.
[3] Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the dust: Different urls with similar text. ACM Trans. Web, 3(1):3:1– 3:31, jan 2009.
[4] G. Blackshields, F. Sievers, W. Shi, A. Wilm, and D. G. Higgins. Sequence embedding for fast construction of guide trees for multiple sequence alignment. Algorithms Mol Biol, 5:21, 2010.
[5] C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the trec 2004 terabyte track. In E. M. Voorhees and L. P. Buckland, editors, TREC, volume Special Publication 500-261. National Institute of Standards and Technology (NIST), 2004.
[6] A. Dasgupta, R. Kumar, and A. Sasturkar. De-duping urls via rewrite rules. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, pages 186–194, New York, NY, USA, 2008. ACM.
[7] D. F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. Journal of molecular evolution, 25(4):351–360, 1987.
[8] K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Research, 30(14):3059–3066, 2002.
[9] H. S. Koppula, K. P. Leela, A. Agarwal, K. P. Chitrapura, S. Garg, and A. Sasturkar. Learning url patterns for webpage deduplication.In Proceedings of the third ACM international conference on Web search and data mining, WSDM '10, pages 381–390, New York, NY, USA, 2010. ACM.
[10] J. P. Kumar and P. Govindarajulu. Duplicate and near duplicate documents detection: A review. European Journal of Scientific Research, 32:514–527, 2009.
[11] T. Lei, R. Cai, J.-M. Yang, Y. Ke, X. Fan, and L. Zhang. A pattern tree-based approach to learning url normalization rules. In Proceedings of the 19th international conference on World wide web,WWW '10, pages 611–620, New York, NY, USA, 2010. ACM.
[12] X. Mao, X. Liu, N. Di, X. Li, and H. Yan. Sizespotsigs: an effective deduplicate algorithm considering the size of page content. In Proceedings of the 15th Pacific-Asia conference on Advances in knowledge discovery and data mining - Volume Part I, PAKDD'11, pages 537–548, Berlin, Heidelberg, 2011. Springer-Verlag.
[13] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48(3):443–453, 1970.
[14] K. W. L. Rodrigues, M. Cristo, E. S. de Moura, and A. S. da Silva. Learning url normalization rules using multiple alignment of sequences. In O. Kurland, M. Lewenstein, and E. Porat, editors, SPIRE, volume 8214 of Lecture Notes in Computer Science, pages 197–205. Springer, 2013.
[15] M.Theobald, J. Siddharth, and A. Paepcke. Spotsigs: robust and efficient near duplicate detection in large web collections. In In SIGIR08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pages563–570. ACM, 2008.