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.

  E.g. *http://google.com/news* & *http://news.google.com*.

- *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 Σ starts with "^" and ends with "$".
- DUST: *(u1, u2)* are DUST if *doc(u1)* and *doc(u2)* 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: (u1, u2) Rule: $\alpha \rightarrow \beta$ if u1=pαs and u2=pβs.

u1=^http://www.site.com/index.html$

u2=^http://www.site.com$

DUST rule: "index.html$" →"$"

Large Support Heuristic

- Main idea: The support of a valid DUST rule is large.
- How to compute support size
  - **Envelope** of string α
    - URL u=pαs, (p, s) is the envelop of α
    - u=http://www.site.com/index.html α="index" p="^http://www.iste.com/" s=".html$"
    - **$E_L(\alpha)$:** a set of envelops of α in URLs, each of which appear in £ and has α as a substring.
  - Theorem:

$$|\text{support}_L(\alpha \rightarrow \beta)| = |E_L(\alpha) \cap E_L(\beta)| \qquad |\text{support}_L(\alpha \rightarrow \beta)| = |\text{support}_L(\beta \rightarrow \alpha)|$$

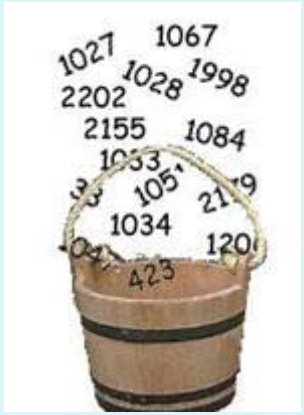
α≠β, α and β are non-empty and non-semiperiodic.

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 α satisfying that pαs∈£.
  - Namely, if (p, s) belongs to many envelope set
    - $E_L(\alpha 1)$, $E_L(\alpha 2)$, ……$E_L(\alpha k)$, then α1, α2, …… αk constitutes the bucket of (p, s).

^http://www.domain.name/story_



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.
  - (u1, u2) is similar if interval overlaps.

Algorithm Framework

- Input: URL list
- Detect likely DUST rules
- Eliminate redundant rules

No Fetch Required

- Validate DUST rules using samples:
  - Eliminate DUST rules that are “wrong”
  - Further eliminate duplicate DUST rules

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

```
1:Function DetectLikelyRules(URLList 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 ∈ ℒ do
6:    for ℓ = 0 to S do
7:      for each substring α of r.url of length ℓ do
8:        p := prefix of r.url preceding α
9:        s := suffix of r.url succeeding α
10:       add (α, 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 t1, t2 ∈ B do
15:     if (LikelySimilar(t1, t2))
16:       add (t1.substring, t2.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 ≥ 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 (γ, δ) 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 |support(A)|=|support(B)|, then remove the refined one.

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

Validating DUST rules

- Validate rules via fetching a small number of pages.

```
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
          in a different URL
6:    v := outcome of application of R to u
7:    fetch u and v
8:    if (fetch u failed) continue
9:    if (fetch v failed OR DocSketch(u) ≠ DocSketch(v))
10      negative := negative + 1
11:   else
12:     positive := positive + 1
13:   if (negative ≥ εN )
14:     return FALSE
15: return TRUE
```

Figure 3: Validating a single likely rule.

