Automatic Extraction of Information Blocks Using PAT Trees[*]

Chia-Hui Chang
Dept. of Computer Science and Information Engineering
National Central University
Chung-Li, 320, Taiwan
chia@csie.ncu.edu.tw

Chun-Nan Hsu
Institute of Information Science
Academia Sinica
Taipei, 115, Taiwan
chunnan@iis.sinica.edu.tw

Abstract

Information extraction from semi-structured Web documents is a critical issue for software agents on the Internet. Previous work in *wrapper induction* aim to solve this problem by applying machine learning to automatically generate extractors, but this approach still requires human intervention to provide training examples. In this paper, we present a novel approach that extracts information blocks without training examples using a data structure called a *PAT tree*. PAT trees allow the system to efficiently recognize repeated patterns in a semi-structured Web page. From these repeated patterns, information blocks can be easily located based on some domain independent selection criteria. The entire system runs automatically without any human intervention. Experimental results show that our approach performs well with a recall rate near 90 percent on a wide range of output pages of popular search engines.

1 Introduction

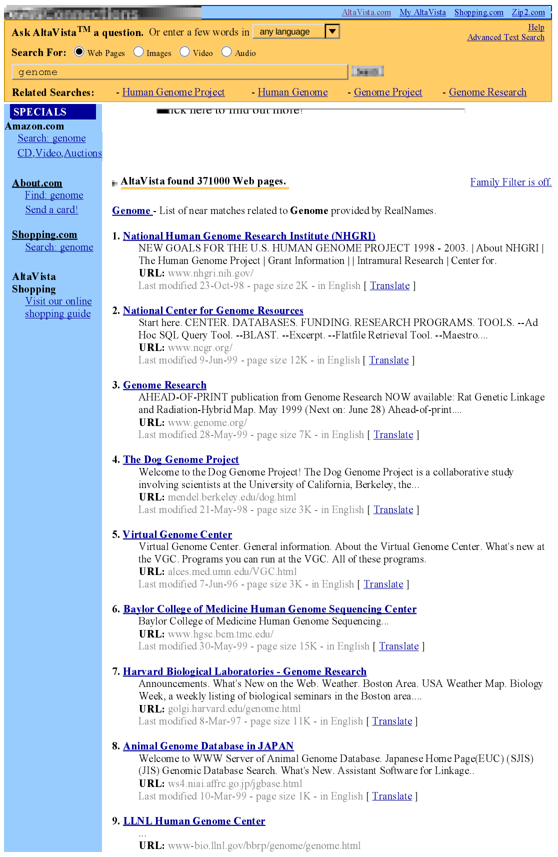
In recent years, collation of information on the World Wide Web has become the main issue in applications that provide value-added services. From multi-search engines, to software agents or softbots, retrieving and integrating data from various information sources can significantly increase the utility of individual Web sites. Extracting data from the Web requires a *wrapper* that "wraps" around a Web site and extracts data from Web pages. However, programming *wrappers* is often labor-intensive and error-prone. Besides, since the formats of Web pages are subject to change, maintaining these wrappers can be expensive and impractical. To save the efforts of hand-coding wrappers, researchers are developing new approaches to automatize wrapper construction. For example, *wrapper induction* is one of such techniques that learns extraction rules by generalizing from training examples. The key idea underlying these wrappers is that the information to be extracted can be located based on "landmarks". That is, the wrappers extract a tuple by scanning the input Web page, recognizing the "landmarks" that are used to delimit information records. In [6], Kushmerick et al. identified a family of wrapper classes (including LR, HLRT, OCLR, HOCLRT, etc.) and the corresponding induction algorithms which generalize from labeled examples to extraction rules. More expressive wrapper structure are introduced lately. *Softmealy* by Hsu and Dung [4] uses a wrapper induction algorithm to generate extractors that are expressed as finite-state transducers. Meanwhile, Muslea et al. [7] proposed *"STALKER"* that generates wrappers based on a set of disjunctive landmark automata organized as a hierarchy.

[*]This work is partially done while the first author was at the Institute of Information Science, Academia Sinica.

```
<HTML><TITLE>Some Country Codes</TITLE>
<BODY>
<B>Congo</B> <I>242</I><BR>
<B>Egypt</B> <I>20</I><BR>
<B>Belize</B> <I>501</I><BR>
<B>Spain</B> <I>34</I><BR>
</BODY></HTML>
```

Figure 1: Sample HTML page P_{cc}

In all these work, wrappers are induced from training examples such that "landmarks" or "delimiters" can be generalized from common prefixes or suffixes. However, labeling these training examples is sometimes time-consuming. Hence, it will be interesting if we can eliminate the effort of labeling and extract information block automatically. One observation from these input pages is that the information to be extracted is often placed in a particular order such that repetitive patterns can be found in these Web pages when multiple records aligned together. For instance, in the example given by Kushmerick in [6] (presented in Figure 1), the sequence "<B>Alph</B> <I>Num</I><BR>" is repeated four times, where text strings "Congo", "Egypt", "Belize" and "Spain" are regarded as token class *Alph*, and "242", "20", "501" and "34"



AltaVista.com My AltaVista Shopping.com Zip2.com

Ask AltaVista™ a question. Or enter a few words in any language

Help
Advanced Text Search

Search For: Web Pages Images Video Audio

genome

Related Searches: - Human Genome Project - Human Genome - Genome Project - Genome Research

SPECIALS
Amazon.com
Search: genome
CD,Video,Auctions

About.com
Find: genome
Send a card!

