Automatic Generation of Deep Web Wrappers based on Discovery of Repetition

Nakatoh, Tetsuya
Computing and Communications Center, Kyushu University

Yamada, Yasuhiro
Department of Informatics, Kyushu University

Hirokawa, Sachio
Computing and Communications Center, Kyushu University

出版情報：Proceeding of the First Asia Information Retrieval Symposium (AIRS2004). (AIRS2004), pp.269-272, 2004-10. AIRS2004 Secretariat
バージョン：
権利関係：

Automatic Generation of Deep Web Wrappers based on Discovery of Repetition

Tetsuya Nakatoh
Computing and Communications Center, Kyushu University
Hakozaki 6-10-1, Fukuoka, 812-8581, JAPAN
nakatoh@cc.kyushu-u.ac.jp

Yasuhiro Yamada
Department of Informatics, Kyushu University
Hakozaki 6-10-1, Fukuoka, 812-8581, JAPAN
yshiro@matu.cc.kyushu-u.ac.jp

Sachio Hirokawa
Computing and Communications Center, Kyushu University
Hakozaki 6-10-1, Fukuoka, 812-8581, JAPAN
hirokawa@cc.kyushu-u.ac.jp

ABSTRACT

A *Deep Web wrapper* is a program that extracts contents from search results. We propose a new automatic wrapper generation algorithm which discovers a repetitive pattern from search results. The repetitive pattern is expressed by token sequences which consist of HTML tags, plain texts and wild-cards. The algorithm applies *a string matching with mismatches* to unify the variation from the template and uses FFT(fast Fourier transformation) to attain efficiency. We show an empirical evaluation of the algorithm for 51 Web databases.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models, Search process*

General Terms

Algorithm, Experimentation

Keywords

Deep Web, Wrapper, Web Minig, Search Engine, Meta Search, String Matching with Mismatches, FFT

1. INTRODUCTION

A huge number of text data exist on the Web and they are growing exponentially. These Web pages have a lot of variety in types, purposes, and qualities and are written in many different languages. Moreover, they have no rigid structures as relational database systems do. These features are the main problems in aggregating data on the Web into databases. Therefore, information integration of heterogeneous sites is one of the significant research themes of Web mining.

Meta search engines [4, 8] are practical approach for information integration. They integrate search result pages generated from search sites. The key problem of information integration is to construct a program which extracts contents from Web pages. Such a program is referred to as a *Web wrapper*. It is a costly and mistakable work to code a wrapper manually, due to huge quantities of Web pages and a large variety of contents. Thus the semi-automatic and fully automatic construction of a Web wrapper is of special importance. This area of research has drawn a lot of attention recently.

Most of semi-automatic wrapper generation algorithms, *e.g.*, Muslea *et al.* [6], Kushmerick *et al.* [5] and Cohen *et al.* [2], use supervised machine learning techniques. However, we need many appropriate examples which should be prepared manually. Constructing such examples is as costly as to code a wrapper manually.

Our wrapper generation algorithm is based on the discovery of repetitive pattern. Therefore we do not need manually prepared examples. A similar idea was used to detect record-boundary in a HTML file in Embley *et al.* [3]. However they consider the number of characters between the tags and used standard deviation. Our method is more robust to the change in the contents. Chang *et al.* [1] have extracted the regularity of tags using Patricia tree. This method also does not need examples. However, all plain texts are regarded as the same token. Therefore, the tags determine the structure. In our method, the structure is determined by tags and repetitive plain texts. That helps distinguishing similar patterns of heterogeneous lists, and excludes the accidental coincidence of patterns.

Such an exhaustive search of repetitive sequence requires $O(n^2)$ time for the input text size n and is not efficient. We apply *a string matching algorithm with mismatches* [7] to discover repetitive sequence of tokens in the search results, thus the wrapper generation algorithm runs in $O(kn \log n)$ time where k is the number of samples in randomized algorithm.

We conducted an empirical evaluation of the algorithm by applying the method to 51 Web databases, which are available as a testbed for information extraction [9].

2. DISCOVERY OF REPETITION WITH MISMATCH

We generate the candidate patterns in three steps. In the first step, we translate an HTML file into a sequence of tokens of tags, plain texts and wild-cards. This process of transformation is described in Section 3. We consider these

tokens as symbols in the sequel of this section and use the word "pattern" to refer a repetitive sequence of tokens.

