Multi-level Alignment for Attribute Extraction in IEPAD

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

Abstract

The problem of information extraction (IE) regards automatic generation of extraction programs (also called wrappers). Similar to compiler generator, the core problem is to generate extraction rules. In this paper, we introduce IEPAD (an acronym for Information Extraction based on PAttern Discovery), a system that generalizes extraction patterns from Web pages without user-labeled examples. The system includes a rule generator, which applies sequence mining techniques to discover possible patterns, a rule viewer, which provides an interface for users to see what each pattern can extract, and an extractor, which extracts information from Web pages based on designated extraction rules. To allow finer extraction, multi-level analysis as well as the alignment of multiple records are adopted for attribute extraction. This new track to IE involves no human effort (to label examples) and content-dependent heuristics. Experiments show that it can achieve 96% retrieval rate with only one training example over 14 popular Web search sites. Among them, ten are able to achieve 100% extraction with less than five training pages.

1 Introduction

The best thing about the WWW is that it provides an easy way to publish and get information on the Internet. In the past few years, the increasing number of data sources that can be queried across the WWW has made it a medium for information exchange and integration. Therefore, it promises better Web services to be created from the existing ones. For example, Web information integration systems like meta-search engines [Selberg and Etzioni, 1995] or shopping agents [Doorenbos et al., 1997], are such examples that provide distilled information for users. We can expect that more and more Web services will be created through data integration on the Web.

However, current Web accessible data sources are not designed for application programs, which expect structured data. Typically, the information to be relevant is embedded in a format to be displayed for users. Hence, there presents a special need for *wrappers* to extract the desired information from the machine-generated Web pages. Contrast to *traditional* information extraction which roots from linguistic analysis [Soderland, 1997], the wrapper generation field relies on structure identification marked by HTML tags (see [Kushmerick, 1999, Muslea, 1999] for a survey). The

markups in Web pages together with the multiple tuples contained in one document contribute the so called semi-structured documents.

The major challenge of IE is the problem of scalability since the extraction rules must be tailored for each particular data source. The key component of wrappers is the extraction rules that is used to extract information relevant to a particular extraction task. As writing extraction rule is a difficult, time-consuming task, several research efforts have focused on learning the extraction rules from training examples provided by the user. For example, Kushmerick et. al. identified a family of wrapper classes including *LR*, *HLRT*, *OCLR*, etc. [Kushmerick et al., 1997]. More expressive wrapper structure are introduced by Hsu and Dung who use a finite-state transducer as the architecture for the extractor [Hsu and Dung, 1998]. Meanwhile, Muslea et al. proposed *"STALKER"* that generates single-slot extraction rules and performs hierarchical information extraction with extra scans over the documents [Muslea et al., 1999].

Basically, these researches exploited machine learning techniques to the limits and create tools with very expressive power. However, acquiring these training examples requires labeling which is still time-consuming. Hence, another track of research tries to explore new approaches to fully automate extraction rule generation. 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 choosing a consensus separator tag based on a heuristic combination [Embley et al., 1999].

On the other hand, Chang et. al. took a totally different approach which is based on repetitive pattern discovery [Chang et al., 2001]. Contrary to the idea of the above systems which use markups and system-defined tokens as landmarks or delimiters to recognize boundaries of a record or attribute, this approach relies on the discovery of the display pattern from the abstracted input. Most of all, this pattern discovery based approach involves no human effort and content-dependent heuristics making it superior to other approaches.

However, the work in [Chang et al., 2001] only shows the extraction of the record boundary. In this paper, we further extend the work and describe the extraction of attribute values within each record. In section 2, we briefly describe Chang's work based on pattern mining. Next, the architecture of this wrapper generation system, IEPAD, is described to show how extraction rules can be specified to extract attribute values in each record. Section 4 shows the experimental results and section 5 concludes the paper.

2 Rule Generator for Record Boundary

The approach in Chang's paper [Chang et al., 2001] differs from that of other wrapper induction systems in the very first place where the extraction rules are *context-based* instead of *delimiter-based*. The reason we use context-based rule instead of delimiter-based rule is that the data to be extracted are often generated based on some predefined HTML templates (e.g. job postings, flight schedules, query results from search engines, etc.). This naturally inspires the idea to discover such templates (or patterns) since we do observe that most information we desire is aligned *regularly* and *contiguously*. Such characteristics enable the application of repetitive pattern mining to automate the generation of extraction patterns which correspond to the display templates. In other words, the task of extraction rule discovery can be solved by repetitive pattern mining without user-labeled training examples that are required for prior wrapper induction systems.

In [Chang et al., 2001], a data structure called a *PAT tree* [Gonnet et al., 1992] is utilized for repeat discovery. Once all repetitive patterns are found, we can then check if the pattern occurs *regularly and contiguously*. Finally, the filtered repeats are examined to compose extraction patterns by advanced technique called *multiple string alignment*.

