Do Not Crawl in the DUST: Different URLs with Similar Text

Ziv Bar-Yossef, Idit Keidar, Uri Schonfeld
WWW’07

CSE 450 Web Mining
Presented by Zaihan Yang
Introduction & Contribution

- Propose a novel algorithm DustBuster for uncovering DUST.
  - Discover DUST rules from a URL list
    - Mainly focus on the substring substitution rules
    - Introduce 3 heuristic methods.
    - Eliminate redundant rules.
    - Validate DUST rules.
  - Use DUST rules to transform URLs into canonical form

- Main feature: Mine DUST from crawl logs or web server logs instead of examining page content.
Problem Identification

- **What is DUST?** Different URLs with Similar Context.
  

- **How generated?**
  
  Aliases, redirection, dynamically generated pages, etc.

- **Features of DUST?**
  
  Not casual: with certain rules.

  Not universal: specific to web sites.

- **What advantage for uncovering them?**
  
  Reduce overhead in crawling, indexing, and catching.

  Increase accuracy of page metrics, like PageRank.
Problem Definition

- **URL**: strings over $\Sigma$ starts with “^” and ends with “$”.
- **DUST**: $(u_1, u_2)$ are DUST if $\text{doc}(u_1)$ and $\text{doc}(u_2)$ similar. *Shingling resemblance measure.*
- **URL List**: 
a URL, http return code, size of the document, document sketch.
- **DUST rules**: instance of rules, support of rules
  URL pair: $(u_1, u_2)$ Rule: $\alpha \rightarrow \beta$ if $u_1=p\alpha s$ and $u_2=p\beta s$.
  $u_1=^\text{http://www.site.com/index.html}$$
  $u_2=^\text{http://www.site.com}$
  DUST rule: “index.html$” $\rightarrow$$”$”
Large Support Heuristic

- Main idea: The support of a valid DUST rule is large.
- How to compute support size
  - **Envelope** of string $\alpha$
    - URL $u=p\alpha s$, $(p, s)$ is the envelop of $\alpha$
    - $u=\text{http://www.site.com/index.html}$ $\alpha=\text{“index”}$
      $p=\text{“^http://www.iste.com/”}$ $s=\text{“.html$”}$
    - $E_L(\alpha)$: a set of envelopes of $\alpha$ in URLs, each of which appear in $\xi$ and has $\alpha$ as a substring.
- Theorem:
  \[
  | \text{support}_L(\alpha \rightarrow \beta) | = | E_L(\alpha) \cap E_L(\beta) | \\
  | \text{support}_L(\alpha \rightarrow \beta) | = | \text{support}_L(\beta \rightarrow \alpha) | \\
  \]
  $\alpha \neq \beta$, $\alpha$ and $\beta$ are non-empty and non-semiperiodic.
Small Bucket Heuristic

- **Main idea:** Much of the support of valid DUST rules belong to small buckets.

- **Bucket:**
  - For an envelope \((p, s)\), a bucket \((p, s)\) is the set of all substrings \(\alpha\) satisfying that \(p\alpha s \in \mathcal{L}\).
  - Namely, if \((p, s)\) belongs to many envelope set
    - \(E_L(\alpha_1), E_L(\alpha_2), \ldots, E_L(\alpha_k)\), then \(\alpha_1, \alpha_2, \ldots, \alpha_k\) constitutes the bucket of \((p, s)\).
Similarity Likeliness Heuristic

- **Main Idea:** The likely similar support of a valid DUST rule is large.
- **Similar Content:**
- **Document sketch:**
  - Obtained from previous crawl.
  - Shingling resemblance.
- **Size match:**
  - Obtained from web server logs.
  - For each URL, a min and max size, size interval.
  - \((u_1, u_2)\) is similar if interval overlaps.
Algorithm Framework

- Input: URL list
- Detect likely DUST rules
- Eliminate redundant rules
- Validate DUST rules using samples:
  - Eliminate DUST rules that are “wrong”
  - Further eliminate duplicate DUST rules