In the second step, we evaluate a likelihood of a length ℓ as a repetitive pattern. If the input string s contains successive occurrences of the same substring w of length ℓ, the k-th occurrence of w appears exactly in the distance of ℓ after $(k-1)$-th occurrence. As an evaluation measure of the length ℓ, we use the number c_ℓ of matches between $s = s_1, \ldots, s_{n-\ell}$ and $t = \mathrm{shift}(s, \ell)$, *i.e.*, $t = s_{\ell+1} s_{\ell+2} \ldots s_n$. It is defined as $c_\ell = \sum_{i=1}^{n-\ell} \delta(s_i, s_{i+\ell})$, where δ is the Kronecker function.

Naive computation of $c_1, \ldots, c_n$ requires $O(n^2)$. But we can obtain them in $O(kn \log n)$ applying FFT. This sequence $c_1, \ldots c_n$ is referred as score vector of the input string.

In the third step, we choose one pattern for each candidate length ℓ. We choose the pattern according to the number of occurrences of the pattern in the string s. However, it is not always the case that a frequent pattern represents snippets in the search result. The pattern should appear consecutively as well as frequent. Consider the case of $s = abcdabcdabcd$ for $\ell = 3$, where the repetitive pattern we have in mind is $abcd$ of length 4. We see three occurrences of abc. But there are separated by d and are not consecutive.

3. REPETITIVE PATTERN IN SEARCH RESULT

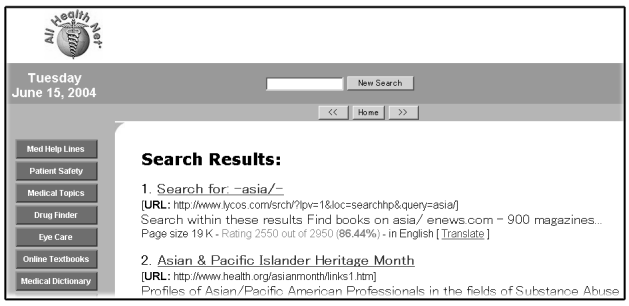
Input of our algorithm is five HTML files obtained from the same search site as search results. Output is a pattern that describes the snippets of search results. The basic structure of the algorithms as follows. Details are explained in subsections.

1. Translate HTML files into token sequences.
2. Guess the length of the pattern.
   (a) Compute the score vector of each sequence by the algorithm described in Section 2.
   (b) Sum up the score vectors of all sequences.
   (c) Return top 5 lengths of highest score.
3. Output pattern candidates with ranking.
   (a) Extract all the patterns for each length.
   (b) Evaluate the patterns by the length and the frequency and make a ranking.
   (c) Output top 3 patterns.

3.1 Token Sequence

In most of search results, records appear continuously and repetitively. We apply the method shown in the previous section to identifying the repetitive pattern in a search result. We do not consider an HTML document as a sequence of characters. We recognize it as a sequence of texts which are divided in the boundary of the HTML-tags. Namely, we consider an HTML documents as a sequence of tokens which consists of tags and plain texts.

Each token is considered as a symbol of the string matching algorithm, and the repetition discovery algorithm is applied to these token sequences. Fig. 1 shows the capture of a search result page, its HTML source code and the token sequence.



(a) Display with Browser

```
<b>1. </b> <a href="go.cgi?url=http://www.lycos.com/srch/?lpv=1&loc=se
archhp&query=asia/&keyword=asia"><b>Search for: -asia/-</b></a><br><fo
nt face=arial size=-2>[<b>URL:</b> http://www.lycos.com/srch/?lpv=1&lo
c=searchhp&query=asia/]</font><br>Search within these results Find boo
ks on asia/ enews.com - 900   magazines...<br><font face=arial size=-2
>Page size 19 K - <font color=red>Rating 2550 out of 2950 (<b>86.44%</
b>)</font> - in English [ <a href="http://jump.altavista.digital.com/t
rans.go?urltext=http%3A%2F%2Fwww.lycos.com%2Fsrch%2F%3Flpv%3D1%26loc%3
Dsearchhp%26query%3Dasia%2F&language=en">Translate</a> ]</font><P>
```

(b) HTML Source

```
<b>.</b>.<a><b>.</b></a><br><font>.<b>.</b>.
</font><br>.<br><font>.<font>.<b>.</b>.</font>
.<a>.</a>.</font><p>
```

(c) Token Sequence

Figure 1: Token Sequence of Search Result

Remember that, in the third step of the algorithm of pattern discovery, we generate pattern candidates and evaluate according to their score ranking. Considering texts as tokens increases the ranking of the pattern which uses plain texts as the structural description. Some texts are considered as contents to be extracted and others are considered as a part of pattern. The pattern texts should be matched strictly, but the contents texts should be matched with a wild-card. This is one of the reason we introduced "mismatch".

