URL Normalization for De-duplication of Web Pages

Amit Agarwal * Hema Swetha Koppula Krishna P. Leela

Krishna Prasad Chitrapura Sachin Garg Pavan Kumar GM

Yahoo! Labs
Bangalore, India
{amitja,shema,krishna,pkrishna,gsachin,pavank}@yahoo-inc.com

ABSTRACT

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, we present a set of techniques to mine rules from URLs and utilize these learnt rules for de-duplication using just URL strings without fetching the content explicitly. Our technique is composed of mining the crawl logs and utilizing clusters of similar pages to extract specific rules from URLs belonging to each cluster. Preserving each mined rules for de-duplication is not efficient due to the large number of specific rules. We 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. We demonstrate the effectiveness of our techniques through experimental evaluation.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process; I.7.0 [**Document and Text Processing**]: General

General Terms

Algorithms, Performance

Keywords

Search engines, URL de-duplication, Page importance, Decision tree

1. INTRODUCTION

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

Duplicate URLs, referencing the same page are present due to many reasons. These include session-ids or cookie information being stored in the URLs, e.g., `ytsession` in `http://www.youtube.com/index?&ytsession=YTAdOJpcz` and `http://www.youtube.com/index?&ytsession=72syq1gfF`. URLs are made search engine friendly by making both static and dynamic URL available, e.g., `http://en.wikipedia.org/wiki/Casino_Royale` and `http://en.wikipedia.org/?title=Casino_Royale`. Also there are a large number of URLs with irrelevant or superfluous components contained within them, e.g. `Lord-Rings` in `http://www.amazon.com/Lord-Rings/dp/B000634DCW` instead of URL `http://www.amazon.com/dp/B000634DCW`. Sometimes webmasters construct URLs with custom delimiters making detection of duplicates difficult, e.g., `http://catalog.ebay.com/The-Grudge_UPC_043396062603_W0QQ_fclsZ1QQ_pcatidZ1QQ_pidZ43973351QQ_tabZ2` and `http://catalog.ebay.com/The-Grudge_UPC_043396062603_W0?_fcls=1&_pcatid=1&_pid=43973351&_tab=2`.

An estimate by [6] shows that approximately 29 percent of web-pages are duplicates and the magnitude is increasing. Clearly, this prompts for an efficient solution that can perform de-duplication without fetching the content of the page. As duplicate URLs have specific patterns which can be utilized for de-duplication of web-pages, in this paper we will focus on the problem of de-duplication using just URLs without fetching the content.

1.1 Related Work

Conventional methods to identify duplicate documents involved fingerprinting each document's content and group documents by defining a similarity on the fingerprints. Many elegant and effective techniques using fingerprint based similarity for de-duplication have been devised [4, 8, 10]. [8, 10] also emphasized and showed results with large scale experiments. However, with the effectiveness also comes the cost of fingerprinting and clustering of documents. Recently, more cost-effective approach of using just the URLs information for de-duplication has been proposed, first by Bar-Yossef et.al. [2] and extended by Dasgupta et.al. [5].

Bar-Yossef et al. [2] call the problem, "DUST: Different

*The author's current affiliation is Infibeam, Bangalore.

CIKM'09, November 2–6, 2009, Hong Kong, China.

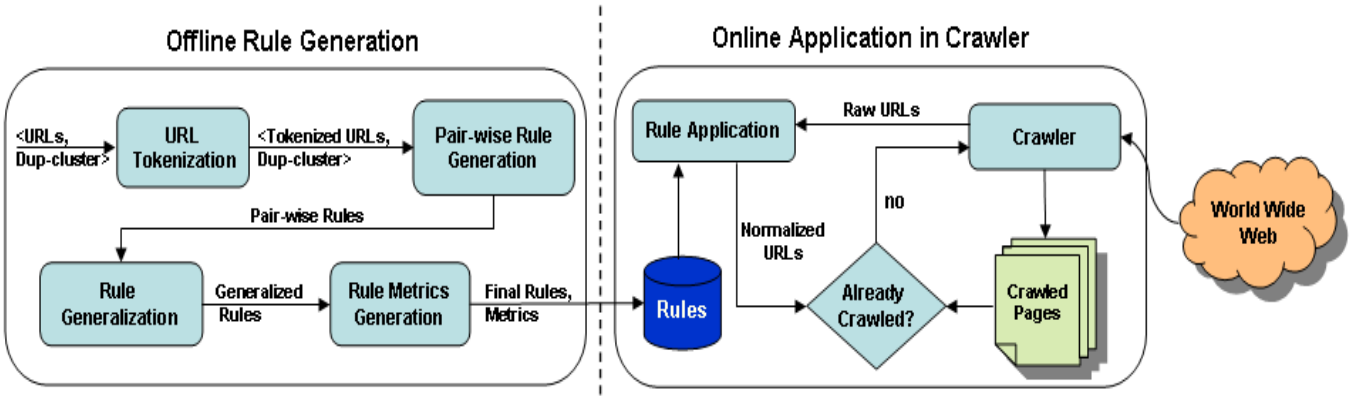

Figure 1: Flow diagram showing the Offline Rule Generation and Online Rule Application