2.1 Maximal Repeats Discovery

By repetitive pattern, we generally mean any substring that occurs at least twice in a string. To better capture the idea of repeats and also reduce the number of patterns discovered, the concept of *maximal repeats* is used to refer to the longest repetitive patterns. The idea is to extend a repeat in both directions to its longest. We call a repeat *left maximal* (right maximal) if the repeat can not be extended on the left (right) direction (See [Chang et al., 2001]). We say a repeat is maximal if it is both left maximal and right maximal. However, repetitive patterns are not easily seen in the plain Web page. To reveal such repetitive patterns in an input Web page, an encoding scheme is used to translate the Web page into a token string of abstract representation. Since HTML tags are the basic components for document presentation and the tags themselves carry a certain structure information, it is intuitive to examine the tag token string formed by HTML tags and ignore text content between two tags to see the display template. Hence, the simplest abstraction is as follows:

1. Each tag is encoded as a tag token <tag_name>
2. Any text between two tags are regarded as a special token called <STRING>

Of course, there are various ways to encode a Web document. With different abstraction mechanisms, different patterns can be produced. For example, HTML tags, according to their functions, can be divided into two distinct groups: block-level tags and text-level tags. The former defines the structure of a document, and the latter defines the characteristics (format and style, etc.) of the text contents. The user can choose an encoding scheme depending on the desired level of information to be extracted. For example, block-level encoding scheme, which is the highest level encoding scheme used in [Chang et al., 2001], has a little modification to rule 1 where only block-level tags are encoded as tag tokens and other text-level tags are simply ignored.

PAT Tree

To automatically discover repetitive patterns, a data structure called *PAT trees* is used to index all suffixes in the encoded token strings. A PAT tree is a Patricia tree (Practical Algorithm to Retrieve Information Coded in Alphanumeric [Morrison, 1968]) constructed over all the possible suffix strings. A Patricia tree is a particular implementation of a compressed binary (0,1) digital tree such that each internal node in the tree shows the different bit between suffix strings in the subtree. To build the encoded token string into a PAT tree, each tag token is denoted by a fixed length binary representation. Like a suffix tree [Gusfield, 1997], the Patricia tree stores all its suffix strings at the external nodes. For a token string with n indexing point (or n suffixes), there will be n external nodes in the PAT tree and $n-1$ internal nodes. This makes the tree $O(n)$ in size.

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 tandem repeats in genomes [Kurtz and Schleiermacher, 1999]. It has also been used in Chinese keyword extraction [Chien, 1997] for its simpler implementation than suffix trees and its great power for pattern discovery.

PAT trees organize input in such a way that all suffixes with the same prefix are put stored the same subtree. Therefore, it has very good characteristics for pattern discovery:

- First, all suffixes in a subtree share a common prefix, which is the *path label* that leads from the tree root to the subtree root.
- Second, the number of leaves in the subtree is exactly the number of occurrence of the path label.
- Third, each path label represents a right maximal repeat in the input.

Therefore, maximal repeat discovery can be achieved by simply traversing the PAT tree to enumerate all path labels to discover all right maximal repeats. As to verify whether a path label is left maximal, we can check the left tokens of the positions where the repeat occurs. If all left tokens are the same, then this repeat can be extended and is not left maximal.

2.2 Pattern Validation Criteria

As described above, most information we want is generated based on some predefined templates and is commonly aligned *regularly* and *contiguously*. To discover these display patterns, two measures, called "variance" and "density", are defined to evaluate whether a maximal repeat is a promising extraction pattern. Let the occurrences of a maximal repeat α are ordered by its position such that $p_1 < p_2 < p_3 \ldots < p_k$, where p_i denotes the position in the encoded token string.

Variance of a pattern is computed by the coefficient of variance of the interval between two adjacent occurrences $(p_{i+1} - p_i)$. That is, the quotient of the standard deviation of the interval to the mean length of the interval.

Density is defined as the percentage of repeats in the interval between the first and the last occurrences of the repeat. That is, $\frac{(k-1)*|\alpha|}{p_k - p_1}$, where $|\alpha|$ is the length of α in number of tokens.

Generally speaking, machine-generated Web pages often embed relevant information in templates which has small variance and large density. To filter potentially good patterns, a simple approach will be to use a threshold for each of these measures. Only patterns with variance less than the variance threshold and density greater than the density threshold are considered validated patterns.

The above approach is easy to implement. However, it can miss some display templates if the variance threshold is not set properly. The reason is that regular patterns can sometimes have large variance coefficient. For example, advertisement banners inserted among the search results can divide the occurrences into several parts like "Lycos" does. These exceptions result in maximal repeats with large variance.