Shopping.com
Search: genome

AltaVista Shopping
Visit our online shopping guide

AltaVista found 371000 Web pages. Family Filter is off.

Genome - List of near matches related to **Genome** provided by RealNames.

1. National Human Genome Research Institute (NHGRI)
NEW GOALS FOR THE U.S. HUMAN GENOME PROJECT 1998 - 2003. | About NHGRI | The Human Genome Project | Grant Information | | Intramural Research | Center for.
URL: www.nhgri.nih.gov/
Last modified 23-Oct-98 - page size 2K - in English [Translate]

2. National Center for Genome Resources
Start here. CENTER. DATABASES. FUNDING. RESEARCH PROGRAMS. TOOLS. --Ad Hoc SQL Query Tool. --BLAST. --Excerpt. --Flatfile Retrieval Tool. --Maestro....
URL: www.ncgr.org/
Last modified 9-Jun-99 - page size 12K - in English [Translate]

3. Genome Research
AHEAD-OF-PRINT publication from Genome Research NOW available: Rat Genetic Linkage and Radiation-Hybrid Map. May 1999 (Next on: June 28) Ahead-of-print....
URL: www.genome.org/
Last modified 28-May-99 - page size 7K - in English [Translate]

4. The Dog Genome Project
Welcome to the Dog Genome Project! The Dog Genome Project is a collaborative study involving scientists at the University of California, Berkeley, the...
URL: mendel.berkeley.edu/dog.html
Last modified 21-May-98 - page size 3K - in English [Translate]

5. Virtual Genome Center
Virtual Genome Center. General information. About the Virtual Genome Center. What's new at the VGC. Programs you can run at the VGC. All of these programs.
URL: alces.med.umn.edu/VGC.html
Last modified 7-Jun-96 - page size 3K - in English [Translate]

6. Baylor College of Medicine Human Genome Sequencing Center
Baylor College of Medicine Human Genome Sequencing...
URL: www.hgsc.bcm.tmc.edu/
Last modified 30-May-99 - page size 15K - in English [Translate]

7. Harvard Biological Laboratories - Genome Research
Announcements. What's New on the Web. Weather. Boston Area. USA Weather Map. Biology Week, a weekly listing of biological seminars in the Boston area....
URL: golgi.harvard.edu/genome.html
Last modified 8-Mar-97 - page size 11K - in English [Translate]

8. Animal Genome Database in JAPAN
Welcome to WWW Server of Animal Genome Database. Japanese Home Page(EUC) (SJIS) (JIS) Genomic Database Search. What's New. Assistant Software for Linkage..
URL: ws4.niai.affrc.go.jp/jgbase.html
Last modified 10-Mar-99 - page size 1K - in English [Translate]

9. LLNL Human Genome Center
...
URL: www-bio.llnl.gov/bbrp/genome/genome.html

Figure 2: The search result of 'genome" from AltaVista

are regarded as token class *Num*, following some parsing or tokenizing convention. In many other examples on the Web, especially search engines, the matched results are often placed in a regular manner, e.g., the output page of AltaVista (Figure 2). From these examples, we can find that repeats that occur regularly and closely in a Web page often represent a block of meaningful information that might be interesting to users. That is, the target block to be extracted. These insights and observations motivate us to look for an approach to recognizing repeated patterns that occur regularly and closely in a given Web page.

In this paper, we utilize a data structure called a PAT tree [2] in which repeated patterns in a given input string can be efficiently identified. A PAT tree is an efficient data structure successfully used in the area of information retrieval for indexing a continuous data stream Using this data structure to index an input string, all possible character strings, including their frequency counts and their positions in the original input string can be easily retrieved.

In the next section, we give some backgrounds of the PAT tree and its application in string searching. Section 3 describes how to translate an input HTML page so as to construct a PAT tree. Section 4 discusses the pattern validation criteria for selecting candidate repeats. Section 5 reports experimental results. The last section presents our conclusion and the directions of future work.

2 The PAT Tree

A PAT tree is a Patricia tree (Practical Algorithm to Retrieve Information Coded in Alphanumeric) constructed over all the possible semi-infinite strings (called *sistrings*) [2]. A Patricia tree is a particular implementation of a binary digital tree (or trie in short) such that the abstract data type *sistring* is represented as a suffix string that ends with a special character not occurring anywhere in the input string. Like a suffix tree [3], the Patricia tree stores all its data at the external nodes and keeps one integer, the bit-index, in each internal nodes as an indication of which bit of a query is to be used for branching. This avoids empty subtrees and guarantees that every internal node will have non-null descendants. For a set of n sistrings to be indexed, there will be n external nodes in the PAT tree and $n-1$ internal nodes. This makes the tree $O(n)$ in size.

