Annotation Free Information Extraction from Semi-structured Documents

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

Abstract

The vast amount of online information available has led to renewed interest in information extraction (IE) systems that analyze input documents to produce a structured representation of selected information from the documents. However, the design of an IE system differs greatly according to its input: from unrestricted free-text to semi-structured Web documents. This paper introduces OLERA – OnLine Extraction Rule Analysis to the rapid generation of IE systems that can extract structured data from semi-structured Web documents. In this novel framework, extraction rules can be trained not only from a multiple-record Web page but also from multiple single-record Web pages (called singular pages). Most of all, this framework requires no annotation labor that is required for many machine-learning based approaches. Evaluation results show a high level of extraction performance for both singular page and multi-record page.

Keywords: information extraction, IEPAD, online sequence analysis, singular pages, encoding hierarchy

1 Introduction

Due to the heterogeneity and the lack of structure of Web information sources, Web mining and searching tools often require a lot of efforts for manipulation among various data formats. Hence, there presents a special need for information extraction to extract information of interest from these documents. Tailoring an IE system to new requirements is a task that varies in scale dependent on the text type, domain, and scenario [8]. Therefore, designing a trainable IE system has been an attractive application in machine learning research and the computational linguistics forum, Message Understanding Conference (MUC).

The task of Web IE differs greatly from traditional IE tasks [12]. Traditional IE involves free-text that are fully unstructured. Web IE, on the other hand, process online documents that are marked by HTML tags and sometimes presented in nearly structured format (e.g. semi-structured Web pages generated by CGI scripts). Free-text information extraction roots from linguistic analysis

and message understanding [25], whereas Web IE often relies on landmark identification marked by HTML tags and other delimiters.

Previously, several research efforts have focused on wrapper generation for Web-based sources, for example, WIEN, Softmealy, and STALKER, etc. [19, 16, 21]. The IE template typically involves multiple fields, some of which may have multiple instantiations in a record. Basically, the wrapper induction systems generate a specialized extractor for each Web data source. Their work produce accurate extraction results, but the generation of the extractors requires human-labeled/annotated Web pages accompanied with the training examples to "tell" a wrapper induction program how to extract data from a given Web page. By providing examples on how to partition a Web page and how to group sub-strings into fields and data records, wrapper induction systems learns the common landmarks as extraction patterns for each field and record boundary.

IEPAD (an acronym for information extraction based on pattern discovery) [4, 5, 6] is a novel trainable IE system which attempts to eliminate the need of user-labeled examples. The user does not need to tell the system what information to extract, but simply choose among patterns (discovered by IEPAD from the training example) to see if the pattern can extract the desired information. This amazing feature is based on the assumption that the input is a multiple-record Web page so that sequential pattern mining can be applied to discover the repeats. However, if the training page contains only one record, IEPAD fails.

This paper provides a novel framework – OLERA, for training extraction rules from Web pages with arbitrary number of records. Annotation of the training pages is replaced by three operators: *enclosing* an information block of interest, *drill-down* through an encoding hierarchy, and *specifying* relevant information slots for each field of the template. Extraction rules are learned by pattern matching and string alignment techniques in a way similar to how people generalize them. The experiments are conducted for both multi-record pages and single-record pages. Evaluation results show 96% retrieval rate for multiple-record pages with two training pages. As for single-record pages, the system achieves 93% retrieval rate with 5 training tuples.

The remainder of the paper is organized as follows. Section 2 reviews background material on information extraction and IEPAD. We describe OLERA's system framework in Section 3. Section 4 presents extraction examples on singular pages, multi-record pages, and pure texts. Section 5 gives the implementation of the OLERA extractors. Experimental results are shown in Section 6. Section 7 discusses related work in applying machine-learning to IE and finally, section 8 concludes the paper.

2 Background

2.1 Information Extraction from Semi-structured Data

Information Extraction (IE) is concerned with extracting the relevant data from a collection of documents. The goal is not necessarily to produce a general-purpose IE system, but to create tools that would allow users to build customized IE systems quickly. Since a basic component of any IE system is a pattern matching procedure, research efforts have focused on the work to generalize useful extraction patterns from training examples.