Occurrence Clustering

To handle patterns with variance greater than the specified threshold, the occurrences of a pattern are carefully clustered to see if any partition of the pattern's occurrences can form an independent and regular block. The idea here is to cluster the occurrences into partitions so that we can examine

each partition respectively to see if it is regular enough. For this 1-dimension clustering, a simple loop can accomplish the job. Let $C_{i,j}$ denotes the list of occurrences $p_i, p_{i+1}, ..., p_j$ in increasing order. Initialize i and j to 1. For instance p_{j+1}, if the variance coefficient of $C_{i,j+1}$ is less than γ then p_{j+1} is included as part of the current partition; otherwise, $C_{i,j}$ is exported as a partition and a new partition is created by assigning $j+1$ to i.

Once the occurrences are partitioned, we can then compute the variance for each individual partition. If a partition includes more than the minimum occurrence count and has variance less than threshold v_ϵ, the pattern as well as the occurrences in this partition are exported. Note that the threshold v_ϵ is set to a small value close to zero to control the number of generated partitions.

2.3 Composing Extraction Patterns

In addition to the large variance, patterns with density less than 1 are another problem we need to deal with. Since PAT trees compute only "exact match" patterns, templates with exception (e.g. emphasizing on keywords) can not be discovered through PAT trees. To allow inexact or approximate matching, the technique for *multiple string alignment* is used to find a good presentation of the critical common features of multiple strings.

Suppose a validated partition has k occurrence, p_1, p_2, ..., p_k in the encoded token string. Let string P_i denote the string starting at p_i and ending at $p_{i+1}-1$. The problem is to find the multiple alignment of the $k-1$ strings $\mathcal{S} = \{P_1, P_2, ..., P_{k-1}\}$ so that the generalized pattern can be used to extract all records we need. For example, suppose "adc" is the discovered pattern for token string "*adcbdadcxbadcxbdadc*". Suppose we have the following multiple alignment for strings "*adcbd*", "*adcxb*" and "*adcxbd*":

$$\begin{array}{cccccc} a & d & c & - & b & d \\ a & d & c & x & b & - \\ a & d & c & x & b & d \end{array}$$

The extraction pattern can be *generalized* as "$adc[x|-]b[d|-]$" to cover these three instances.

Multiple string alignment is a generalization of *alignment* for two strings which can be solved in $O(n * m)$ by *dynamic programming* to obtain optimal *edit distance*, where n and m are string lengths. Extending dynamic programming to multiple string alignment yields a $O(n^k)$ algorithm. Alternatively, an approximation algorithm is available such that the score of the multiple alignment is no greater than twice the score of optimal multiple alignment [Gusfield, 1997]. The approximation algorithm starts by computing the *center string* S_c in k strings $\mathcal{S}$ that minimizes *consensus error*. Once the center string is found, each string is then iteratively aligned to the center string to

construct multiple alignment, which is in turn used to construct the extraction pattern.

For each pattern with density less than 1, the *center star* approximation algorithm for multiple string alignment is applied to generalize the extraction pattern. Note that the success of this technique lies in the assumption that extraction patterns often occur contiguously together. If the multiple alignment results in extraction rules with alternatives at more than k positions, such a pattern is very unlikely to be an extraction rule. Therefore, we set an upper bound that there can be at most k mismatches.

Pattern Rotation

Suppose a generalized pattern is expressed as "$c_1c_2c_3...c_n$", where each c_i is either a symbol or a subset of $\Sigma \cup \{-\}$ containing symbols that can appear at position i. Since c_1 might not be the start of a record, we use a (right) rotating procedure to compare the pattern with the *left string* of the first occurrence p_1 from right to left. The purpose is to find the correct starting position and generate pattern to "$c_jc_{j+1}...c_nc_1...c_{j-1}$". If the last token c_n is a token with no option, and the left character of the first occurrence is the same as c_n, we rotate the pattern to "$c_nc_1c_2...c_{n-1}$". The process continued until the last token has alternative options or the left character of the first occurrence is not the same as the last token; and when the rotation stops, the final pattern is output. Similarly, we use a (left) rotating procedure to compare the pattern with the *right string* of the last occurrence p_k from left to right. If the first token is a token with no option, and the right character of the last occurrence is the same as the first token, we rotate the pattern to "$c_2c_3...c_nc_1$". The process continued until the first token has alternative options or the right character of the last occurrence is not the same as the first token; and when the rotation stops, the final pattern is output. With this pattern rotation, the correct record boundary can be identified.

This technique of pattern rotation is different from that used in [Chang et al., 2001], where a heuristic is used to "guess" possible beginning tags[1]. In summary, we can efficiently discover all maximal repeats (with pattern length and occurrence count greater than default thresholds) in the encoded token string through the constructed PAT tree $\mathcal{T}$. Second, validation criteria by variance coefficient and density control allows filtering of promising patterns. For patterns with large variance, occurrence partition can cluster a pattern into information blocks of interest. As for low density pattern, multiple string alignment is applied to produce more complete extraction pattern based on the assumption of contiguous occurrence.