Figure 3 shows an example of a PAT tree over a sequence of bits. External nodes are indicated by squares that contain a reference to a sistring, and internal nodes are indicated by a circle and contain a displacement (or index) to the root. More specifically, the Patricia tree for a string S of size n uses an index at each internal node to indicate the bit used for that node's branching. A "zero" bit will cause a branch to the left subtree, and a "one" bit will cause a branch to the right subtree. Conceptually, each edge between two nodes is *labeled* with a nonempty substring: the left edges begin with a zero and the right edges begin with one. Hence, the concatenation of the *edge-labels* on the path from the root to a leaf node with reference i exactly spells out the sistring i of S. For example, sistring 3 is the concatenation of edge-labels, "0", "110" and "1100..." between nodes 1, 2, 5, and leaf 3.

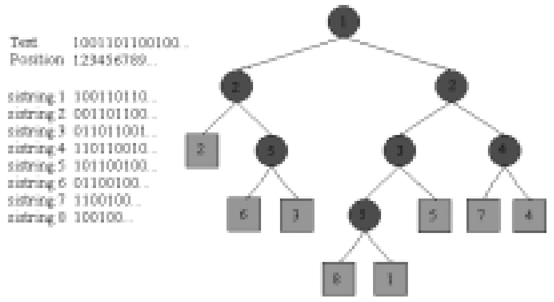


Figure 3: PAT tree when the sistrings 1 through 8 have been inserted

Note that if the tree is to index a sequence of characters, the binary codes for the characters can be used. (For simplicity, each character is encoded as fixed-length binary code.) In this case, only those bit positions that are the beginning of a character needs to be indexed. For example, given a finite alphabet Σ of a fixed size, each character $x \in \Sigma$ is represented by a binary code of length $l = \lceil \log_2 |\Sigma| \rceil$. For a sequence S of n characters, the binary input B will have $n * l$ bits, but only the $[i * l + 1]$th bit has to be indexed for $i = 0, \ldots, n-1$. The constructed PAT tree $\mathcal{T}$ will have n external nodes pointing to sistrings numbered $1, \ldots, n$.

It follows from the tree construction algorithm that every subtree of a PAT tree has all its sistrings with a common prefix. Hence, it allows surprisingly efficient, linear-time solutions to complex string search problems. For example, string prefix searching, proximity searching, range searching, longest repetition searching, most frequent searching, etc [2, 3]. In this paper, we focus especially on the problem of finding *maximal repeat* using a PAT tree.

Definition Given an input string S, we define *maximal repeat* α as a substring of S that occurs in two positions p_1, p_2 in S such that $p_1 \neq p_2$ and the $(p_1 - 1)$th character in S is different from the $(p_2 - 1)$th character and the $(p_1 + |\alpha|)$th is different from the $(p_2 + |\alpha|)$th.

Since every internal node indicates a branch, it also shows a different bit following the common prefix between two sistrings. Hence, the concatenation of the edge-labels on the path from the root to an internal node represents one repeated sequence in the input string. That is, to find maximal repeats we only need to consider path-labels that end at internal nodes in the suffix tree $\mathcal{T}$. However, not every path-label or repeated sequence is a maximal repeat. Let's call character $(p_1 - 1)$ of S the *left character* of sistring p_1. For a path-label of a node v to be a maximal repeat, at least two leaves in the v's subtree should have different left characters. Let's call such a node a *left diverse*. Followed by definition, the property of being left diverse propagates upward in $\mathcal{T}$. Therefore all maximal repeats in S can be found in linear time. In summary, we come to the following lemma with this discussion.

Lemma A string α which labels a path from the root to a node v in $\mathcal{T}$ is a *maximal repeat* if and only if v is a left diverse.

The definition of maximal repeats is necessary for our problem since it captures all meaningful repetitive structures in a clear way and avoids generating overwhelming outputs. In addition, by recording the frequency counts and the reference positions in the nodes of a PAT tree, we can easily know how many times a substring is repeated and decide whether such a pattern corresponds to a potential information block we want to extract.

