SEMI-STRUCTURED INFORMATION EXTRACTION APPLYING AUTOMATIC PATTERN DISCOVERY

Chia-Hui Chang, Shao-Chen Lui, and Yen-Chin Wu
Dept. of Computer Science and Information Engineering
National Central University, Chung-Li, 320, Taiwan
Email:chia@csie.ncu.edu.tw, {anyway, trisdan}@db.csie.ncu.edu.tw

Abstract

Information extraction (IE) from semi-structured Web documents is a critical issue for information integration systems on the Internet. Previous work in *wrapper induction* aim to solve this problem by applying machine learning to automatically generate extractors. For example, WIEN, Stalker, Softmealy, etc. However, this approach still requires human intervention to provide training examples. Hence, the other track to information extraction tries to save human effort. For example, Embley et. al. and Chang et al. present different approaches to record boundary identification of a single Web pages without any training example. Embley's work relies on the intra-page structure constructed by HTML tags (the parse tree), while Chang's work is motivated by repeated patterns formed by multiple aligned records. This paper expands Chang's work to IE and discuss the issues when applying pattern discovery for record identification, including the encoding schemes of HTML and ranking criteria of patterns to extract record boundary.

1 Introduction

Information extraction is a key enabling technology for information integration systems on the Internet. To integrate information from heterogeneous Web sites, information mediators must interpret Web pages as structured database-like knowledge sources. Contrast to "traditional" information extraction [3], which roots in natural language processing techniques such as linguistic analysis, Internet information extraction rely on syntactic structures identification marked by HTML tags. The difference is due to the nature of Web such that the page contents have to be open-and-shut for browsing. Thus, "itemized list" and "tabular format" have been the main presentation style for Web pages on the Internet. This is especially true for the e-commerce era when all the pages are attached with myriads of advertisements for business consideration, while the query results or main content that are intended for integration are structured regularly for easy browsing. Indeed, semi-structured information or data that are presented in regularity often constitutes a Web page's core semantic content.

To automate the construction of extractors, recent research has identified important wrapper classes and induction algorithms. For example, Kushmerick et al. identified a family of wrapper classes and the corresponding induction algorithms which generalize from labeled examples to extraction rules [12]. More expressive wrapper structure are introduced lately. *Softmealy* by Hsu and Dung [8] uses a wrapper induction algorithm to generate extractors that are expressed as finite-state transducers. Meanwhile, Muslea et al. [13] proposed *"STALKER"* that generates wrappers based on a set of disjunctive landmark automata organized as a hierarchy.

In all this 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, researchers are exploring new approaches to fully automate wrapper construction. For example, Embley et. al. describe a heuristic approach to discover record boundaries in Web documents by identifying *candidate separator tags* using five independent heuristics and selecting a consensus separator tag based on a heuristic combination [5]. However, the combination is based on the confidence measure of the given independent evidence; and one of the five heuristics is the ontology matching which requires the specific domain knowledge. Consequently, it is hard to say this approach is one hundred percent automatic.

On the other hand, our work here attempts to eliminate human intervention by pattern discovery. As we observe, the syntactic regularity often forms repeated patterns which can be discovered by sequential pattern mining technologies. To enable pattern discovery, we utilizes a data structure called a PAT tree [6] in which repeated patterns in a given input string can be

<HTML><TITLE><Text></TITLE><BODY>
<B>Congo</B><I>242</I>
<B>Egypt</B><I>20</I>
<B>Belize</B><I>501</I>
<B>Spain</B><I>34</I>
</BODY></HTML>

Figure 1: Sample HTML page

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 an example showing the repeated pattern formed by multiple aligned records. We then describe the data structures that are used for pattern discovery and discuss what kind of patterns can be discovered in section 3. In section 4, various issues regarding pattern formulation are considered such as encoding schemes of HTML and ranking criteria of pattern validation. Section 5 describes the performance measures for extraction and reports experimental results of pattern ranking and various HTML encoding. The last section presents our conclusion and the directions of future work.

2 Motivation

One observation from Web 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 example, query-able or search-able Internet sites such as Web search engines often produce Web pages with large itemized matches which are displayed in a template format. The template can be recognized when the content of each match is ignored or replaced by some fixed-length string. Therefore, repetitive patterns are formed.