[1]Record patterns often start with $< DL >$, $< DT >$, $< TR >$, $< P >$, $< LI >$, etc.

3 System Architecture

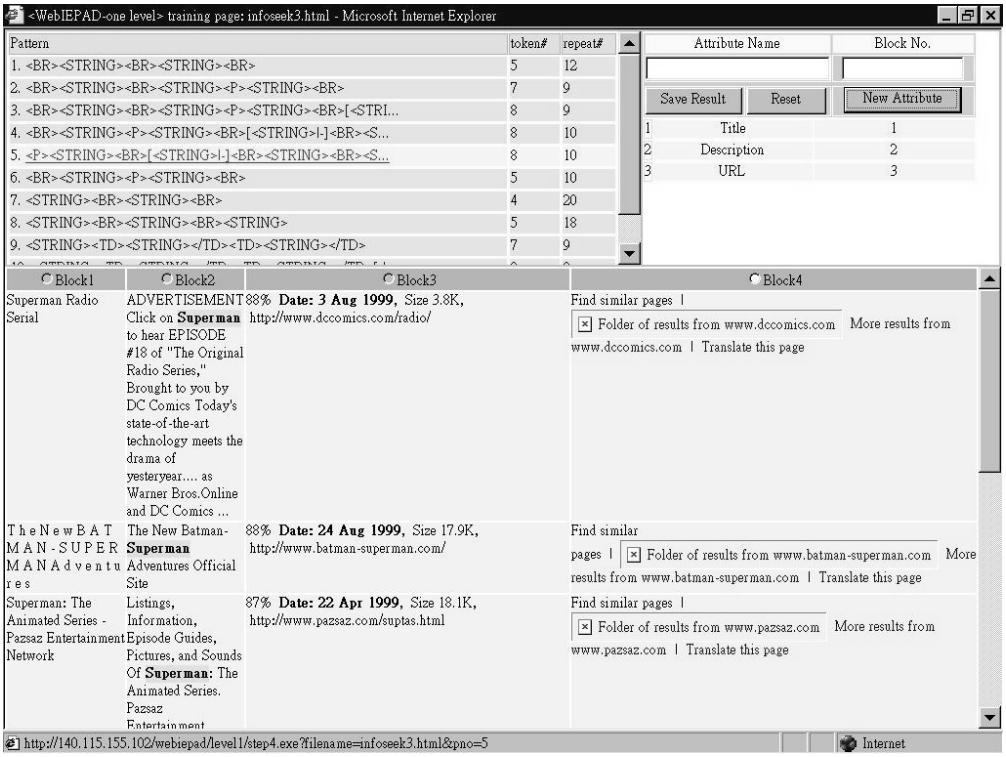
The IEPAD system includes three components: the *rule generator*, the *extractor*, and the *rule viewer*. The core technique is the rule generator described in last section which outputs the extraction patterns constructed. The *rule viewer* provides an interface for the user to view what each pattern can extract and assign one as the *record extraction rule*. As for the *extractor*, it accepts the Web document as input and extract relevant information based on the designated extraction rule. The extractor is actually a string matching algorithm that match all occurrences of the *record extraction rule* in the encoded token string of the Web document. The extractor can be implemented in various languages so that it can be incorporated in different platform.

3.1 Rule Viewer

The purpose of the rule viewer is to provides a graphic user interface so that the user can view the extracted content of each constructed pattern. Since the record extraction patterns are discovered automatically without any prior knowledge from the user and there might be more than one information blocks, an interface is necessary for the user to choose a proper record extraction rule. Note that this selection is different from labeling task of machine learning based research, since the users are not to give labeled example for pattern discovery but to select the pattern discovered. In addition, the user can also adjust the parameters including the encoding schema, the minimum pattern length, the minimum occurrence count, the variance and density thresholds.

Figure 1 shows a snapshot of the rule viewer. The upper-left window shows the patterns discovered and the lower window shows the corresponding information when a pattern is selected. For example, in Figure 1(a), the user chooses the fifth pattern and the corresponding information are shown in the lower window. Note that all the records extracted are further divided into blocks (or slots) and aligned for selection through the upper-left window, where the users can key in the attribute names and select the desired information blocks by clicking the check boxes above each block.