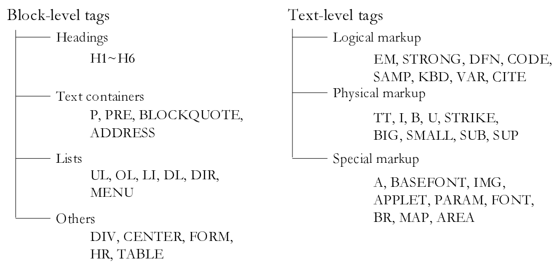

Block-level tags
Headings
H1~H6
Text containers
P, PRE, BLOCKQUOTE, ADDRESS
Lists
UL, OL, LI, DL, DIR, MENU
Others
DIV, CENTER, FORM, HR, TABLE
Text-level tags
Logical markup
EM, STRONG, DFN, CODE, SAMP, KBD, VAR, CITE
Physical markup
TT, I, B, U, STRIKE, BIG, SMALL, SUB, SUP
Special markup
A, BASEFONT, IMG, APPLET, PARAM, FONT, BR, MAP, AREA

Figure 4: Block-level tags vs. text-level tags

3 Translating HTML Pages

To apply the PAT-tree technique to extract information from HTML pages, the input Web page is first parsed into *tokens*. Each token is denoted by a corresponding token class. For instance, one common token class that one can come up with for Web pages is the HTML tag classes, denoted as Html(<tag_name>). We call this class as "tag class." The tags themselves can also be classified into hierarchy depending on what level of information we want to extract. According to the HTML structure tree, the tags in the BODY section of a document can be grouped in two distinct groups: *block level* and *text level* tags. The former make up the document's structure, and the latter "dress up" the contents of a block [8]. As shown in Figure 4, block-level tags include headings, lists, text containers, and others such as tables, forms etc; while text-level tags include logical markups, physical markups, and special markups that are used to mark up text inside block-level tags. On the other hand, tag can also be divided into start tags and end tags. In addition to the tag class, we can also define "text token class", Text(_), which embraces all text strings except HTML tags. Detailed classification like *word* and *non-word* token classes can be found in Hsu and Dung [4].

Re-examing the Web page from AltaVista in Figure 2, part of the HTML source and the corresponding translation are presented in Figure 3. This search result for query "genome" from AltaVista contains ten matches that can be extracted. The translation of this HTML page forms a repeated segment "Html(<dl>)Html(<dt>)Html(<b>)Text(_)... Html(</a>)Text(_)Html(</dl>)" which occur contiguously for ten times. This repeat can be easily recognized in PAT trees as described in Section 2. If we apply the PAT tree construction algorithm to the translated HTML codes, we will have a subtree with the above pattern as a path-label between the PAT tree root and the subtree root; and there will be ten leaves in the subtree indicating the occurring positions of the pattern. Thus, the problem of semi-structured information extraction is converted into an easier problem of repeated pattern recognition.

```
<html><head><title>AltaVista: Simple Query
&quot;genome&quot;</title>
<style><!– a:color:#000099 a:vlinkcolor:#663366
a:hovercolor:#007FFF ></style></head>
<body bgcolor="#ffffff" text="#000000"
link="#000099" vlink="#663366" alink="#ff0000">
<table
⋮
<font size=-1>List of near matches related to <b>Genome
</b> provided by RealNames.</font> <P> </font>
<font face=Arial size=-1><dl><dt><b>1. </b><a
href="http://www.nhgri.nih.gov/"><b> National Human
Genome Research Institute (NHGRI)</b></a><dd>
NEW GOALS FOR THE U.S. HUMAN GENOME
PROJECT 1998 - 2003. — About NHGRI — The Human
Genome Project — Grant Information — — Intramural
Research — Center for.<br><b> URL:</b> <font
color=gray>www.nhgri.nih.gov/<br>
Last modified 23-Oct-98 - page size 2K - in English
</font> [&nbsp; <a
href="http://jump.altavista.com/trans.go?urltext=
http://www.nhgri.nih.gov/&language=en">
Translate</a>&nbsp;]</dl>
<dl><dt><b>2. </b><a
href="http://www.ncgr.org/"><b> National Center for
Genome Resources</b></a><dd>
Start here. CENTER. DATABASES. FUNDING.
RESEARCH PROGRAMS. TOOLS. –Ad Hoc SQL Query
Tool. –BLAST. –Excerpt. –Flatfile Retrieval Tool.
–Maestro....<br><b> URL:</b> <font
color=gray>www.ncgr.org/<br>
Last modified 9-Jun-99 - page size 12K - in English
</font> [&nbsp; <a
href="http://jump.altavista.com/trans.go?urltext=
http://www.genome.org/&language=en">
Translate</a>&nbsp;]</dl>
```

(a) The HTML source code

```
Html(<html>)Html(<head>)Html(<title>)Text(_)
Html(</title>)Html(<style>)Html(comment)
Html(</style>)Html(</head>)Html(<body>)
Html(<table>)
⋮
Html(<font>)Text(_)Html(<b>)Text(_)Html(</b>)
Text(_)Html(</font>)Html(<P>)Html(</font>)
Html(<font>)Html(<dl>)Html(<dt>)Html(<b>)
Text(_)Html(</b>)Html(<a>)Html(<b>)Text(_)
Html(</b>)Html(</a>)Html(<dd>)Text(_)Html(<br>)
Html(<b>)Text(_)Html(</b>)Html(<font>)
Text(_)Html(<br>)Text(_)Html(</font>)Text(_)
Html(<a>)Text(_)Html(</a>)Text(_)Html(</dl>)
Html(<dl>)Html(<dt>)Html(<b>)Text(_)Html(</b>)
Html(<a>)Html(<b>)Text(_)Html(</b>)
Html(</a>)Html(<dd>)Text(_)Html(<br>)Html(<b>)
Text(_)Html(</b>)Html(<font>)Text(_)
Html(<br>)Text(_)Html(</font>)Text(_)Html(<a>)
Text(_)Html(</a>)Text(_)Html(</dl>)
```

(b) The parsed token classes

Figure 5: The HTML source for Figure 2 (a) and translated code (b).