Fig.2 is an example of search result page and the extracted patterns from the page. We notice that some strings express the structure of repetitive pattern and that some kinds of tags have nothing to do with the repetitive pattern. Inline element tags, *e.g.*, `<br>`,`<i>` and `<b>`, are generally used for visual effect, while block tags, *e.g.*, `<ol>`,`<ul>` and `<table>` are used to describe the logical structure of the documents. However, in practice, it depends on the site whether a tag is used to represent the structure or not. If `<br>` is used to separate one search result from the next one, this tag should be considered as a component of the pattern.

We consider three types of pattern and hence three types of wrapper. The difference is the set of tags the wrapper recognizes as structural description.

A wrapper of **type a** (all) recognizes all of tags as tokens.A wrapper of **type b** (no-inline) recognizes all of tags except

The first 2 snippets in a search result :

```
<dt>1.  <a>1994-95 In Review</a><dd><small>Size:  <b>38 kb</b>; Last updated <b>19 Jan 2004</b></small><br>1994-95..
<dt>2.  <a>1992-94 In Review</a><dd><small>Size:  <b>28 kb</b>; Last updated <b>19 Jan 2004</b></small><br>1992-94..
```

a pattern described by tags :

```
<dt>.<a>.</a><dd><small>.<b>.</b>.<b>.</b></small><br>.
```

a pattern by our method :

```
<dt>.<a>.</a><dd><small>Size:  <b>.</b>; Last updated <b>.</b></small><br>.
```

Figure 2: String as Structure

for inline tags as tokens. A wrapper of **type c** (block) considers only the block-level tags as tokens. A tag which is not regarded a token is removed.

3.2 Selecting Pattern

An HTML document of search result usually contains many repetitions. Besides a listing of search results, which is an object of an extraction, many repetitions are contained in the advertising list, the link to the other contents, the casual repetition and so on. Therefore, the pattern detection algorithm yields many candidates of repetitive patterns. It is necessary to select the pattern which corresponds to the target of the extraction.

We adopted the minimality principle for pattern selection. When a pattern p appears repetitively, the doubled pattern pp appears as well. We choose only p as a minimal pattern. Besides, we demand the following constraints as search result patterns. (1) An anchor tag should appear. (2) If an open tag appears, then the corresponding close should appear. Anchor tags are necessary to all search results. When a continuous and repetitive pattern is extracted, a starting position of the pattern sometimes becomes ambiguous by the tags before and after that. The requirement to open tag and close tag is necessary to dissolve that ambiguity. We do not adopt the pattern which fails conform to these conditions.

We adapt the measurement $E(p) = C(p) \cdot L(p)$ as a ranking of the candidate patterns, where p is a pattern, $C(p)$ is a frequency of occurrence of p and $L(p)$ is the length of tags in p. This measurement realizes the following two heuristics. (1) An appropriate pattern appears many times. (2) An appropriate pattern is long enough.

4. EMPIRICAL EVALUATION

We use the testbed [9] to evaluate the accuracy of the patterns. It covers 51 databases randomly chosen from 114,540 Web pages with search forms. For each site of 51 databases, it provides 5 pages of search results with 5 keywords, and for each page of search results it provides the first snippet in the page and the list of result URLs. We evaluate the accuracy of a pattern by the number of first snippets that matches the pattern. So, the accuracy is in the range of 0 to 5.

In accordance with the process of the last section, we extracted repetitive patterns in each search site of the testbed. We introduced three types of tag sets, and extracted 5 patterns with high ranking. Therefore, 15 patterns are extracted for each search site, and we remove some of them by the constraints.

The extracted patterns and their accuracies are shown in the Table 1 for sites #7 and #21. Only 4 in 15 extracted patterns are left in both search sites by applying the constraints. In the site #7, three in four patterns correspond to the first snippet. The three patterns represent the same tag sequence. In the site #21, only one pattern with highest ranking corresponds to the first snippet.

The result of all sites is shown in Table 2.