URLs with Similar Text" and propose a technique to uncover URLs pointing to similar pages. The DUST algorithm focuses on discovering substring substitution rules, which are used to transform URLs of similar content to one canonical URL. Dasgupta et. al. [5] extended this formulation by considering a broader set of rule types, subsuming DUST rules. Different rule types which they consider are: DUST rules, session-id rules, irrelevant path components and complicate rewrites. We extend this work by proposing scalable techniques to learn high precision rules which can attain better de-duplication of URLs.

The rest of the paper is organized as follows. In Section 2 we formalize the problem and describe the representations of a URL and a Rule. Section 3 presents the algorithm with details of each technique. Experimental evaluation is presented in Section 4 and we conclude in Section 5.

2. PROBLEM DEFINITION

In this section, we present the problem definition using the URL and Rule representations from [5].

A URL is tokenized using standard delimiters and the primary components of the URL, namely protocol, hostname, path components and query-args [3] are extracted. A URL u can be represented as a function from $K \rightarrow V$ where K is composed of keys and V is composed of values from both static path components and query-args. While query-arg keys inherit the key name from the query name, the path component keys k_n are indexed with an integer n, where n is the position index from the start of the URL with protocol corresponding to n equals 1. For example, `http://en.wikipedia.org/wiki?title=Web_crawler` is represented as $\{k_1 = http,\ k_2 = en.wikipedia.org,\ k_3 = wiki$ and $k_{title} = Web\_crawler\}$.

A Rule is generated from a source, target URL pair and is composed of context and transformation. "Context" represents the source URLs, i.e., the URLs on which the Rule can be applied. "Transformation" represents the changes on the source URL to transform it to the target URL. The changes include change in the value of a key, addition of new keys and removal of keys. New value for a key, if already present in the source URL for a different key, is represented using a key reference. For example, the Rule with context $c(k_1) = http$, $c(k_2) = en.wikipedia.org$, $c(k_3) = wiki$, $c(k_{title}) = Web\_crawler$ and transformation $t(k_4) = k_{title}$, $t(k_{title}) = \perp$ is formed from source URL `http://en.wikipedia.org/wiki?title=Web_crawler` and target URL `http://en.wikipedia.org/wiki/Web_crawler`. Complete definitions of URL and Rule are supplied in Definition 1 and 2 respectively.

DEFINITION 1. (URL) A *URL* u is defined as a function $u : K \rightarrow V \cup \{\perp\}$ where K represents the set of all keys from the URL set and V represents the set of all values. A key not present in the URL is denoted by $\perp$.

DEFINITION 2. (RULE) A Rule r is defined as a function $r : C \rightarrow T$ where C represents the context and T represents the transformation of the URL. Context C is a function $C : K \rightarrow V \cup \{*\}$ and transformation T is a function $T : K \rightarrow V \cup \{\perp, K\}$, where * represents any value.

Given clusters of URLs with similar page content (such a cluster is referred to as a *duplicate cluster* or a *dup-cluster*), we learn Rules from URL strings which can identify duplicates. These learnt Rules can then be utilized for normalizing duplicate URLs into an unique normalized URL. Figure 1 demonstrates the various steps involved in offline Rule generation and the online application of the generated Rules by a crawler. The first step in offline processing is tokenizing the URLs into <key,value> tuples. These tuples along with the dup-cluster information are used for the Rule generation. Pair-wise Rules are generated from selected URL pairs with in a dup-cluster and then the pair-wise Rules are generalized, as described in Section 3.2. Generalization not only reduces the number of Rules but also give Rules which can efficiently normalize unseen URLs. Various performance metrics for the generalized Rules are computed. The metrics we used are described in Section 4. These metrics can be used for Rule selection based on the precision requirements. Applications such as crawlers, while crawling obtain a set of new URLs to crawl. These URLs are normalized using Rules generated from offline processing. Normalized URLs are compared with already crawled URLs to find duplicates. This process of de-duplication avoids the overhead of crawling duplicate documents. As crawlers and other real-time applications have resource constraints, offline processing has to generate a small set of Rules which can achieve maximum reduction in duplicates.