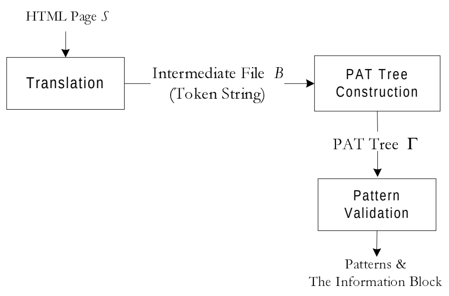


Figure 6: Information extraction procedure

4 Pattern Validation

In an overview point, our information extraction procedure can be briefed by three steps: HTML translation, PAT tree construction, and pattern validation. The flowchart is shown in Figure 6. The necessity of "HTML translation" is to stand out the repeat patterns, while the PAT tree is the most appropriate data structure to facilitate the finding of repeat patterns. As for pattern validation, it is to ensure accurate extraction. In the previous sections, we have discussed the first two steps. This section describes the pattern validation step.

In Section 2, we define maximal repeats which capture meaningful repetitive structures while avoid overwhelming repeats to be presented. However, not every maximal repeat corresponds to an interesting information block to us. What we are interested is those maximal repeats that occur regularly in vicinity. An example is the maximal repeat that occurs continuously for ten times in the example Web page from AltaVista (Figure 3). Such a repeat satisfies the requirements for maximality, regularity, adjacency, etc. Nonetheless, the validation step has to be quantified since not all information we want is placed in such a perfect order. Hence, we also need to verify whether a maximal repeat satisfies the following criteria during the top-down traversing of the PAT tree $\mathcal{T}$.

Regularity To validate this property, we demand all sistrings in the subtree (of a maximal repeat) appear spaced at an interval of approximate equal distance. Suppose the sistrings of a maximal repeat α are sorted by its position such that sistrings $p_1 < p_2 < p_3 \ldots < p_k$. We say the maximal repeat appears *regularly* if the standard derivation of the interval between two adjacent occurrences is less than a certain amount (say ϵ=0.5) of the interval's mean.

Localization This property is required to avoid extracting repeats that are scattered too far across the input. Thus, localization can be seen as a remedy to the cases when the requirement of regularity is loosen. We measure

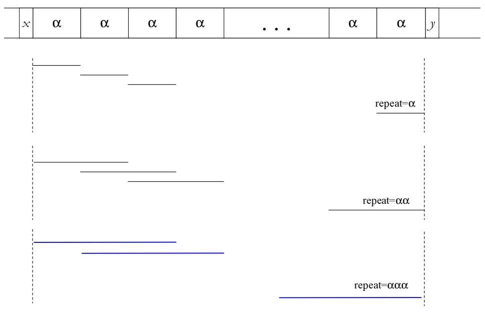


Figure 7: A tandem repeat of string α for ten times

the degree of localization through the computation of repeat density:

$$\frac{k * |\alpha|}{p_k - p_1 + |\alpha|} \quad (1)$$

Only maximal repeats with density greater than a bound θ are considered qualified to provide enough information to be extracted.

Vicinity Consider the perfect example we mentioned above where sequence α is repeated ten times (Figure 7). In such cases, not only α, but also $\alpha\alpha$, $\alpha\alpha\alpha$, etc. all qualify for maximal repeats. Hence, the property of vicinity is required to avoid output multiple maximal repeats which originate from the same repeat sequence. To satisfy this requirement, the mean distance between adjacent occurring positions should be no less than the length of the maximal repeat. That is, overlapping of repeats are disallowed. In practice, we require the interval's mean be greater than three quarters the length of a maximal repeat.

Heuristic Finally, the repeat patterns themselves also manifest whether they contain potential information to be extracted. For instance, a repeat which consists of all tag tokens (H1) and no text token can be excluded from consideration since no text information is contained. Besides, patterns that begin with ending tags (H2) or text tokens (H3) can also be eliminated. However, the later heuristic (H3) is not perfect since not all information block begin with a tag token. That is, this heuristic can help reduce unwanted patterns as well as target patterns. The default heuristics used are H1 and H2. The effect is discussed in next section.

5 Experiments

To demonstrate the effect of our extraction procedure, we choose fourteen state-of-the-art search engines. The first output pages of 10 test queries are used as the test pages for each search engine. The experiments will show that different translation of the input HTML pages joined with various degree of regularity and localization yield different degrees of extracting performance.