Let us explain how the data are aligned like this. Recall that patterns are composed of tag tokens and text tokens; and only *text tokens* and some special tag tokens (which contain hyperlinks such as <A> and <IMG> tags) might contain useful information. Therefore, whenever there is a text token or special tag tokens in the pattern, the corresponding information should be presented. Since we have recorded the starting position of each token in the Web page during the translation phase, we can always trace the corresponding information of any tokens. Therefore, for each token substring



(a)

(b)

Figure 1: The extraction rules and the user interface (a) training page: infoseek3.html (b)testing page: infoseek4.html

```
Block_division(α, R) {
    for each r_i in R do
        r'_i= Align(α, r_i);
        m= 0;
        for j=1 to |α| do
            if α[j]=<STRING> or <A> or <IMG> then
                m= m + 1;
                if r_i[j]≠'-' then block[i][m]= r'_i[j];
                else block[i][m]= null;
                endif
            endif
        endfor
    endfor
    return block;
}
```

Figure 2: Block division procedure

matched by the record extraction rule, the contents that are encoded as text tokens and the hyperlinks that are embedded in <A> or <IMG> tags can be extracted accordingly. The procedure to divide the set of all records that pattern α can match is outlined in Figure 2. In summary, if there are m text tokens and special tag tokens in a record extraction pattern, all the records can be divided into m blocks. For example, as shown in Figure 1(a), there are four text tokens in the fifth pattern, "<P><STRING>
[<STRING>]
<STRING>
<STRING>"; hence, each extracted record is divided into four blocks.

From the rule viewer, users can also key in the attribute names and select the desired information blocks by clicking the check boxes above each block. For example, in the upper-right corner of Figure 1(a), we have chosen block 1, 2, 3 for "title", "description" and "score", respectively. The saved extraction rule can then be used to extract other Web pages fetched from the same Web site. As shown in Figure 1(b), the testing page is uploaded and the extraction results are shown in the lower window.

Multi-Level Alignment

Although the text tokens can divide the record information into several blocks, this may not be enough. When higher level abstraction, say block-level encoding is used for record pattern discovery, the content in a block may contain not only text but also text-level tags. If we would like to extract "finer" information, the content in each block has to be further processed. Of course, lower-level

```
MultiLevel_alignment(block, Encoding_scheme){
    for j=1 to m do
        for i=1 to k do
            r_i = Encoding_scheme(block[i][j]);
        endfor
        R= {r_1, r_2, ..., r_k};
        β_j= MultipleAlignment(R);
        A_j= Block_division(β_j, R);
    endfor
    A= A_1 + A_2 + ... + A_m;
    return A;
}
```

Figure 3: Multi-level alignment

encoding scheme can be employed to save further process. However, it becomes much difficult to discover and compose the extraction rule for record boundary since the success of the pattern discovery approach lies in good abstraction of the Web pages.

To extract finer contents, a compromised method is to employ *multi-level alignment* and apply block division again. Let's call the encoding for record boundary as the first-level encoding. We will apply a second-level encoding to the text contents in each block and align these encoded token strings for further block division as shown in Figure 3. For example, the text contents that belong to the third block in Figure 4 can be further aligned to a generalized rule "<FONT><STRING><B><STRING></B><STRING></FONT>". With three text tokens inside, it will in turn divide the contents into three sub-blocks, where the first text token corresponds to "score" and the second text token corresponds to "date". The same step can be applied until the desired information can be successfully separated from others. As we shall see in next section, two-level's encoding can extract the target information quite well in our experiments.

3.2 The Extractor

Search engine-like data sources are easier to wrap compared to hand-crafted static pages since these Web pages are produced by programs. For these data sources, we can select one page as input to our system and choose the proper pattern as the extraction rule. Figure 5 shows an example of the extraction rule for the above example. Line one shows the encoding scheme used and the designated record pattern. For each text token in the record extraction rule, the second-level encoding scheme and the aligned regular expression are shown from line two to line four. Following are the attribute