For instance, in the example given by Kushmerick in [12] (presented in Figure 1), the sequence "<B>Alph</B><I>Num</I>< BR>" is repeated four times, when text strings "Congo", "Egypt", "Belize" and "Spain" are replaced by token class *Alph*, and number string "242", "20", "501" and "34" are replaced by token class *Num*.

This is a simple example that demonstrates a repeated pattern formed by tag tokens in a Web page following a simple translation convention. In practice, many search-able Web sites also exhibit such repeated patterns since they usually extract data from relational database and produce dynamic Web pages with a predefined format style. Therefore, what we ought to do is kind of reverse engineering to discover the original format style and the content we need to extract. Remember that HTML tags are the basic components for data presentation and the text string between tags are exactly what we see in the browsers. Hence, it is intuitive to regard the text string between two tags as one unit as well as each individual tag. This is a simple version of *HTML translation* that we will use in the following paper where any text string between two tags is translated to one unit called Text(_) and every HTML tag is translated to a token Html(<tag>) according to its tag name.

Such translation convention enable the show-up of many repeated patterns. By *repeated patterns*, we mean any substring that occurs twice in the encoded token string. Thus, not only the sequence "Html(<B>) Text(_) Html(</B>) Html(<I>) Text(_) Html(</I>) Html(
)" conforms to the definition of repeated pattern but also the subsequence "Html(<B>) Text(_) Html(</B>)," "Text(_) Html (</B>) Html(<I>)", "HMLT(<I>)Text(_)Html(</I>)," etc. To distinguish from these repeats, we define *maximal repeats* to uniquely identify the longest pattern as follows.

Definition Given an input string S, we define *maximal repeat* α as a substring of S that occurs in position $p_1, p_2, \ldots, p_k$ in S such that $p_i \neq p_j$ and the $(p_i - 1)$th character in S is different from the $(p_j - 1)$th character and the $(p_i + |\alpha|)$th is different from the $(p_j + |\alpha|)$th character for at least one i, j pair.

The definition of maximal repeats is necessary for identifying the well-used and popular term, repeats. Besides, it also captures all interesting repetitive structures in a clear way and avoids generating overwhelming outputs. In summary, we start from the observation that search-able Internet sites often produce information that are presented in an itemized style or tabular format such that the information to be extracted forms certain kind of patterns after proper translation of its text content. Meanwhile, we also find that repeats that occur regularly and closely in a Web page usually correspond to segments of interesting information that might be the *core semantic content* (or the main information block as defined in [5]), i.e., the target block to be extracted. These observations motivate us to look for an approach to discover repeated patterns. If such patterns can be successfully identified, we can then use them as extraction rules to identify the target information fragment.

3 Finding Repeated Patterns

In this section, we introduce a data structure called PAT trees to recognize repeated patterns in a given character string. The input character string here is the translated Web pages and the output is *maximal repeats* defined in the last section.

3.1 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*) [6]. A Patricia tree is a particular implementation of a binary (0,1) 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* [7], 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 character string with n indexing point, there will be n external nodes in the PAT tree and $n-1$ internal nodes. This makes the tree $O(n)$ in size.

When a PAT tree is to index a sequence of characters not just 0 or 1, 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, and the i-th suffix of S starts at $[i * l + 1]$th bit to the end for $i = 0, \ldots, n-1$. The constructed PAT tree $\mathcal{T}$ will have n external nodes pointing to sistrings numbered $1, \ldots, n$.

Back to our application, the translated Web page in our example is regarded as a sequence of *tokens* including HTML tags and a special Text(_) token (which denotes the text string between two tags). With a total of 35 tokens in our example in Figure 1, there will be 35 corresponding sistrings to be indexed. Figure 3.1 shows the first six sistrings where each tag is replaced by its corresponding tag class HTML(<tag name>). Since the number of tag classes are counted up to 170 for now (including *start tags* and *end tags*), each token class is encoded as 8-bit long binary code for future extension.

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. [6, 7]. Since every internal node in a PAT tree indicates a branch, it implies 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. However, not every path-label or repeated sequence represents a maximal repeat. Let's call character $(p_1 - 1)$ of S the *left character* of sistring p_1. For a path-label of an internal 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 v *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 based on the following lemma.