In each table, we show the following performance statistics. The second column shows the number of sistrings indexed in the PAT tree, that is also the number of tokens counted when each HTML tag corresponds to a token and other texts between two tags are replaced by the Text(_) token. The third column represents the number of maximal repeats computed from the PAT tree. The following columns show the number of maximal repeats remained after each validation criteria vicinity, regularity, localization and heuristic are validated in turn.

In Table 1, we adopt two heuristics, the first one removes patterns with all text tokens (H1) and the second one removes patterns beginning with end tag tokens (H2). Note that the order of the application of these criteria does not affect the final result, which is evaluated by the *recall rates* shown in the last column. The recall rate is defined by the ratio of the number of information records enumerated by a maximal repeat and the number of information records contained in the test document. An information record is said to be enumerated by a pattern if one third of the record's content is part of the repeated pattern. If there are more than one maximal repeat, the largest recall rate is used to represent its result. For example, suppose one target information block contains 20 matches and a maximal repeats contains 18 matches, then the recall is 18/20.

Opposed to recall, the other main measure that our pattern recognition system tries to keep minimum is the number of output maximal repeats. In a sense, the inverse of this number corresponds to the concept of *precision*, which indicates the correct answer is selected from several candidate maximal repeats. If the number of output maximal repeats is 1, this only maximal repeat possibly represents the target information block. In such a case, the precision is (1 or 0), which means our system correctly finds the information block (or not). If there are more than one maximal repeats, the decision of which repeat corresponds to our target information block is left to the user.

For Table 1, with regularity threshold ϵ set to 0.5 and localization threshold θ set to 0.45, the system identifies on average 4.8 patterns and extracts 74 percent of target records. As shown in Table 1, we can find that regularity, vicinity and localization all play an important role in filtering maximal repeats. Heuristic validation, including all string pattern elimination (H1) and removing repeats starting with ending tags (H2) seems the most inefficient in reducing unsolicited patterns. But if we

Search Engine	sistrings	maximal	regularity	vicinity	localization	H1&H2	recall
AltaVista	1363.70	116.10	61.80	7.40	6.80	6.80	100/100
Cora	1073.80	73.70	29.70	17.30	5.60	5.10	61/100
Excite	1113.30	67.20	12.30	3.90	1.10	1.10	30/100
Galaxy	1771.10	62.60	26.50	19.50	6.90	6.90	186/196
Hotbot	727.80	38.80	25.00	7.70	6.30	6.30	100/100
Infoseek	1080.10	77.10	18.60	18.00	10.40	10.40	93/100
Lycos	900.10	49.90	6.90	1.20	1.20	1.20	100/100
Magellan	467.30	8.10	7.00	1.00	1.00	1.00	100/100
Metacrawler	1367.00	118.10	33.70	24.40	8.50	7.60	114/200
Northernlight	1180.30	54.60	17.40	5.60	3.30	3.30	72/100
Openfind	1216.60	35.50	19.50	10.20	1.50	1.50	52/200
Savvysearch	1092.60	71.60	43.60	12.30	6.90	6.90	144/150
Stpt.com	2045.50	93.00	34.50	14.60	6.90	6.60	0/250
Webcrawler	966.10	54.70	17.20	4.00	2.50	2.50	250/250
			$\epsilon = 0.5$		$\theta = 0.45$	4.8	0.74

Table 1: No. of maximal repeats computed for each search engine (averaged from 10 test pages). The middle block shows the number of maximal repeats which pass the validation criteria regularity, vicinity, localization, and heuristic in turn.

remove repeats starting with ending tags (H2) and text token (H3) during indexing, i.e. skip sistrings that start with ending tags or text token, a lot of effort can be reduced as shown in Table 2, where 22 percent sistrings are reduced due to ending tags and 27 percent are due to text tokens. However, as we discussed above, target repeats can also start with text token class hence they can be eliminated during this phase just like some test pages from Infoseek, Openfind, Savvysearch and Stpt.com where the pattern begins with a text token. Therefore, the recall rate is reduced to 0.60. Hence, in the following experiments H3 is not used, i.e. sistrings that begin with text token are still indexed in the PAT tree.

As mentioned above, the goal of the validation criteria is to keep minimum the number of maximal repeats so the final one represents our target information block. However, the results indicate that there is a tradeoff between recall and precision (the inverse of the number of output maximal repeats). While we try to increase precision, i.e. reduce the number of output repeats, the recall is often decreased accordingly. For example, applying heuristic H3 removes many useless repeats which beginning with text tokens. However, it also decrease recall because some target patterns are eliminated. Similarly, when the requirement for localization is loosen, say the localization threshold is reduced to 0.15, the number of final patterns almost doubles from 4.8 to 8.4, where 85 percent target records are captured by the additional 3.6 maximal repeats (Table 3).