There are four types of IE tasks defined by MUC: name entity recognition (NE), coreference resolution (CO), template element construction (TE), and scenario template production (ST), from single-slot filling to complex template extraction [12]. For example, the template may be composed of several objects, each containing several slots. Slots may contain pointers to related objects [1]. Extracted information is to be stored in an object-oriented database in order to represent complex relationships among entities. For semi-structured IE, the template can be considered as a relation of k-tuples, where k is the number of attributes of interest in the record. The extraction task is a single-slot filling when k equals 1.

As mentioned above, tailoring an IE system is a task that varies with the text type, domain, and scenario [8]. In addition, there are three factors that affect the design a trainable IE system:

- First, whether the training examples are annotated may influence the design. Generally speaking, the input to a training system contains one or several pages as well as the answer keys in these pages. Sometimes, the answer keys are annotated by users in the pages as labeled-training examples. Most machine learning based approaches rely on user-annotated training examples, either free-text IE [22, 3] or semi-structured Web IE [18, 16, 21]. Very few systems generate extraction rules based on unlabeled text. AutoSlog-TS [23] and IEPAD [4] are two of the few systems for free-text and Web IE, respectively.

- Second, depending on the characteristics of the application domains, IE systems use extraction patterns based on one of the following approaches: delimiter-based constraints, content-based constraints, or a combination of both, which we call context-based rules. For example, wrapper induction systems for Web IE such as WIEN [18], Softmealy [16], Stalker [21], etc. generate delimiter-based extraction rules, while single-slot filling such as date or country names can be extracted based on content constraints. In addition, some systems [23, 3, 4] generate context-based rules which concerns both delimiters and content constraints.

- Finally, most IE systems may rely on background knowledge for pattern generalization. For example, RAPIER [3] imposes content constraints based on the WordNet semantic classes. SRV [11] involves several features to map text tokens into some discrete value, for instance, token length, character type (numeric or alphabetic), part-of-speech, etc. Softmealy [16] defines a simplified token hierarchy including HTML tag token and non-HTML token, word and nonword token classes, capitalized word and all lower case word, etc. With such background knowledge, IE systems can generalize either delimiter-based rules or context-based rules, etc.

These three perspectives can help us to estimate the difficulty of individual IE tasks and develop an trainable IE system. For example, it is possible to develop an IE system with un-annotated training example for semi-structure IE tasks. However, the difficulty to design such IE systems for free-text IE tasks is comparably high. In addition, we also rely on system developers's knowledge to add features or delimiters so that programs can generalize extraction rules, either delimiter-based or context-based.

2.2 The IEPAD System

IEPAD [6] is a wrapper generator which learns extraction rule for semi-structured Web pages. IEPAD focuses on Web pages that contains multiple-records in a training example. The extraction template is a record of fields. A field may has zero or more instances. IEPAD does not require answer keys or user-annotations in the training pages. IEPAD learns context-based extraction rules for a Web page class that is obtained from the same information source or generated by the same CGI script.

Since multiple-record pages (say, k records in a training page) are often rendered in a regular format, the key idea of IEPAD is to discover regular and adjacent patterns that occur k times in the training pages. To automate the discovery without annotated information, IEPAD applies an encoding scheme to translate the training page into a token string, and utilizes a PAT tree structure and string alignment technique for constructing the pattern. A pattern in IEPAD is a subclass of regular expressions over the alphabet of tokens used by the encoding scheme. A pattern may contain options and alternatives. An example of a pattern is given below:

`<B><TEXT></B><I><TEXT></I>[<BR>]`, (i)

where `<B>, <I>, <BR>, <TEXT>` etc. are tokens that match HTML tags `<b>, <i>, <br>` and text strings, respectively. Options are denoted by "[...]". In this example, the sixth token `<TEXT>` is optional. The following string matches this pattern twice:

```
<b>Congo</b><i>242</i><br><b>Egypt</b><i>20</i>
```
(ii)