Lemma The path labels of an internal node v in a PAT tree $\mathcal{T}$ is a *maximal repeat* if and only if v is left diverse.

The essence of a PAT tree is a binary suffix tree, which has also been applied in several research field for pattern discovery. For example, Kurtz and Schleiermacher have used suffix trees in bioinformatics for finding repeated substring in genomes [10]. Research in sequential pattern mining, especially Web log mining, have also utilized data structures similar to suffix tree for finding user browsing patterns, [14, 15]. As for PAT trees, they have been applied for indexing in the field of information retrieval since a long time ago [6]. It has also been used in Chinese keyword extraction [1] for its simpler implementation than suffix trees and its great power for pattern discovery. However, in the application of information extraction, we are not only interested in repeats but also repeats that appear regularly in vicinity.

By recording the frequency counts and the reference positions in the nodes of a PAT tree, we can easily know how many times a pattern is repeated. However, there could be more than one maximal repeat pattern in one Web page and not every one corresponds to an interesting text fragment to us. For example, patterns that occur only twice in a Web page or patterns that occur far across a Web page are less intriguing than patterns that occur 10 times in vicinity. Practically, what we are interested are repeats that occur regularly and closely in a Web page. Hence, repeated patterns have to be further validated or compared to find the best one that corresponds to the information to be extracted.

sistring 1: Html(<HTML>)Html(<TITLE>)Text(_)Html(</TITLE>)Html(<BODY>)...
sistring 2: Html(<TITLE>)Text(_)Html(</TITLE>)Html(<BODY>)Html(<B>)...
sistring 3: Text(_)Html(</TITLE>)Html(<BODY>)Html(<B>)Text(_)Html(</B>)...
sistring 4: Html(</TITLE>)Html(<BODY>)Html(<B>)Text(_)Html(</B>)Html(<I>)...
sistring 5: Html(<BODY>)Html(<B>)Text(_)Html(</B>)Html(<I>)Text(_)Html(</I>)...
sistring 6: Html(<B>)Text(_)Html(</B>)Html(<I>)Text(_)Html(</I>)Html(
)...
...
sistring 13: Html(<B>)Text(_)Html(</B>)Html(<I>)Text(_)Html(</I>)Html(
)...
...
sistring 20: Html(<B>)Text(_)Html(</B>)Html(<I>)Text(_)Html(</I>)Html(
)...
...

Figure 2: The Sistrings of Figure 1 to be indexed.

4 Pattern Validation Criteria

In the above section, we discussed how to find maximal repeats in a PAT tree. Once the PAT tree is constructed, we can easily traverse the tree to find all maximal repeats given the expected pattern frequency and length. However, this is not the end of the story. The number of discovered maximal repeats may be more than sixty, and which of them corresponds to the core information block? What other characteristics of patterns can be used to filter or select maximal repeats? Recall that maximal repeats only explain the idea of repeat. The main property that wrappers rely on, i.e., regularity, has not yet been applied. In this section, we quantify several criteria to measure the quality of a maximal repeat. Let the sistrings of a maximal repeat α are ordered by its position such that sistrings $p_1 < p_2 < p_3 \ldots < p_k$, where p_i denotes the position of each sistring in the encoded token sequence. The criteria include regularity, locality, vicinity, and coverage as described below.

Regularity Regularity of a pattern is measured by computing the standard deviation of the interval between two adjacent occurrences $(p_{i+1} - p_i)$. Let $\sigma(\alpha)$ returns the standard deviation of this interval and $\mathcal{M}(\alpha)$ returns the interval's mean for maximal repeat α. The regularity of a maximal repeat α is computed as follows:

$$R(\alpha) = \frac{\sigma(\alpha)}{\mathcal{M}(\alpha)} \tag{1}$$

Locality This property is required to avoid extracting repeats that are scattered too far across the input. We measure the degree of locality of a maximal repeat through the computation of density:

$$\mathcal{D}(\alpha) = \frac{k * |\alpha|}{p_k - p_1 + |\alpha|} \tag{2}$$

Vicinity Vicinity is a special criterion that is designed to deal with "redundant" maximal repeats. Consider the example we mentioned above where sequence α = "Html(<B>)Text(_) Html(</B>) Html(<I>) Text(_) Html(</I>) Html(
)" is repeated four times (see Figure 1 for illustration). In such cases, not only α, but also $\alpha\alpha$ and $\alpha\alpha\alpha$ are qualified for regular maximal repeats. Hence, we define vicinity as

$$\mathcal{V}(\alpha) = \frac{|\alpha|}{\mathcal{M}(\alpha)} \tag{3}$$

Coverage Coverage measures the volume of content a maximal repeat contains. Suppose the function $\mathcal{P}(i)$ returns the position of the i-th sistring in the original Web page, i.e. the HTML file. We calculates coverage by the spread of a maximal repeat in the page as follows:

$$\mathcal{C}(\alpha) = \frac{\mathcal{P}(p_k + |\alpha|) - \mathcal{P}(p_1)}{|Webpage|} \tag{4}$$

If all occurrences in the subtree of a maximal repeat appear spaced at an interval of approximate equal distance, the regularity will be close to zero. If these occurrences are further aligned close, the density will be close to one. However if there are overlaps between two adjacent occurrences, the vicinity will be grater than one. To avoid output multiple maximal repeats which originate from the same pattern sequence, vicinity is required to be less than two. In addition to the above criteria, we can also measure the pattern length ($|\alpha|$) and the occurrence frequency counts $\mathcal{F}(\alpha)$ for each maximal repeat to predict whether it corresponds to the core information block we would like to extract. For each criteria, a threshold is set as a minimum requirement. If the threshold is set lower for regularity, then only few maximal repeats with regularity smaller than

Web site	sistrings	maximal	regularity	vicinity	locality
AltaVista	1535	127.6	44.8	11.4	6.8
Cora	799	75.1	31.6	24.5	10.3
Excite	1740	65.2	9.2	2.2	1.0
Galaxy	611	55.6	23.0	19.0	12.8
HotBot	1455	38.1	17.6	12.8	12.7
Infoseek	987	77.8	15.6	15.3	11.1
Lycos	629	121.0	10.8	10.1	7.4
Magellan	1375	26.0	9.3	4.3	1.0
MetaCrawler	382	142.9	31.5	25.7	15.4
NorthernLight	1144	46.9	16.0	5.5	4.2
OpenFind	1144	48.0	25.3	18.1	7.8
SavvySearch	1127	71.2	32.0	15.0	12.8
StptCom	1777	56.5	37.5	4.0	4.0
Webcrawler	1092	69.1	19.3	5.4	4.1
Average	1128	72.9	23.1	12.4	8.0

Table 1: No. of maximal repeats found for each Web site (averaged from 10 test pages). The right block shows the number of maximal repeats which pass the validation criteria regularity, vicinity, locality in turn.

the threshold will be qualified for our selection. Similarity, if the threshold is set higher for locality, then only few maximal repeats with locality higher than the threshold will be qualified for our selection. The upper bound for Vicinity is two, while coverage is an index for the richness of a pattern. Additionally, the quantitative characteristics are then used to rank the maximal repeats as discussed below.

5 Empirical Results

To demonstrate the effect of our extraction procedure, we choose fourteen Web sites and use the returned pages of 10 queries as the test pages for each Web site. The system only output patterns with length at least 5 tokens long, occurring at least 4 times. This thresholds are chosen because of the application domain we are involved, that is, search engines where each query contains ten or more matches in one page. Besides, we do not want to miss any possible patterns.

Table 1 shows the the number of maximal repeats identified for each Web site applying the three validation criteria: regularity, vicinity and locality. The "*sistring*" column shows the number of sistrings indexed in the PAT tree. The "*maximal*" column represents the number of maximal repeats discovered from the PAT tree. The following column shows the number of maximal repeats remained after applying the criteria for regularity, vicinity and locality in turn. Note that the order of the application of these criteria does not affect the final result. The thresholds for regularity and locality are 0.5 and 0.25, respectively. Other results with different threshold setting can be found in [2]. The thresholds are chosen because the number of maximal repeats remained varies slightly for regularity smaller than 0.5 and locality higher than 0.25.