3. ALGORITHM

In Section 3.1, we discuss two techniques for extracting tokens from URLs: generic tokenization and host specific tokenization. Section 3.2 covers Rule generation algorithms: pair-wise Rule generation, which generates Rules specific to URL pairs and Rule generalization, which generalizes both context and transformation of pair-wise Rules.

3.1 URL Preprocessing

Tokenization is performed on URLs to generate a set of $< key, value >$ pairs as represented in Definition 1. This involves two stages of tokenization. In the first stage, Basic Tokenization, the URLs are tokenized by parsing them according to RFC 1738 [3]. The standard delimiters are used to extract the protocol, hostname, path components and query-args from the URL. The second stage, Deep Tokenization, further tokenizes these tokens using host-level delimiters learnt from the URLs of a host. We use an unsupervised technique to learn the custom encodings and extract syntactic features from URLs. Our technique is influenced by sequence based techniques of computational biology [7].

Starting with the generic pattern $*$, the Deep Tokenization algorithm recursively detects more specific patterns matching the given set of tokens. This results in generation of a pattern tree for a given set of tokens. For example, pattern cat-*.html matches all tokens matching the pattern regex. The algorithm refines this pattern into specific patterns, eg. cat-*-sku-*.html and cat-*-sort-*.html each of which matches a subset of the tokens. Once a leaf pattern in the tree is reached, it is used to extract deep tokens from corresponding tokens. For example the token cat-1205234-sku-B00006HW5W-item-ibm_128mb_thinkpad.html can be tokenized into deep tokens *cat*, −, 1205234, −, *sku*, −, $B00006HW5W$, −, *item*, −, $ibm\_128mb\_thinkpad$, . and *html* using the pattern cat-*-sku-*-item-*.html. The deep tokens thus obtained are also represented as <key,value> pairs similar to the rest of the tokens in the URL.

3.2 Rule Generation

The Rule generation stage involves generation of Rules for pairs of URLs within a dup-cluster and then generalizing these pair-wise Rules to form a smaller set of generalized Rules. The generalized Rules can accommodate new values, making them applicable to unseen URLs. The smaller memory footprint of the generalized Rules makes them usable on-the-fly by the crawlers.

3.2.1 *Pair-wise Rule Generation*

Pair-wise Rule Generation Algorithm generates pair-wise Rules for selected URL pairs within dup-clusters. A Rule is constructed from a source, target URL pair by setting the source URL as the Context of the Rule. All changes required to transform source URL to target URL are added to the Transformation of the Rule. At web scale generating the pair-wise Rules for all pairs of URLs within a dup-cluster is not feasible. The number of URL pairs in all the dup-clusters is $\sum_i C_i^2$ where C_i is the number of URLs contained in dup-cluster i. This number can run in billions due to large sized dup-clusters. These large sized dup-clusters are common on web due to presence of session-ids and irrelevant components in the URLs. Due to the scale problems of pair-wise Rule generation, we introduce an approach for selective sampling of source and target URLs.

Target Selection As target URLs are used for generating transformations, selecting better targets yield better transformations and compact Rules. Typically, dup-clusters have a small set of URLs which are close to the normalized URL of that dup-cluster. Some of the characteristics of an ideal normalized URL include static type of the URL, shorter length of URL, minimum hop distance from the domain root and high number of in-links. As these set of characteristics closely match with host-level page rank discussed in [9], we considered this approach for ranking URLs and selected the top-k as target URLs. This not only achieves significant reduction in number of Rules but also generates coherent Rules for better generalization.

Source Selection As source URLs are used for generating context, we select URLs with high importance. The URLs seen by the crawlers and search engines follow power law distribution where some URLs are seen more often than others. Learning Rules for these high traffic URLs will give high reduction in duplicates. URL importance can be obtained by URL ranking methods such as PageRank [11] or the On-Line Page Importance [1]. For our experiments we used the crawl time computed page importance metric [1], available from the crawl logs. We sampled ranked URLs from each duplicate cluster using stratified sampling. URLs are divided into equal sized buckets based on the page importance. URLs are sampled from each bucket proportional to the contribution of the bucket to the total importance of the dup-cluster.

As the number of target URLs selected per dup-cluster is constant and the number of source URLs selected is proportional to the size of the dup-cluster, the number of pair-wise Rules generated are linear in the number of URLs.

3.2.2 *Rule Generalization*

Rule generalization captures generic patterns out of the pair-wise Rules and generalizes both the contexts and transformations. Previous efforts used heuristics to perform generalization, however these heuristics do not guarantee precision of the Rules. We use a Decision Tree [12] for context generalization. Advantages of Decision Tree over heuristics are the proven error bounds and the robustness of the technique as it is used in multiple domains.