Given a pattern and a Web page, the IEPAD extractor works as follows: the extractor translates the Web page into a token string with the encoding scheme and scans the token string to find all substrings that matches the pattern. It then outputs the substrings as data records. In this case, the extractor will output the following given the example pattern (i):

⟨"Congo", "242"⟩ ⟨"Egypt", "20"⟩

In terms of the three factors in IE system design, an encoding scheme is a kind of background knowledge we apply for pattern generalization. Since any text string between two adjacent tags is encoded as a special token <TEXT>, potential patterns can be revealed. This is a process similar to how we might generalize extraction patterns by observation. Therefore, for a training page that contains k records, the problem becomes discovering patterns that occur k times. PAT-tree is a perfect structure for discovering exact repeats, while multiple string alignment is incorporated to tolerate variance for information that contains missing attributes. Finally, regularity and adjacency are two characteristics used to sift patterns, otherwise users might be left with too many patterns to choose from.

A suitable encoding scheme to reveal the pattern is important, otherwise, we will discover a lot of patterns. Besides, such patterns may not present enough regularity and adjacency for a potential pattern, since these two characteristics are computed in numbers of tokens. For different Web page classes, the best encoding schemes may be different, or users have to adjust the parameters for regularity and adjacency to avoid too many patterns. Fortunately, many semi-structured Web pages reveal good extraction patterns in (all) HTML-tag encoding scheme or block-level-HTML-tag encoding scheme [4, 5].

Their are two potential problems in this pattern discovery based approach. First, it is hard to breaking ties when several alignments for two strings all have the optimal score. This will cause alignment error and increase the number of generated patterns to choose from. Second, IEPAD can only apply to multiple-record pages. For singular pages that contain only one record, it simply fails. Therefore, we propose a new approach to solve both these problems.

3 The OLERA's System Framework

In this section, we present the framework of our novel approach. The approach is mainly designed to solve two problems. The first and also the inspiring motivation is learning extraction rules for pages containing single-record. Though many systems are capable of this job with users' annotation of the

answer keys, it requires considerable efforts to label answer keys for k fields of a record. Therefore, we focus on methods to handle un-annotated training pages. The second effort is to extend this approach to other semi-structured documents other than HTML pages. For example, many semi-structured texts are delimited by punctuation marks. If such delimiters are employed, we can extract more than just snips separated by HTML tags.

OLERA is an acronym for On-Line Extraction Rule Analysis, which borrows the idea of cube operations from OLAP for multi-dimensional data model [15]. We incorporate a set of background knowledge and define an encoding hierarchy for abstracting training examples. Based on the encoding hierarchy, OLERA offers three operations: enclosing, drill-down, and specifying, to manipulate the information of interest. Detail description of these operations are given below:

- Enclosing an information block of interest

  Users can mark a block and use the "Ctrl+C" hot key to indicate a block of interest. The enclosed block is the target where extraction rule analysis proceeds. Comparing to annotated training pages, OLERA only requires users to mark the global scope of relevant information. However, this operation triggers a series of procedures, including identifying similar occurrences of the enclosed block, aligning these occurrences to a general format, and displaying them in a spread sheet for users, etc.