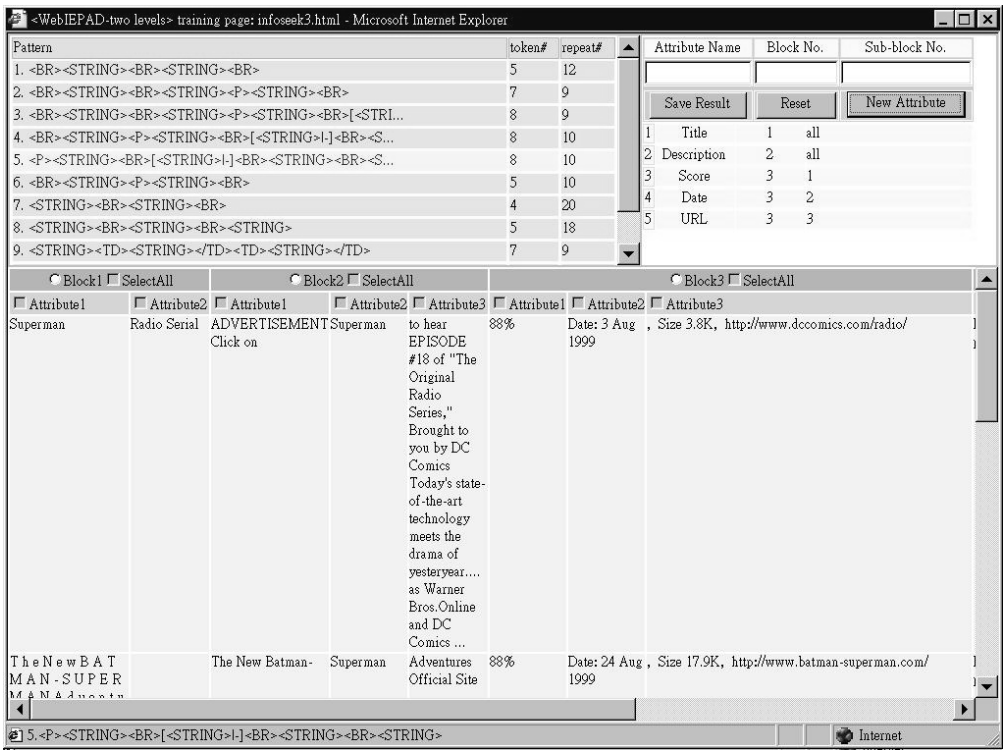


Figure 4: Two level attribute value extraction

names and the information slots specified by the user. For example, the first attribute, "title" contains all text contents in block one. While the third attribute, "score", refers to contents in sub-block one of block three.

With the extraction rule, we can then test it on other Web pages to extract relevant information. According to the extraction rule, the extractor first translate the Web pages into the token string based on the first-level encoding scheme and then match all occurrences of the record extraction pattern in the encoded token string (see Figure 6). Then, the contents corresponding to each text token in the record rule are further translated into token string based on the second-level encoding schemes. The encoded token string is then aligned to the second-level extraction rule. For multi-level extraction, this process may be continued to divide block information into sub-blocks. Finally, the attribute values for each attribute can be extracted according to the block number specified.

The record extraction here is actually a pattern matching algorithm. Typical pattern matching algorithms, like the Knuth-Morris-Pratt's algorithm or Boyer-Moore's algorithm [Gusfield, 1997], both can do the work. Note that each extraction rule composed by multiple string alignment actually represents several patterns. The patterns are expressed in regular expression with alternatives. In other words, there are alternatives routes. Therefore, several patterns can apply when matching

Block-level: *<P><STRING>
[<STRING>]
<STRING>
<STRING>*
Text-level: *<B><STRING>[</B>][<STRING>]</A></B>*
Text-level: *<STRING>[<B>][<STRING>][</B>][<STRING>]*
Text-level: *<FONT><STRING<B><STRING></B><STRING></FONT>*
Title: 1; all
Description: 2; all
Score: 3; 1
Date: 3; 2

Figure 5: Extraction rule example

Extractor(page, rule){
    S= **call** *Encoding*_1(page) to translate page into token string;
    R= **call** Boyer_Moore(S, α) to find all occurrence of pattern α;
    $block_1$= Block_division(α, R);
    for j=2 **to** l **do**
        A= MultiLevel_alignment($block_{j-1}$, *Encoding_j*);
        $block_j$= A;
    endfor
    Extract designated information slots for each attribute;
}

Figure 6: Extractor procedure

the rule against the translated token sequence. In such cases, the longest match is considered.

4 Experimental Results

The experiments here use two test data sets. The first one contains ten state-of-the-art search engines used in [Chang et al., 2001]. However, each data source has much more test pages (200) than used in the previous work. The second data set are Okra, IAF, BigBook, and QuoteServer taken from Kushmerick's thesis. These four sources have also been used in Hsu, et al. and Muslea's paper for purposes of comparison. Table 1 shows the basic description of each data source, including the number of records in each page, the number of attributes in each record, the existence of missing attribute in a record or multiple values for one attribute, and the number of test pages.

The average document size is 28K bytes and 11K bytes for the above two data sets, respectively. The search results of the first data set typically contain at least 10 matches and more advertisements,

Table 1: Data description

Data source	# of records	# of attributes	Missing	Unordered
AltaVista	10	4	Yes	No
DirectHit	10	4	Yes	No
Excite	10	4	Yes	No
Hotbot	10.2	4	Yes	No
Infoseek	15	3	Yes	No
MSN	10	3	No	No
NorthernLight	10	3	Yes	Yes
Sprinks	20	4	Yes	No
Webcrawler	25	3	No	No
Yahoo	20	4	Yes	No
Okra	18.5	4	Yes	No
Bigbook	14.2	6	No	No
IAF	3.7	6	Yes	Yes
QuoteServer	5.9	18	No	No

Table 2: Data size with different encoding schemes

Data Set	Search Engines (Doc size=28K)		Okra, etc. Doc size=11K)	
Encoding	# of tokens	Percentage	# of tokens	Percentage
All-tag	1023	4.0%	584	5.0%
NoPhysical	839	3.0%	473	4.0%
NoSpecial	835	3.0%	447	3.9%
Block-level	639	2.0%	333	3.0%

while test pages for the second data set contain less matches and advertisements. The size of the encoded token string is much smaller than the document size. It's about 5% the page size for the lowest level encoding when all tags are considered (see Table 2). Therefore, the effort to build PAT trees and the tree size can be kept small.