Context generalization involves constructing the decision tree with transformations as Classes/Targets. The key set K is considered as attributes and the value set $V \cup \{*\}$ is considered as instances for the attributes. We construct a decision tree where an attribute (or key k) is selected at every iteration. Nodes at the current iteration are assigned a particular value v of the selected attribute k. We assign $*$ (wildcard) to a node if there is no single value v which holds majority. After constructing the decision tree, we traverse it top-down to generate the generalized context and corresponding transformation.

While context generalization is done through decision tree, we also perform transformation generalization by considering all transformations corresponding to the same context. If a key is generalized to take wildcard $(*)$ in the context and the same key takes multiple values in the transformation, then the value is replaced by wildcard in the transformation. The motivation for doing this generalization is, if the generalized key in the context can take any value $(*)$, the key is irrelevant for any matched URL, and hence the transformation can also have a wildcard for that key.

4. EXPERIMENTAL EVALUATION

In this section we present experimental setup and key metrics used for measuring the performance of our techniques.

Metrics We used the following metrics for our experiments:

1. *Coverage* of a Rule is the number of URLs the Rule applies to, denoted by r_{cov}
2. *Precision* is also a Rule level metric. If r_{cov} is the coverage of Rule r and f is the number of URL pairs $(u, v) \ni r(u) = v$ and u and v are not in the same dup-cluster, precision of r is $\frac{r_{cov}-f}{r_{cov}} * 100$.
3. *ReductionRatio* is a metric for set of Rules. It is the percentage reduction in the number of URLs after transforming the URLs with a set of Rules. It is defined as $\frac{|U_{orig}|-|U_{norm}|}{|U_{orig}|}$ where U_{orig} is the original URL set and U_{norm} is the normalized URL set.

We considered the data set presented in [5], consisting of dup-clusters having size of at least 2. Data set consists of 7.87 million URLs from a set of 356 randomly selected hosts and it contains 1.83 million dup-clusters. We performed 50-50 test-train split by randomly assigning each dup-cluster to either training set or test set. We learned Rules on the train set and evaluated the Rules on the train+test set.

Results Pair-wise Rule generation produced 12.37 million Rules on the data set. The generalization, which includes both context and transformation generalization, reduced the Rule set to 93,230 Rules. During our evaluation, we observed that there are large number of generalized Rules which have very less coverage. So we have filtered Rules with coverage < 10. This gave us a final set of 38,830 Rules. At 100% precision, with 649 Rules we achieved 6.2% reduction compared to 3% reduction of [5].

Precision threshold	Num Rules	% of Rules	Reduction Ratio (%)
precision = 1	649	1.67	6.2
>= .95	1707	4.40	26.42
>= .9	2375	6.12	32.98
>= .80	3547	9.13	41.53
All	38830	100.00	68.65

Table 1: Metrics comparison with URL rewrite approach for small data set

In Table 1 we present reduction ratio and number of Rules for different levels of precision. It can be seen that our approach achieves good reduction ratios with a small number Rules. High reduction ratios with less number of Rules holds for all precision levels above 80%.

5. CONCLUSIONS

In this paper we presented a set of scalable and robust techniques for de-duplication of URLs. Our techniques are scalable to Web due to feasible computational complexity. We 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. We 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 decision tree algorithm to generate precise generalized Rules. We evaluated the effectiveness of our techniques on 7.78M URL data set. Our results show that we perform significantly well on key metrics.

6. ADDITIONAL AUTHORS

Additional authors: Chittaranjan Haty (Yahoo! Inc., Bangalore, India. email: `chitta@yahoo-inc.com`), Anirban Roy (Yahoo! Inc., Bangalore, India. email: `anirbanr@yahoo-inc.com`) and Amit Sasturkar (Yahoo! Inc., Sunnyvale, CA. email: `asasturk@yahoo-inc.com`).

7. REFERENCES

[1] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 280–290, May 2003.

[2] Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the dust: different urls with similar text. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 111–120, May 2007.

[3] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform resource locators (url), 1994.

[4] A. Broder. On the resemblance and containment of documents. In *SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997*, page 21, June 1997.

[5] A. Dasgupta, R. Kumar, and A. Sasturkar. De-duping urls via rewrite rules. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 186–194, August 2008.

[6] D. Fetterly, M. Manasse, and M. Najork. On the evolution of clusters of near-duplicate web pages. In *LA-WEB '03: Proceedings of the First Conference on Latin American Web Congress*, page 37, November 2003.

[7] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, New York, 1997.

[8] 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*, pages 284–291, August 2006.

[9] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[10] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 141–150, May 2007.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, November 1999.

[12] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986.