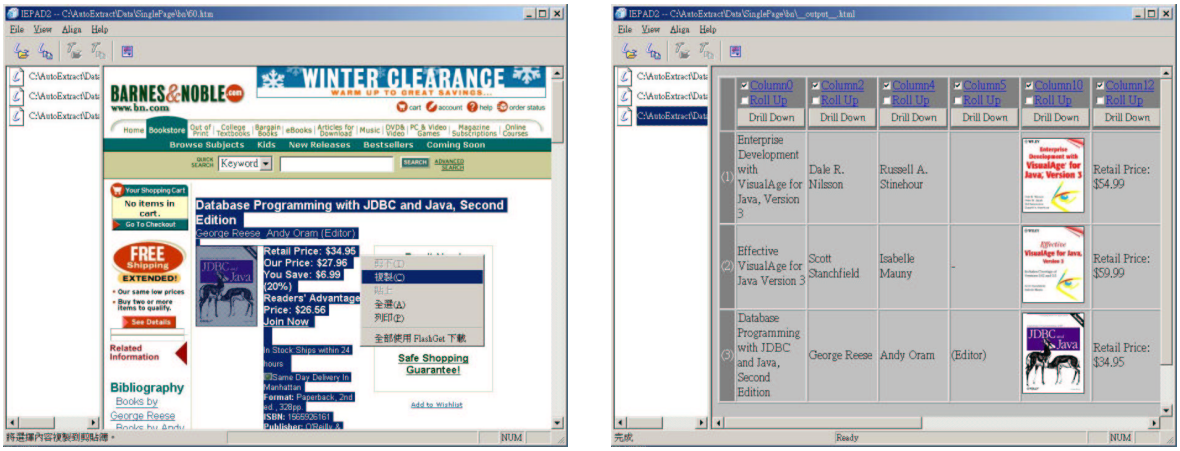
- Drilling down an information slot

  Drill-down operation allows users to navigate from a text fragment to its detailed components. This is realized by stepping down an encoding hierarchy which is based on markup language syntax and general string custom as will be described in section 3.1.

- Specifying relevant information slots

  The result of the above operations is presented by a spread sheet with multiple slots decomposed from the enclosed information block. Users can then specify relevant slots by mark them for extraction.

Figure 1(a) shows an example of Barnes&Noble's Web pages where a block of book information is enclosed for analysis. This page is what we called a `singular page` since each page contains only one record describing one book's information. Extraction rule analysis is performed by three steps. First, translate the training page using an encoding scheme in the encoding hierarchy. Second, match the pattern of the enclosed block to discover possible records in other training pages. Third, align matched strings to generalize the record pattern. The result of this analysis is a `record`



(a) (b)

Figure 1: (a) Singular Web page – Barnes&Noble (b) Aligned records from training pages

`extraction rule` and the discovered records (in the training examples) which are displayed in a spreadsheet with m rows and n columns for records and information slots, respectively (Figure 1(b)).

In order to extract finer information, drill-down operations can be applied to individual columns by executing step 1 and 3. The process can be repeated until all desired information is decomposed. Information slots of interest can then be specified and saved for later use by OLERA extractors (or wrappers). The mining procedure of the extraction rule is similar to the learning process of pattern discovery by human beings. In the following, we describe these three steps in detail.

3.1 Encoding Hierarchy

As mentioned above, an encoding scheme is used to translate a training page into an abstracted token string in order to reveal regular patterns. We adopt the encoding scheme introduced in IEPAD. Given a set of delimiters, the encoding procedure translates each delimiter, `X`, in the input page as a delimiter token `X` and translates any text string between two delimiters as a special token `<TEXT>`. The encoding approach not only translates an input page into a token string, but also enforces a natural segmentation of the data by the delimiters. In other words, an encoded token string of length n can divide the data into n segments, where each `<TEXT>` tokens represent a text segment between two adjacent delimiter tokens. We call this segment the *primitive data* of the `<TEXT>` token.

The advantage of this encoding scheme is to translate the training page into a better format for pattern discovery and alignment as shown in IEPAD. However, pattern discovery often requires a high level abstraction which may not be appropriate for extraction of finer information. Hence, multi-level analysis is required [6]. Suppose we have an encoding hierarchy, we can choose a higher-

Category	Encoding scheme	Delimiters
Markup-level	Block-level tag	block-level tags
	text-level tag	text-level tags
Text-level	Paragraph	NewLine, CarriageReturn, Tab
	Sentence	Period, Question Mark, Exclamation Mark (followed by a blank space)
	Phrase	Colon, Comma, Semicolon, Bracket, Quotation Mark
Word-level	Word/Nonword	(,), $, -, /, @, etc.
		Blank

Table 1: The Encoding Hierarchy for Web Documents

level encoding scheme for pattern discovery and use another encoding scheme to decompose a segment into smaller pieces. This is similar to the drill-down operation in OLAP data cube by stepping down a *encoding hierarchy*.