The reason behind this phenomenon is due to the unstructured characteristic of the information block, hence a target information block may be presented in more than one patterns. For this reason, we have the rule that when we compute the recall rate for each pattern, only those records with one third of contents spelt out by the pattern are counted. Thus, even if all records of the target information block are contained in a maximal repeat, the work of extending each record's (repeat's) border between two repeats still remains. Indeed, the length of the repeat for patterns with a higher recall rate is often shorter than the patterns with a lower recall rate. In other words, when validating the localization of a repeat with a lower density value, it becomes harder to locate the border of the data record since the common repeated patterns are shorter. Table 3 shows the result of the experiment designed to demonstrate this point. In this table, the last column also shows the corresponding density of the pattern that is chosen for identifying the target information. According to the result, the density of the chosen pattern is greater than 0.5 for half the examples. For those repeats with density less than 0.5, further filtering is needed to locate the border of each record.

In addition to adjusting the parameters for regularity and localization, another factor that might produce different repeat patterns is how we translate our original HTML pages for constructing PAT trees. Table 1, 2, 3 present the case when all tag classes are involved in the translation (each tag is translated to their corresponding token class). With different translation, say ignoring text-level tags, target blocks that cannot be recognized may be extracted, but it may also cause other patterns to be destroyed. For example, skipping physical markups, including <TT>, <I>, <B>, <U>, etc. (Fig 4), will decrease 13.2 percent tags. With the regularity threshold ϵ set to 0.5 and localization threshold θ set to 0.45, 96 percent targets can be recognized. This difference is due to the increasing for Excite, Openfind, etc. comparing to the result

Search Engine	sistrings	H2	H3	maximal	regularity	vicinity	localization	H1	recall
AltaVista	1363.7	1006.3	612.5	91.7	60.5	4.5	4.00	4	100/100
Cora	1073.8	937.7	611.5	34.9	14.3	10.2	4.00	4	38/100
Excite	1113.3	840.7	597.8	55.8	11.8	3.4	1.00	1	30/100
Galaxy	1771.1	1333.5	774.9	59.5	25.5	18.5	7.00	7	186/196
Hotbot	727.8	561.6	365.4	32.4	24.3	6.7	6.00	6	100/100
Infoseek	1080.1	804.3	560	57.1	10.9	10.9	5.00	5	39/100
Lycos	900.1	686.2	444.6	29.2	6.9	1.2	1.00	1	100/100
Magellan	467.3	353.7	257.9	7.1	7	1	1.00	1	100/100
Metacrawler	1367	1079.3	651.3	63.6	18	9.8	4.10	3.2	36/200
Northernlight	1160.7	930.4	762.9	53.5	16.5	4.4	3.30	3.3	72/100
Openfind	1216.4	994.5	512.7	16.5	5.9	5.6	1.20	1.2	40/200
Savvysearch	1092.6	837.6	579.6	30.8	26.1	3.9	1.80	1.8	49/150
Stpt.com	2045.5	1548.9	1067.1	75.3	30.4	11.5	4.80	4.5	0/250
Webcrawler	966.1	800.9	554.9	47.8	17.2	4	2.60	2.6	250/250
		-22%	-27%		$\epsilon = 0.5$		$\theta = 0.45$	3.2	0.60

Table 2: Sistrings that start with ending tags (H2) or text token (H3) are ignored during PAT tree construction phase.

Search Engine	maximal	regularity	vicinity	localization	H1&H2	recall	density
Altavista	91.70	60.50	4.50	4.50	4.50	100/100	0.85
Cora	73.70	29.70	17.30	12.90	12.10	86/100	0.36
Excite	67.20	66.90	12.40	4.00	1.30	30/100	0.26
Galaxy	62.60	26.50	19.50	15.80	15.80	186/196	0.68
Hotbot	38.80	26.30	7.70	7.70	7.70	100/100	1.01
Infoseek	77.10	18.60	18.00	16.40	15.50	95/100	0.51
Lycos	49.90	6.90	1.20	1.20	1.20	100/100	0.95
Magellan	8.10	7.00	1.00	1.00	1.00	100/100	1.04
Metacrawler	119.10	33.70	24.40	20.90	19.90	177/200	0.37
Northernlight	54.60	17.40	5.70	4.70	4.70	95/100	0.55
Openfind	35.67	20.22	11.22	9.00	9.00	74/200	0.37
Savvysearch	71.50	43.90	11.80	10.50	10.50	150/150	0.41
Stpt.com	92.90	36.50	14.70	11.30	10.20	175/250	0.34
Webcrawler	51.60	16.90	4.00	4.00	4.00	250/250	0.94
		$\epsilon = 0.5$		$\theta = 0.15$	8.4	0.85	

Table 3: No. of repeats output when localization threshold is set to 0.15.