{No Fetch Required}
Detecting likely DUST rules

- **Input**: a URL list \( L \).
- **Output**: an ordered list of pairs, each representing two DUST rules whose support beyond a threshold \( MS \).
- **ST**: Substring table
- **IT**: Instance table
- **RT**: Rule table

```plaintext
1: Function DetectLikelyRules(URLList \( \mathcal{L} \))
2: create table ST (substring, prefix, suffix, size_range/doc_sketch)
3: create table IT (substring1, substring2)
4: create table RT (substring1, substring2, support_size)
5: for each record \( r \in \mathcal{L} \) do
6:   for \( \ell = 0 \) to \( S \) do
7:     for each substring \( \alpha \) of \( r \) of length \( \ell \) do
8:       \( p := \) prefix of \( r \) preceding \( \alpha \)
9:       \( s := \) suffix of \( r \) succeeding \( \alpha \)
10:      add \((\alpha, p, s, r.size\_range/r.doc\_sketch)\) to ST
11:     group tuples in ST into buckets by (prefix, suffix)
12:   for each bucket \( B \) do
13:     if (\(|B| = 1 \) OR \(|B| > T\)) continue
14:     for each pair of distinct tuples \( t_1, t_2 \in B \) do
15:       if (LikelySimilar\( (t_1, t_2)\))
16:         add \((t_1.substring, t_2.substring)\) to IT
17:     group tuples in IT into rule\_supports by (substring1, substring2)
18:   for each rule\_support \( R \) do
19:     \( t := \) first tuple in \( R \)
20:     add tuple \((t.substring1, t.substring2, |R|)\) to RT
21:   sort RT by support\_size
22: return all rules in RT whose support size is \( \geq MS \)
```
Eliminating Redundant Rules

- (".co.il/story?id=", ".co.il/story_") & ("story?id=", "story_")

**Refinement of rule pairs**
- A rule A refines a rule B if support(A) ⊆ support(B)
- Rule $\alpha' \rightarrow \beta'$ refines rule $\alpha \rightarrow \beta$, if there is envelope $(\gamma, \delta)$ satisfying $\alpha' = \gamma \alpha \delta$ and $\beta' = \gamma \beta \delta$.

**Eliminate method**
- If A refines B and support(A) ⊂ support(B), then keep the broader one.
- If A refines B and $|\text{support}(A)| = |\text{support}(B)|$, then remove the refined one.

```
1: Function EliminateRedundancies(pairs_list \mathcal{R})
2: for i = 1 to |\mathcal{R}| do
3:   if (already eliminated \mathcal{R}[i]) continue
4:   for j = 1 to min(MW, |\mathcal{R}| - i) do
5:     if (\mathcal{R}[i].size - \mathcal{R}[i + j].size >
           \max(MRD \cdot \mathcal{R}[i].size, MAD)) break
6:     if (\mathcal{R}[i] refines \mathcal{R}[i + j])
7:       eliminate \mathcal{R}[i + j]
8:     else if (\mathcal{R}[i + j] refines \mathcal{R}[i]) then
9:       eliminate \mathcal{R}[i]
10:      break
11: return \mathcal{R}
```
Validating DUST rules

- Validate rules via fetching a small number of pages.

Figure 3: Validating a single likely rule.

```
1: Function ValidateRule(R, L)
2:   positive := 0
3:   negative := 0
4:   while (positive < (1 - ε)N AND negative < εN) do
5:     u := a random URL from L on which applying R results
6:       in a different URL
7:     v := outcome of application of R to u
8:     if (fetch u failed) continue
9:     if (fetch v failed OR DocSketch(u) \neq DocSketch(v))
10:    negative := negative + 1
11:  else
12:    positive := positive + 1
13:  if (negative \geq εN )
14:    return FALSE
15:  return TRUE
```

Figure 4: Validating likely rules.