From Table 1, we can see that regularity and locality both play an important role in filtering maximal repeats. Vicinity and locality are coincident with each other, thus can be combined. Of the fourteen search engines, Magellan shows the simplest case when only one maximal repeat remained after the validation procedure; while the AltaVista example demonstrates a general case that several maximal repeats remained after the validation procedure. Generally speaking, search engines require a "while loop" to output their results in some template. However, they may use "if clauses" inside the loop to decorate the content. For example, the keywords that are submitted to search engines are shown in bold face for Infoseek and MetaCrawler, thus, breaking their "while loop" patterns. This is why there are more than one maximal repeat found and the difficulty of information extraction based on pattern discovery. Hence, heuristic analysis is required to rank among maximal repeats to filter the good ones.

5.1 Evaluating Maximal Repeats

To evaluate the identified maximal repeats, two metrics are proposed: the *retrieval rate* and the *accuracy rate*. The retrieval rate of a maximal repeat is defined by the ratio of the number of tuples *enumerated* by a maximal repeat to the number of matches returned by the query-able Web site, where a tuple is said to be enumerated by a pattern if the *overlapping percentage* between the record and pattern is greater than θ. While the accuracy rate is defined as the ratio of the number of enumerated tuples to the pattern's occurrence count. For example, suppose a query-able Web site contains 10 matches in one response, and a pattern

enumerates 7 of the matches in the pattern's 8 occurrences with the given θ. Then, the retrieval rate is 7/10 and the accuracy is 7/8. Because of the "if-effect" not every record was displayed in the same way. If the enumeration threshold θ is set higher, few records can be enumerated thus retrieval rate is low. If the threshold is set lower, each record is only partially matched thus the accuracy rate is low.

To automate the computation of retrieval rate and accurate rate, we use Softmealy to record the positions of each tuple for all test pages and compare them to the positions recognized by a pattern. Since PAT trees are confined to discover exact match, it is not easy for the main information block to be captured in a maximal repeat unless a good encoding scheme is used. Therefore, most maximal repeats discovered only enumerate or overlap a part of the record to be extracted; and we have to use both accuracy and retrieval rate to measure the performance of a pattern. In addition to the binary decision whether a record is enumerated, the overlapping percentage of all records are averaged as *matching percentage* for a pattern.

Take Webcrawler as an example, for each pattern we compute the retrieval rate, accuracy rate and matching percentage as the performance index. In Table 2, five maximal repeats are validated with regularity threshold 0.5 and locality threshold 0.25. The retrieval rate and accuracy are computed using matching threshold $\theta = 0.5$. Other measures such as regularity, locality, vicinity, coverage, the pattern's length, occurrence count are also shown for reference. Averaging the three measures of the best patterns (with the highest matching percentage) for each test page, the results are shown in Table 3. The retrieval rate and accuracy are 0.73 and 0.81, respectively; while matching percentage is 0.60 in average.

Web site	Retrieval Rate	Accuracy Rate	Matching Percentage
AltaVista	1.00	1.00	0.82
Cora	0.63	0.87	0.48
Excite	0.30	0.30	0.30
Galaxy	0.65	0.83	0.50
HotBot	0.90	0.91	0.61
Infoseek	0.81	0.91	0.58
Lycos	0.65	0.79	0.49
Magellan	1.00	1.00	0.76
Metacrawler	0.47	0.67	0.39
NorthernLight	0.94	0.96	0.87
OpenFind	0.21	0.48	0.26
SavvySearch	0.70	0.83	0.57
Stpt.com	0.99	1.00	0.70
Webcrawler	0.98	0.98	0.98
Average	0.73	0.81	0.60

Table 3: Retrieval rate and accuracy rate given matching threshold 0.5.

Web site	# Pattern	Rank	Matching
AltaVista	2.6	1.0	0.82
Cora	3.0	1.5	0.39
Excite	0.6	0.6	0.30
Galaxy	1.2	1.0	0.46
HotBot	1.0	1.0	0.60
Infoseek	2.0	1.0	0.59
Lycos	1.0	1.3	0.47
Magellan	1.0	1.0	0.76
MetaCrawler	1.4	1.1	0.42
NorthernLight	1.0	1.0	0.66
OpenFind	1.0	1.0	0.26
SavvySearch	1.1	1.4	0.52
Stpt.com	2.2	1.0	0.70
Webcrawler	1.0	1.0	0.98
Average	1.43	1.1	0.57

Table 4: The matching percentage for selected patterns after ranking.

