Do not Crawl in the DUST: Different URLs with Similar Text Extended Abstract

Uri Schonfeld
Dept. of Electrical Engineering
Technion, Haifa 32000, Israel
shuri@tx.technion.ac.il

Ziv Bar-Yossef*
Dept. of Electrical Engineering
Technion, Haifa 32000, Israel
zivby@ee.technion.ac.il

Idit Keidar
Dept. of Electrical Engineering
Technion, Haifa 32000, Israel
idish@ee.technion.ac.il

ABSTRACT

We consider the problem of DUST: Different URLs with Similar Text. Such duplicate URLs are prevalent in web sites, as web server software often uses aliases and redirections, translates URLs to some canonical form, and dynamically generates the same page from various different URL requests. We present a novel algorithm, *DustBuster*, for uncovering DUST; that is, for discovering rules for transforming a given URL to others that are likely to have similar content. DustBuster is able to detect DUST effectively from previous crawl logs or web server logs, *without* examining page contents. Verifying these rules via sampling requires fetching few actual web pages. Search engines can benefit from this information to increase the effectiveness of crawling, reduce indexing overhead as well as improve the quality of popularity statistics such as PageRank.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications—Data mining

General Terms: Algorithms, Performance.

Keywords: mining, rules, duplicates, similarity.

1. INTRODUCTION

The web is abundant with DUST, Different URLs with Similar Text. For example, the URLs `http://news.google.com` and `http://google.com/news` return similar content. Many web sites define links, redirections, or aliases, such as allowing the tilde symbol (`~`) to replace a string like `/people`, or `/users`. Some sites allow different conventions for file extensions– `.htm` and `.html`; others allow for multiple default index file names – `index.html` and `default.html`. A single web server often has multiple DNS names, and any can be typed in the URL. As the above examples illustrate, DUST is typically not random, but rather stems from some general rules, which we call DUST *rules*, such as "`~`" → "`/people`", or "`/default.html`" at the end of the URL can be omitted.

Moreover, DUST rules are typically not universal. 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. Some sites use their own conventions; for example, a forum site we studied allows accessing story number `num` on its site both via the URL `http://domain/story?id=num` and via `http://doamin/story_num`. In this paper, we focus on detecting DUST rules within a given web site. We are not aware of any previous work tackling this problem.

Knowledge about DUST can be very valuable for search engines: It can increase the effectiveness of web crawling by eliminating redundant accesses to the same page via multiple URLs. For example, in one crawl we examined the number of URLs fetched would have been reduced by 18%. It can also reduce indexing overhead. More generally, it allows for a *canonical* URL representation, where each page has a single canonical name. Such a representation is essential for any statistical study of web pages, e.g., computing their popularity. Canonization can also reduce caching overhead. We focus on URLs with *similar* contents rather than identical ones, since duplicate pages and different versions of the same document are not always identical; they tend to differ in insignificant ways, e.g., counters, dates, and advertisements. Likewise, some URL parameters impact only the way pages are displayed (fonts, image sizes, etc.) without altering their contents.

Although search engines normally employ different techniques to avoid and detect DUST, they do not discover site-specific DUST rules. Universal rules, such as adding `http://` or removing a trailing slash are used, in order to obtain some level of canonization. They find additional DUST by comparing page shingles, which are hash-based summaries of page content. However, this is conducted on a page by page basis and all the pages must be fetched to employ this technique. By knowing the DUST rules, one can dramatically reduce the overhead of this process. Additionally, our findings show that these rules remain valid for a long period of time and are valid for new pages as well as old ones. But how can search engines learn about site-specific DUST rules?

Contrary to initial intuition, we show that it is possible to discover likely DUST rules without fetching a single web page. We present an algorithm, *DustBuster*, which discovers such likely rules from a list of URLs with observed web page size. Such a *URL list* can be obtained from many sources including a previous crawl and the web server's logs. The rules are then verified (or refuted) by sampling a small number of actual web pages. At first glance, it is not clear that a URL list can provide reliable information regarding DUST, as it does not include actual page contents.

How can one deduce information about likely DUST rules from a URL list? We show two ways for doing so. The first detects *substring substitution* rules, whereby every occurrence of some string α in a URL can be replaced by an-

*Supported by the European Commission Marie Curie International Re-integration Grant.

WWW2006, May 22–26, 2006, Edinburgh, UK.

other string β. Our first observation is that if such a rule is common, we can expect to find in the URL list multiple examples of pages accessed both ways (although not all pages accessed one way will be accessed both ways). For example, in the site where `story?id=` can be replaced by `story_`, we are likely to see in the URL list many different pairs of URLs that differ only in this substring; we say that such a pair of URLs is an *instance* of `story?id=` and `story_`. The set of all instances of a rule is called the rule's *support*. Our first attempt to uncover DUST is therefore to seek rules that have large support.