Search Engine	no logical tags			no special tags			no physical tags		
	sistrings	blk	recall	sistrings	blk	recall	sistrings	blk	recall
AltaVista	1363.7	7.2	100/100	880.8	1.8	100/100	1241.3	6	100/100
Cora	1073.8	5.7	100/100	788.5	7.8	100/100	1004.0	6.2	100/100
Excite	1113.3	1.0	45/100	831.9	1.7	100/100	1040.4	1.6	100/100
Galaxy	1771.1	6.9	194/196	1294.5	6.2	191/196	1024.0	8.0	196/196
Hotbot	727.8	6.0	100/100	406.2	3.6	100/100	637.0	6.0	100/100
Infoseek	1080.1	10.4	100/100	777.2	4.5	82/100	918.9	7.3	100/100
Lycos	900.1	1.2	100/100	900.1	1.0	100/100	900.1	1.3	100/100
Magellan	467.3	1.0	100/100	331.8	1.0	100/100	374.7	1.0	100/100
Metacrawler	1367.0	7.6	177/200	831.9	6.4	120/200	1117.4	7.4	194/200
Northernlight	1180.3	3.3	95/100	950.8	3.2	72/100	1106.5	3.3	72/100
Openfind	1214.6	1.5	50/200	1050.6	4.1	118/200	1061.2	3.6	160/200
Savvysearch	1092.6	6.9	145/150	665.3	5.5	140/150	989.2	4.1	150/150
Stpt.com	2045.5	6.6	0/250	1699.5	5.6	250/250	1893.1	6.8	250/250
Webcrawler	908.4	2.5	250/250	664.7	2.6	250/250	883.0	2.6	250/250
	-0.4%	4.8	0.86	-26.2%	3.9	0.90	-13.2%	4.7	0.96

Table 4: Different translation of HTML pages result in different patterns.

in Table 1. Table 4 shows the effect of different translations. In conclusion, different translations of HTML files may result in different patterns. A translation that enables the extraction of one page may disables the pattern of another page.

6 Conclusion and Future Work

In this paper, we present an approach to semi-structured information extraction based on the recognition of repeated patterns in the translated HTML input. The underlying technique is the string manipulation based on the data structure called the PAT tree. The PAT tree has been applied in the field of information retrieval for indexing for a long time [2]. It has also been used in Chinese keyword extraction [1]. The essence of a PAT tree is a binary suffix tree, which has also been applied in bioinformatics for finding repeated substring in genomes [5] etc. In the application of information extraction, we are not only interested in repeats but also repeats that represent interesting information blocks.

To enable the extraction of repeat patterns, the input HTML pages are translated into a simpler form consisting of token classes and a PAT tree is constructed based on these token classes. Next, the validation criteria: regularity, vicinity, localization and H2 and H3 are applied as validations of interesting patterns, so that high recall can be obtained. In our experiments, with regularity threshold 0.5 and localization threshold 0.45, our extracting procedure performs well on a wide range of semi-structured Web pages.

One interesting direction that could further improve the result of our extracting procedure is the merging of the patterns in the final result. As we described in the previous sections, a target information block may reveal in several patterns. Each stands for a fragment of the target record. Merging similar patterns may potentially help to reconstruct each record's boundary.

Compared to other information extraction research [4, 6, 7], the goal of our approach is to find repeats in the given HTML pages and recognizes potential information blocks to be extracted. The procedure is fully automatic, but is less powerful in terms of handling different permutations of attributes in a record. Indeed, we are currently investigating how to apply such an approach in *SoftMealy* wrapper induction system as a preprocessing step to suggest potential information blocks and record boundaries that serve as training examples. The resulting system can be applied to equip an Internet spider that traverses and extracts interesting information on the Web without any human intervention.

Acknowledgements

We would like to thank Lee-Feng Chien, Ming-Jer Lee and Jung-Liang Chen for providing their PAT tree code for us.

References

[1] L.F. Chien, "PAT-tree-based keyword extraction for Chinese information retrieval," In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval.* pp.50–58. 1997.

[2] G.H. Gonnet, R.A. Baeza-yates, and T. Snider, "New Indices for Text: Pat Trees and Pat Arrays," *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, Chapter 5, pp.66–82. 1992.

[3] D. Gusfield, *Algorithms on strings, trees, and sequences*, Cambridge. 1997.

[4] C.-N. Hsu, M.-T. Dung, "Generating finite-state transducers for semistructured data extraction from the web," *Information Systems.* 23(8). pp.521–538. 1998.

[5] S. Kurtz, C. Schleiermacher, "REPuter: fast computation of maximal repeats in complete genomes," *Bioinformatics*, Vol. 15, No. 5, pp. 426–427. 1999.

[6] N. Kushmeric, D. Weld, and R. Doorenbos, "Wrapper induction for information extraction," In *Proceedings of the 15th International Joint Conference on Artificial Intelligence* (IJCAI). 1997.

[7] I. Muslea, S. Minton, and C. Knoblock, "A hierarchical approach to wrapper induction," In *Proceedings of the 3rd International Conference on Autonomous Agents* (Agents'99), Seattle, WA. 1999.

[8] Overview of all HTML elements, *http://www.htmlhelp.com/reference/wilbur/overview.html*