5.2 Ranking Among Patterns

Due to the semi-structured nature of Web pages and hence not one hundred percent regular display format, sometimes there is no perfect pattern can represent the text fragment we want to extract; and the PAT-tree based pattern extractor might output several patterns for options. Examine these patterns, one typical situation is that the longer the maximal repeat the less frequent it occurs. To a certain degree, this is just like a tradeoff between retrieval rate and matching percentages. Thus, we adopt a multi-parameter ranking algorithm to choose the best maximal repeats. The maximal repeats validated are sorted according to three functions: $f_1 = coverage$, $f_2 = regularity$ and $f_3 = locality$ in turn.

Immediately after each sorting function, a threshold t_i is computed to divide the patterns into superior patterns and inferior patterns, where superior patterns (denoted by S_i) are those with function value f_i greater than t_i, while inferior patterns (denoted by T_i) are those with function value f_i less than the threshold ($i = 1, 3$). For f_2, the superior patterns are those with regularity f_2 less than t_2. Only superior patterns are kept for sorting in the next run. The threshold is decided dynamically by one of the function values which maximize the difference of the corresponding average function values between the superior set and those of the inferior set.

Since the retrieval rate changes with the matching threshold used, we use matching percentage for the following comparison. The extraction performance after ranking are summarized in Table 4. The *# Pattern*

pattern	regularity	vicinity	locality	coverage	length	count	retrieval rate	accuracy	matching percentage
1	0.05	0.82	0.83	0.34	10	20	0.60	0.60	0.48
2	0.04	0.41	0.43	0.33	5	20	0.35	0.35	0.38
3	0.00	1.42	1.38	0.30	17	18	0.90	0.95	0.90
4	0.04	1.24	1.22	0.32	15	19	0.90	1.00	0.88

Table 2: The retrieval rate and accuracy of validated maximal repeats for a test page along with other measures.

column shows the number of patterns selected after ranking. The "*rank*" column shows the best pattern's position after our ranking scheme. Best matching percentage of the average 1.43 patterns are 0.57 which is close to that of 8.0 patterns. From Table 4, we can see that most of the best patterns after ranking are ranked first (ranked at 1.1 in average).

5.3 Other Translation Convention

Since the results of any unsupervised technique will depend tremendously on which features are used, in this section, we try different encoding schemes of HTML files to find better patterns. According to the HTML structure tree (Figure 3) [16], the tags in the body section of a document can be grouped in two distinct groups: *block level* tags and *text level* tags [16]. The former make up the document's structure, and the latter "dress up" the contents of a block. As shown in Figure 3, 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.

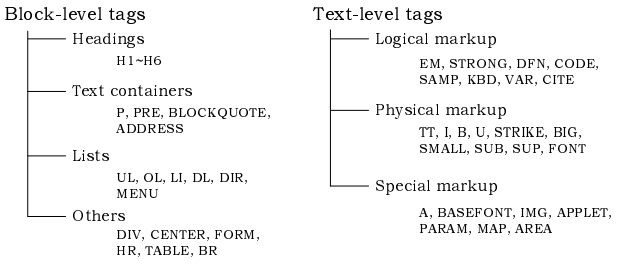

Figure 3: Block-level tags vs. text-level tags

The experiments in the above section present the encoding scheme when all tag classes are involved in the translation (each tag is translated to their corresponding token class). In this section, four encoding scheme are conducted, which skip logical, physical, special and all text-level tags respectively. For example, the second encoding scheme skips physical markups, including <TT>, <I>, <B>, and <U>, etc. Table 5 shows the effect of the last three encoding schemes. The results of the first encoding scheme are not shown because logical markups are less used in HTML files (only 0.4%) and the difference is not obvious. For the second encoding scheme, the performance are increased for Infoseek, Lycos, MetaCrawler, and SavvySearch; while for the third encoding scheme, the matching percentage increased for AltaVista, Cora, and OpenFind. That is, the encoding scheme of skipping some markups/tags though enables the display pattern for some search engines, it may disables the patterns of other Web sites, especially when the pattern gets shower than five tokens. However, high-level abstraction have better performance in average. We conclude that the block-level encoding scheme performs best among others. In addition, the token string for block-level encoding scheme is only two percent of the original HTML file (average 22.7 KB).