```
1:Function Validate(rules_list R, test_URLList L)
2   create an empty list of rules LR
3:  for i = 1 to |R| do
4:    for j = 1 to i - 1 do
5:      if (R[j] was not eliminated AND R[i] refines R[j])
6:        eliminate R[i] from the list
7:        break
8:    if (R[i] was eliminated)
9:      continue
10:   if  (ValidateRule(R[i].α → R[i].β, L))
11:     add R[i].α → R[i].β to LR
12:   else if  (ValidateRule(R[i].β → R[i].α, L))
13:     add R[i].β → R[i].α to LR
14:   else
15:     eliminate R[i] from the list
16: return LR
```

Figure 4: Validating likely rules.

- Validation count: N
- Refutation threshold ε
- α shrinks β if |α| >= |β|
- Prefer to be considered.

URL canonization

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

Substring Length	Bucket Size	MRD	MAD	MW	MS	Validation Count	Refutation Threshold	Max Iterations
35	(6, 11)	5%	1	1100	3	100	5%-10%	10

- Metrics:

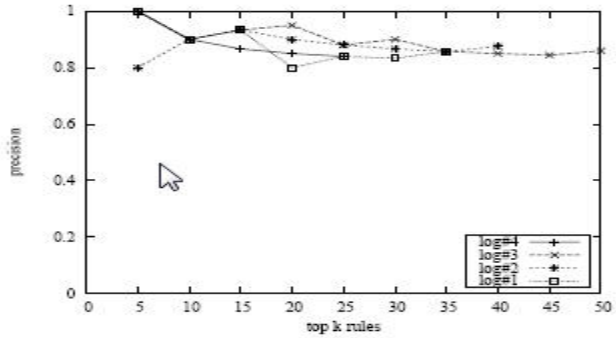
Precision Discovered Redundancy Coverage

Results

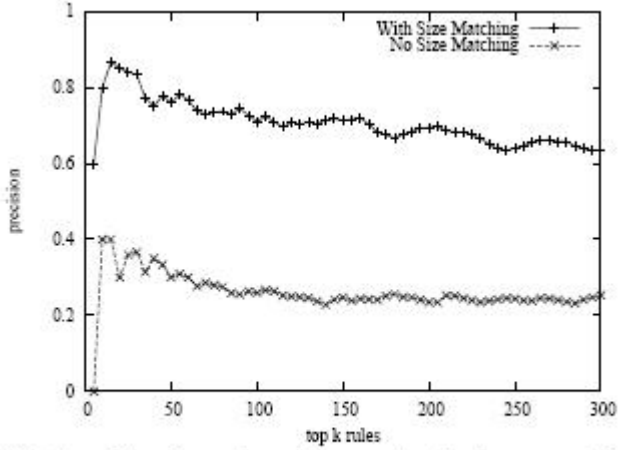
- Detecting and Eliminating

Web Site	Rules Detected	Rules Remaining after 2nd Phase
Forum Site	402	37 (9.2%)
Academic Site	26,899	2,041 (7.6%)
Large News Site	12,144	1,243 (9.76%)
Small News Site	4,220	96 (2.3%)

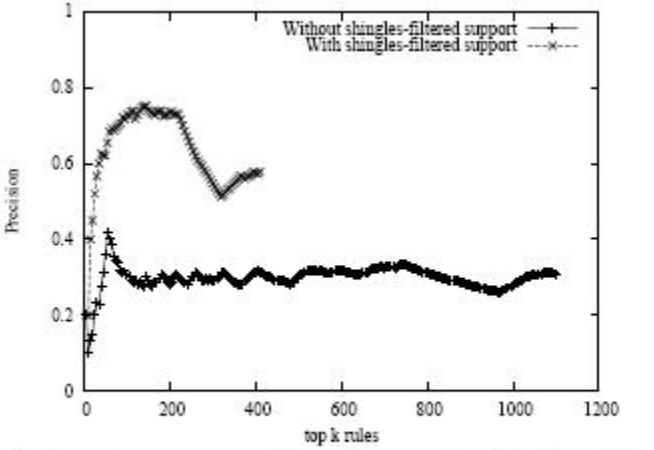
Table 2: Rule elimination in second phase.



(a) Forum, 4 different logs.



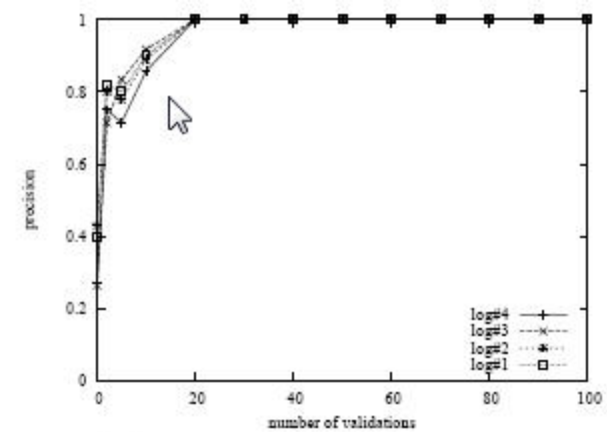
(b) Academic site, impact of size matching.



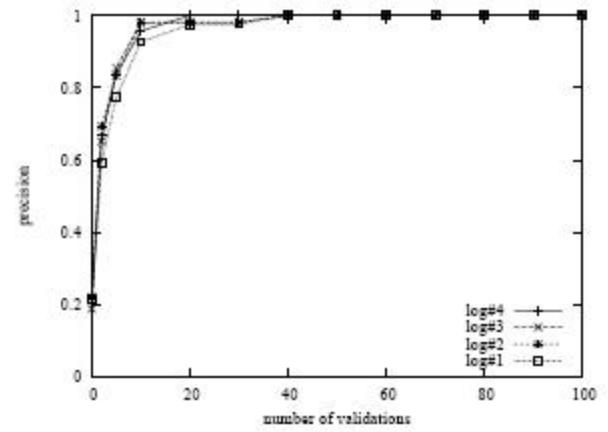
(c) Large news site, impact of shingle matching, 4 shingles used.

Results

- Validation



(a) Forum, 4 different logs.



(b) Academic, 4 different logs.

Web Site	Valid Rules Detected
Forum Site	7
Academic Site	52
Large News Site	62
Small News Site	5

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="
7  "http://www.../story_" → "http://www.../story?id="
```

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

Results

Coverage

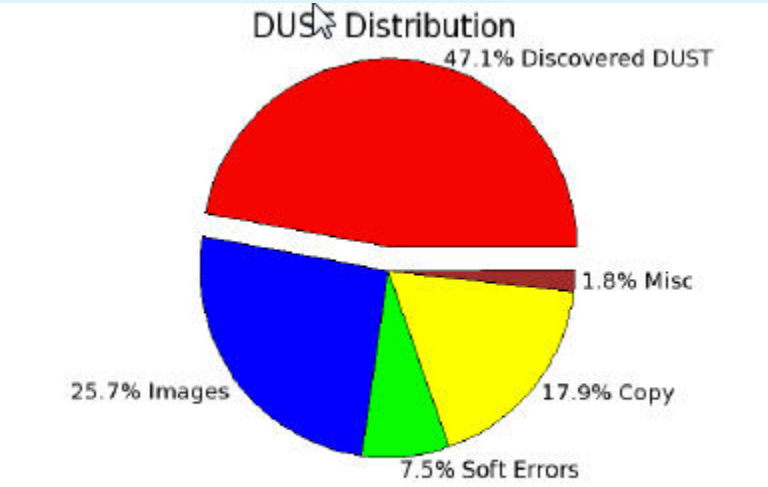


Figure 8: DUST **classification, academic.**

Savings in crawl size

Web Site	Reduction Achieved
Academic Site	18%
Small News Site	26%
Large News Site	6%
Forum Site(using logs)	4.7%

Table 4: Reductions in crawl size.

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.