Unfortunately, one also finds rules that have large support albeit they are not DUST rules. For example, when examining URL lists, we find numerous instances of the digits `1` and `2`, as in `pic-1.jpg` and `pic-2.jpg`, and `story_76541` and `story_76542`, none of which are DUST. We address the challenge of eliminating such false rules using two heuristics. The first employs the size field included in web URL lists. The difficulty with size-based filtering is that the size of a dynamic page can vary dramatically, e.g., when many users comment on an interesting story. To account for such variability, we compare the ranges of sizes seen in all accesses to each page. When the size ranges of two URLs do not overlap, they are unlikely to be DUST. Our second heuristic is based on the observation that false rules tend to flock together, e.g., in most instances of `1` and `2`, one could also replace the `1` by other digits. We therefore assign lower credibility to a URL that pertains to instances of many different rules. Together, these two heuristics are very effective at distinguishing likely DUST rules from false ones. Fig. 1 shows results from 4 logs of an academic site. We observe that, quite surprisingly, DustBuster's detection phase achieves a very high precision rate even though it does not fetch even a single page.

Having detected likely DUST rules, another challenge that needs to be addressed is eliminating redundant ones. For example, the rule "`http://site-name/story?id=`" → "`http://site-name/story_`" will be detected, along with many consisting of substrings thereof, e.g., "`?id=`" → "`_`". However, before performing validations, it is not obvious which rule should be kept in such situations– the latter could be either valid in all cases, or invalid outside the *context* of the former. Nevertheless, we are able to use support information from the URL list in order to remove many redundant likely DUST rules. We remove additional redundancies after performing some validations, and thus compile a succinct list of rules.

In addition to substring substitutions, a second type of DUST that is prevalent in web sites stems from parameters that do not influence the page content. Fortunately, parameters have a predictable syntactical structure, which makes them easy to discover. Each parameter helps define two rules, substituting a default value for this parameter and removing it completely. We use size comparisons in order to filter out parameters that do impact the page content, and leave only likely DUST parameter substitution rule.

Both types of DUST rules are validated using limited sample URLs. For each rule we find sample URLs to which the rule can be applied. The rule is applied to each URL resulting in URL pairs. A rule is valid for a URL pair if both associated pages exist and their content is similar. A rule is deemed valid if the fraction of valid URL pairs is greater than some threshold.

We experiment with DustBuster on two web sites with very different characteristics. We find that DustBuster can discover rules very effectively from moderate sized URL lists, with as little as 20,000 entries. The detection of likely rules entails sorting and a linear traversal of the URL list, hence, on a URL list of length L, its running time is $O(L \log(L))$. On a typical URL list with roughly 50,000 entries, producing likely rules takes around ten minutes on a low-end desktop PC. Limited sampling is then used in order to validate or refute each rule. Figure Fig. 2 shows that 20 validations per rule are enough to achieve 1.0 precision. This process is substantially cheaper than crawling and producing shingles for the entire site in order to detect DUST.

Finally, once the correct DUST rules are discovered, we exploit them for URL canonization. In the general case, the canonization problem is NP-complete. Nevertheless, we have devised an efficient *canonization algorithm* that *typically* succeeds in transforming URLs to a site-specific canonical form. Using the canonization algorithm, we can discover up to 68% of the redundant URLs in a URL list. A log of a complete crawl was canonized and an 18% reduction was observed.

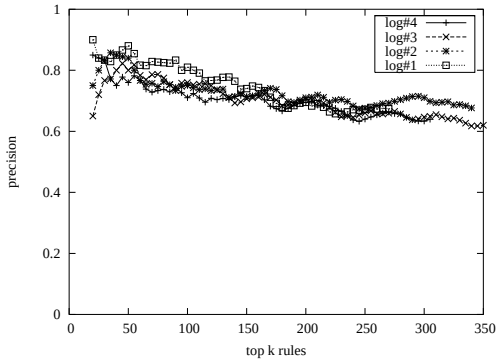

Figure 1: Precision achieved by likely rule detection (DustBuster's first phase) *without* fetching actual content, results from 4 different logs (Academic site).

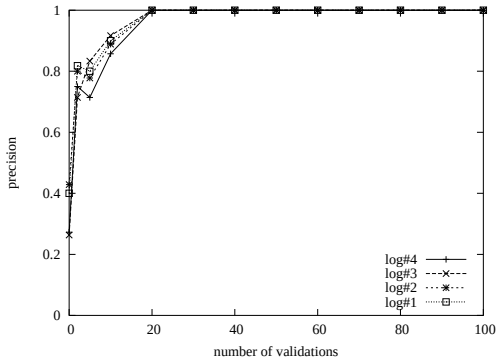

Figure 2: Precision among rules that DustBuster attempted to validate vs. number of validations used (N).(Forum site)

Acknowledgments

We thank Tal Cohen and the forum site team, and Greg Pendler and the `ee.technion.ac.il` admins for providing us with access to web URL lists and for technical assistance.