Table 2: Evaluation of Patterns

ranking of pattern candidates	number of sites	rate
1	18	0.35
≤ 2	28	0.55
≤ 3	30	0.59

We analyzed the reasons where the patterns do not match. The first reason is incorrect HTML sources. Most of such HTML can be viewed with many HTML browsers without problem. However, such sources cause failure of pattern extraction. Most of these problems can be avoided using the tool that fix a mistake of HTML source. The above problem can be avoided by pre-processing HTML source with such a tool. There is also a problem about the omission of the tags. For example, it is allowed to omit `</p>`, `</tr>` and `</td>`. This problem can also be avoided with the above tool.

The second reason is a problem of the range of the pattern. There might be different patterns which can extract the same snippet. For example, "`<table>...</table>`" and "`<p><table>... </table></p>`". The score would be 0 if the extracted pattern did not correspond perfectly to the first snippet, even though the pattern matches the second and the other snippets. The site #2 was such a case.

The third reason is in the ranking score. We used a simple evaluation function in this experiment. The ranking score $E(p) = C(p) \cdot L(p)$ is favorable to **type a**, because all tags are counted in $L(p)$. There are nine sites which failed by this reason. We need some improvement in scoring function $E(p)$. To use the length of the pattern before removing non-tokens would be an improvement.

Table 1: Pattern Candidates and accuracy

Site No.	accuracy score	tag type	token length	tag length $L(p)$	frequency $C(p)$	ranking score $E(p)$	token sequence p
7	5	all	27	17	17	289	`<tr>.<td><font>.</font></td>.<td>.<a><strong><font>.</font></strong></a>.<font>.</font> .</td>.</tr>.`
7	5	no-inline	12	7	17	119	`<tr>.<td>.</td>.<td>.</td>.</tr>.`
7	5	block	12	7	17	119	`<tr>.<td>.</td>.<td>.</td>.</tr>.`
7	0	no-inline	4	3	5	15	`.</form></body>.`
21	5	all	17	13	500	6500	`<font><b>.</b></font> .<font><a>.</a></font> <i>.</i>.  `
21	0	block	58	8	10	80	`<td><a></a></td>.<td>.....<tr>.<td>.....<td>.....`
21	0	block	6	2	25	50	`.</div>....`
21	0	no-inline	16	9	5	45	`<tbody>.<tr>.<td>.<div><a>.</a></div></td></tr></tbody>....`

5. CONCLUSION

We proposed an automatic generation algorithm of Deep Web Wrappers based on the discovery of repetitive patterns. To discover a repetitive pattern, we used a string matching with mismatches and FFT. We showed that repetitive patterns can be extracted for 60% of target sites. Moreover, we showed that the extracted repetitive pattern functioned as a Deep Web Wrapper.

Many improvements will be possible for the ranking score. The testbed [9] provides other information which we can use for more strict evaluation.

6. REFERENCES

[1] Chang, C.-H., and Lui, S.-C. IEPAD: Information Extraction Based on Pattern Discovery. Proceedings of *the 10th International Conference of World Wide Web*, pp. 4–15, 2001.

[2] Cohen, W. W., Hurst, M., and Jensen, L. S. A Flexible Learning System for Wrapping Tables and Lists in HTML Documents. Proceedings of *the 11th International Conference on World Wide Web*, pp. 232–241, 2002.

[3] Embley, D. W., Jiang, Y. S., and Ng, Y.-K. Record-boundary discovery in Web documents. Proceedings of *1999 ACM SIGMOD International Conference on Management of Data*, pp. 467-478, 1999.

[4] Hamilton, N. The Mechanics of a Deep Net Metasearch Engine. Proceedings of *the 12th International World Wide Web Conference*, 2003.

[5] Kushmerick, N. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, Vol.118, No.1-2, pp .15-68, 2000.

[6] Muslea, I., Minton, S., and Knoblock, C. Active Learning for Hierarchical Wrapper Induction. Proceedings of *16th National Conference On Articial Intelligence (AAAI-99)*, 1999.

[7] Nakatoh, T., Baba, K., Ikeda, D., Yamada, Y., and Hirokawa, S. An Efficient Mapping for Score of String Matching. Proceedings of *the Prague Stringology Conference '03*, pp.127-136, 2003.

[8] Taguchi, T., Koga, Y., and Hirokawa, S. Integration of Search Sites of the World Wide Web. Proceedings of *Intern. Forum cum Conf. on Information Technology and Communication*, Vol. 2, (2000) pp.25–32.

[9] Yamada, Y., Craswell, N., Nakatoh, T., and Hirokawa, S. Testbed for Information Extraction from Deep Web. Proceedings of *the 13th International World Wide Web Conference*, Alternate Track Papers and Posters, pp.346-347, 2004.