- Validation count: N
- Refutation threshold ε
- α shrinks β if |α| \geq |β|
- Prefer to be considered.
URL canonization

- **Canonization problem**
  - A canonical URLs subset $C_u \subseteq U_s$
  - A canonization: Mapping $C: U_s \rightarrow C_u$

- **Algorithm**
  - Input: a URL $u$ and a list of valid DUST rule $R$.
  - Repeatedly apply $u$ to all the rules in $R$ until $u$ is unchanged, or
  - A predetermined max iteration is reached.
Experiment Results

- Dataset
  - 4 websites:
    
    | Web Site            | Log Size | Unique URLs |
    |---------------------|----------|-------------|
    | Forum Site          | 38,816   | 15,608      |
    | Academic Site       | 344,266  | 17,742      |
    | Large News Site     | 11,883   | 11,883      |
    | Small News Site     | 9,456    | 9,456       |

- Parameter Settings:

<table>
<thead>
<tr>
<th>Substring Length</th>
<th>Bucket Size</th>
<th>MRD</th>
<th>MAD</th>
<th>MW</th>
<th>MS</th>
<th>Validation Count</th>
<th>Refutation Threshold</th>
<th>Max Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>(6, 11)</td>
<td>5%</td>
<td>1</td>
<td>1100</td>
<td>3</td>
<td>100</td>
<td>5%-10%</td>
<td>10</td>
</tr>
</tbody>
</table>

- Metrics:
  - Precision
  - Discovered Redundancy
  - Coverage
Results

- Detecting and Eliminating

<table>
<thead>
<tr>
<th>Web Site</th>
<th>Rules Detected</th>
<th>Rules Remaining after 2nd Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forum Site</td>
<td>402</td>
<td>37 (9.2%)</td>
</tr>
<tr>
<td>Academic Site</td>
<td>26,899</td>
<td>2,041 (7.6%)</td>
</tr>
<tr>
<td>Large News Site</td>
<td>12,144</td>
<td>1,243 (9.76%)</td>
</tr>
<tr>
<td>Small News Site</td>
<td>4,220</td>
<td>96 (2.3%)</td>
</tr>
</tbody>
</table>

Table 2: Rule elimination in second phase.

(b) Academic site, impact of size matching.

(c) Large news site, impact of shingle matching, 4 shingles used.
Results

• Validation

(a) Forum, 4 different logs.

(b) Academic, 4 different logs.

<table>
<thead>
<tr>
<th>Web Site</th>
<th>Valid Rules Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forum Site</td>
<td>7</td>
</tr>
<tr>
<td>Academic Site</td>
<td>52</td>
</tr>
<tr>
<td>Large News Site</td>
<td>62</td>
</tr>
<tr>
<td>Small News Site</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 3: The number of rules found to be valid.

1 “.co.il/story_” → “.co.il/story?id=”
2 “&LastView=&Close=” → “”
3 “.php3?” → “?”
4 “.il/story_” → “.il/story.php3?id=”
5 “&NewOnly=1&tvqz=2” → “&NewOnly=1”
6 “.co.il/thread_” → “.co.il/thread?rep=”

Figure 7: The valid rules detected in the forum site.
Results

Coverage

Savings in crawl size

<table>
<thead>
<tr>
<th>Web Site</th>
<th>Reduction Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Academic Site</td>
<td>18%</td>
</tr>
<tr>
<td>Small News Site</td>
<td>26%</td>
</tr>
<tr>
<td>Large News Site</td>
<td>6%</td>
</tr>
<tr>
<td>Forum Site(using logs)</td>
<td>4.7%</td>
</tr>
</tbody>
</table>

Table 4: Reductions in crawl size.

Figure 8: DUST classification, academic.
Conclusions

- Introduced a problem of mining site-specific DUST rules.

- Proposed the DustBuster algorithm for mining DUST from a URL list.

- Mining DUST rules can
  - Reduce crawling overhead by up to 26%.
  - Reduce indexing overhead
  - Benefit canonizing URL names, and increase accuracy of page metrics.