Parameters Setup

The input parameters are set according to the experiments in [Chang et al., 2001]. The principle is to control the number of validated maximal repeats to a level of about 10 patterns so that it is easy to choose from. The parameters include encoding scheme, the minimum pattern length, the minimum occurrence count, the threshold for variance coefficient and density. We have chosen block-level encoding scheme to discover record pattern since it is the highest level abstraction

Table 3: Number of attributes extracted

Data source	default	Level 1	Level 2
AltaVista	4	4	4
Direct Hit	4	3	4
Excite	4	4	4
Hotbot	4	4	4
Infoseek	3	3	3
MSN	3	3	3
NorthernLight	3	3	4
Sprinks	4	3	4
Webcrawler	3	3	3
Yahoo	4	4	4
Okra	4	4	4
Bigbook	6	5	6
IAF	6	1	3
QuoteServer	18	3	18

and was shown to perform best in previous work. The default value for the minimum length of maximal repeats and the minimum occurrence count are set to 3 and 5, respectively. As for variance coefficient and density, the threshold 0.5 and 0.25 are found to filter as many good maximal repeats that can be used to compose record patterns. To give an overview, the number of maximal repeats discovered in a page can be over 100. With the control of variance coefficient and density, the number of maximal repeats remained can be reduced to about 10. However, the number of record patterns can increase due to pattern rotation procedure of the aligned pattern.

Generalizing over Unseen Pages

Although the process for the pattern discovery is not like typical machine learning process, the goal is the same to generalize the extraction over unseen pages. Therefore, we have designed the experiments in a way similar to that used in machine learning. For each Web site, we randomly select 30 pages as the training pages and use remaining as validation set. The rule selected from the training set will be used to extract information from the validation set. The performance is evaluated by two measures: retrieval rate and accuracy rate. Retrieval rate is defined as the ratio of the number of desired data records correctly extracted to the number of desired data records contained in the input text. Likewise, accuracy rate is defined as the ratio of the number of desired data records correctly extracted to the number of records extracted by the rule. Similar to the idea of recall and precision, it is often a tradeoff between retrieval rate and accuracy rate.

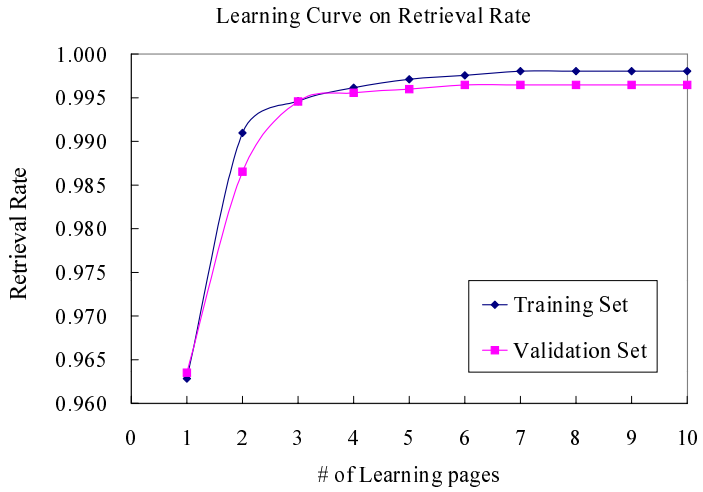


Figure 7: Learning curve on training set and validation set

In the training phase, one page from the training set is fed to the system for the discovery of extraction pattern. The user then choose one (that extracts only correct records) as the record extraction rule and specify the information block for attribute value extraction. This strategy of choosing patterns make sure that accuracy rate is 100%. The extraction rule will then be applied to other training pages to check the performance. If the rule can not extract all the records for a page, the page will then be fed to the system for the discovery of proper extraction rules. Old extraction rule will then be joined with the extraction rule for the second page to form the new extraction rule and applied to extract other pages in the training set. The process goes on to extract as many records as possible. Figure 7 shows the retrieval rate of the extraction patterns joined by several training pages. The number of training pages is 3 in average and 10 at most. The retrieval rate can achieve 96% using the first rule (discovered from one training page). At the use of the second rule (discovered from the mis-extracted page), the retrieval rate has achieved 99%. When five training pages are used, twelve of them are able to achieve 100% extraction except for HotBot and NorthernLight.