Encoding	Retrieval Rate	Accuracy Rate	Matching Percentage	string length
Alltag	0.73	0.81	0.60	1128
NoSpecial	0.82	0.88	0.68	873
NoPhysical	0.84	0.88	0.70	796
Block-level	0.86	0.86	0.78	514

Table 5: Performance comparison for different encoding schemes.

6 Summary and Future Work

Information extraction from Web pages is a core technology for comparison-shopping agents [4], which Doorenbos et al. regard as improvement in the axe of tolerating unstructured information. The three measures regularity, navigation, uniformity, and vertical separation, enable the possibility of learning and Doorenbos et al. delineate the framework for such agents. However, unsupervised learning in their paper is realized through searching a space of possible abstract formats and other assumptions such as every tuple starts on a fresh logical line, etc. Later researches focus on supervised learning approaches and coin the

term wrapper induction and systems such as WIEN [12], Softmealy [8], Stalker [13].

In this paper, we present an unsupervised approach to semi-structured information extraction. We remove the limitations in Doorenbos's agents by recognizing repeated patterns in the encoded HTML files. To summarize, the information extraction procedure can be briefed by three steps: HTML translation, PAT tree construction, and pattern validation. The necessity of "HTML translation" is to protrude the repeat patterns, while the PAT tree is the most appropriate data structure to facilitate the finding of repeat patterns. Next, the validation criteria: regularity, vicinity, and locality are applied as validation of interesting patterns. Finally, ranking functions are used so that the best maximal repeat can be ranked first.

Currently, the block-level-tag encoding scheme can achieve 0.86 retrieval rate and accuracy rate in average. For nine of the fourteen Web sites, the retrieval rate are greater than 0.95. In future work, since PAT trees are confined to find exact patterns, the performance is limited for Web pages that involve exceptions in their display format. In such cases, other techniques are considered to further improve the performance. For example, dynamically choosing an encoding scheme that best fits an input Web page or constructing patterns in regular expression form from discovered patterns, etc.

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] Chien, L.F. 1997. 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] Chang, C.-H.; and Hsu, C.-N. 1999. Automatic extraction of information blocks using PAT trees. In *Proceedings of National Computer Symposium*, Taipei, Taiwan.

[3] Cowie, J. and Lehnert, W. 1996. Information extraction. *Communication of ACM* 39(1):80–91.

[4] Doorenbos, R.B., Etzioni, O. and Weld, D. S. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the first international conference on Autonomous Agents*. pp. 39–48, NewYork, NY, 1997, ACM Press.

[5] Embley, D.; Jiang, Y.; and Ng. Y.-K. 1999. Record-boundary discovery in Web documents. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*. pp. 467–478, Philadelphia, Pennsylvania.

[6] Gonnet, G.H.; Baeza-yates, R.A.; and Snider, T. 1992. New Indices for Text: Pat Trees and Pat Arrays. *Information Retrieval: Data Structures and Algorithms*, Prentice Hall.

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

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

[9] Knoblock, A. et al., eds., 1998. *Proc. 1998 Workshop on AI and Information Integration*, Menlo Park, California.: AAAI Press.

[10] Kurtz, S. and Schleiermacher, C. 1999. REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics* 15(5):426–427.

[11] Kushmerick, N. 1999. Gleaning the Web. IEEE Intelligent Systems 14(2):20-22.

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

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

[14] Pei, J.; Han, J.; Mortazavi-asl, B.; and Zhu. H. 2000. Mining access patterns efficiently from Web Logs. In *Proceedings of 2000 Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD'00), Kyoto, Japan.

[15] Spiliopoulou. M.; and Faulstich. L. 1998. WUM: A tool for Web utilization analysis. In *Proceedings of the 6th International Conference on Extending Database Technology* (EDBT'98), Valencia, Spain.

[16] Web Design Group. 1997. Wilbur – HTML 3.2 http://www.htmlhelp.com/refer-ence/wilbur/