Note that the extraction rule we used here is not only able to extract record boundary as in [Chang et al., 2001], but also able to extract attribute values through multi-level expansion. As shown in Table 3, two-level extraction, where the second-level uses All-tag encoding scheme, has

successfully extract all attributes we desired except for IAF. This is because IAF uses <pre> tag to display the detailed information. The second-level extraction here using All-tag encoding can only divide the record into three slots, which needs text-level encoding for further extraction.

Next, in the validation phase, the extraction rule is applied to validation set for testing. Encouragingly, the retrieval rate is still as high as in the training set, which shows that thirty training pages have exhibited enough variety in the display format for the system to generalize. As we can see in Figure 7, the retrieval rate is still very high: 96.35% for 1 training page and 98.65% for 2 training pages. Of the 14 Web sites, 10 of them are able to achieve 100% extraction with less than five training pages in the validation phase.

The learning curves are different from most machine learning-based approach since there are little noise in the training data. We do encounter errors in the data, say, breaking tags, etc. But the percentage is low. In addition to the accuracy strategy, we can also draw learning curves with higher retrieval rate if different pattern selection strategy is used. For example, with the retrieval rate strategy, the retrieval rate can also be tuned to nearly 100% with the accuracy rate measured around 96%. This can be achieved if we choose an aligned pattern that comprehend more variety in the records. Of course, such patterns may sometimes extract extra information because the patterns generalized from multiple string alignment may comprehends more patterns than the input. This is a tradeoff. If retrieval rate is more important, accuracy rate can only be sacrificed.

5 Conclusion

With the growth of the amount of online information, the availability of robust, flexible IE systems will become a stringent necessity. The key component of IE systems is the set of extraction patterns that is used to extract from each document the information. The application of pattern discovery based approach to IE can save a lot of efforts since no labeling is required. By taking the advantages of HTML tags, we can observe regular and contiguous display pattern in machine-generated pages. This is the basis of this pattern discovery based approach and is shown to be successful in recovering the patterns for record display.

In this paper, we also show how multi-level analysis can be applied to extract finer information by dividing a record into multiple slots. It is with this analysis that the wrapper is complete (not only in record boundary extraction but also in attribute extraction). The key features of this approach are as follows. First, there is no need for user-labeled examples. Second, it is able to generalize over multiple (and only multiple) examples in a page at a time. Third, the extraction

rule is context-based instead of delimiter-based. Finally, it can handle exceptions such as missing attributes, multiple attribute values (which is considered as missing attributes), etc.

In addition to the rule generator, we also implement the extractor and an interface to complete the IEPAD system. Since there is no need for user-labeled examples, the interface of IEPAD is designed for viewing patterns and designating desired information slots rather than for labeling examples. The result of experiments shows that the accuracy-first pattern can achieve 96% retrieval rate with one training page. Of the 14 Web sites, twelve (ten) of them can achieve 100% retrieval rate with less than five training pages in the training set (validation set).

In the future, we plan to implement text-level encoding to extract more delicate information, even though most information is tag-separable. Also, a better presentation is helpful when the number of patterns discovered is too large. In addition, PAC analysis might give a more rigid bound on the number of training examples needed for learning the extraction rules.

References

[Anish and Knoblock, 1998] nish, N. and Knoblock, C. 1998. Wrapper generation for semi-structured internet sources. In *Proceedings of the Workshop on Management of Semi-structured Data.*

[Chang et al., 2001] Chang, C.H., Lui, S.C. and Wu, Y.C. 2001. Applying pattern mining to Web information extraction, In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining.* pp. 4-15.

[Chien, 1997] 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.

[Doorenbos et al., 1997] Doorenbos, R.B., Etzioni, O. and Weld, D. S. 1997. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the 1st International Conference on Autonomous Agents.* pp. 39–48, NewYork, ACM Press.

[Embley et al., 1999] 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.

[Gonnet et al., 1992] 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.

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

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

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

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

[Kushmerick, 1999] Kushmerick, N. 1999. Glearing the Web. *IEEE Intelligent Systems*, pp. 20–22.

[Morrison, 1968] Morrison, D.R. 1968. PATRICIA–Practical algorithm to retrieve information coded in alphanumeric. *Journal of ACM*, 15(4):514–534.

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

[Muslea, 1999] Muslea, I. 1999. Extraction patterns for information extraction tasks: a survey. In *Proceedings of AAAI'99: Workshop on Machine Learning for Information Extraction.*

[Selberg and Etzioni, 1995] Selberg, E. and Etzioni, O. 1995. Multi-engine search and comparison using the MetaCrawler, In *Proc. of the Fourth Intl. WWW Conference,* Boston, USA.

[Soderland, 1997] Learning to extract text-based information from the world wide web. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD-97).*