The concept hierarchy for the drill-down operation in OLERA is composed of a set of encoding schemes classified into three layers: `markup-level encoding > text-level encoding > word-level encoding`. The greater-than sign indicates that the left encoding is a higher level abstraction of the right one. The primitive data of the `<TEXT>` tokens encoding by higher layer's encoding scheme can be encoded by lower-layer's encoding scheme at the drill-down operation.

Each level of the encoding hierarchy contains finer classification of several encoding schemes. For example, markup-level contains `block-level-tag` encoding scheme and `text-level-tag` encoding scheme which are introduced in IEPAD. This is not the only way to classify markup-level encoding schemes, since several HTML tags such as `<BR>, <P>` can be classified as either block-level tags or text-level tags. For other semi-structure documents such as XML, the encoding can be defined by the layer position of the tags in the parse tree of an XML page.

For further drill-down operations and to extend this approach for non-markup documents, we establish text-level and word-level encoding schemes where the delimiter sets might depend on the constituents of the text segments. For example, text data are usually made of paragraphs and sentences separated by control characters such as new-line (NL), carriage-return (CR), etc. For word-level encoding schemes, we concern the constituents of sentences – words separated by blank spaces, and other symbols such as parenthesis, dollar signs ($), dashes, slash etc.

In addition to delimiter-based encoding scheme, language-specific information such as part-of-speech tagging, semantic class information, etc. can be applied too. For instance, sentences can be parsed into proper grammar notations such as `<Subject><verb><dobj>`. Many date-related strings such as "2002/9/1" or "9/1/2002" can be recognized as DATE token. In the context of

semi-structured Web IE, we focus ourselves on simpler text data segmented from Web pages. The encoding hierarchy used in this paper is listed in Table 1 which shows a common knowledge for string composition.

3.2 Approximate Occurrence Identification

The input to OLERA is a set of one-record or multiple-record Web pages. The enclosing operation can be applied to one (or several if necessary) of the training pages. We describe in this section how the records in other training pages are recognized. Similar to IEPAD, the user can specify the number of records in the enclosed block. Let k be the number of records in the enclosed block. For $k = 1$ (default), the encoded token string of the enclosed block is saved as the primitive record pattern, P. For $k > 1$, we can apply IEPAD's algorithm to discover the patterns that occur k times and express the rule as an regular expression R [4]. We defer the discussion of this scenario to Section 3.4.

To discover other records in the input pages for further training, exact pattern matching is of no help in providing new information. Instead, we expect the discovery of substring that are similar to P so that we can generalize these substrings for extending the pattern. We develop a precise definition of similarity between two strings S_1 and S_2 as follows [14].

Definition 3.1 *Let Σ be the alphabet used for strings S_1 and S_2, and let Σ' be Σ with the added character "–" denoting a space. Then, for any two character x, y in Σ', the function $match(x, y)$ denotes the value obtained by aligning character x against character y.*

In this paper, we give each match of characters in Σ a value of 3 (denoted by s), each mismatch a value of -3 (denoted by r). The matching of a character in Σ with a space is given a value of -1 (denoted by d). Minor change on matching score will be discussed later.

Definition 3.2 *An (global)* `alignment` *of two strings S_1 and S_2 is obtained by inserting chosen spaces either into or at the ends of S_1 and S_2, such that the resulting strings S_1' and S_2' are of equal length. The* `value` *of such an alignment is defined as $\Sigma_{i=1}^{l} match(S_1'[i], S_2'[i])$, where l denote the (equal) length of the two strings S_1' and S_2' in the alignment.*

Definition 3.3 *The* `similarity score`*, $SC(S_1, S_2)$, of two strings S_1 and S_2 is the optimal value of all (global) alignments between S_1 and S_2. We also define* `similarity ratio` *as the similarity score divided by $s \cdot \min\{|S_1|, |S_2|\}$, the maximum value by matching S_1 and S_2.*

The value of two strings S_1 and S_2 and the associated optimal alignments can be computed by dynamic programming with recurrences with base conditions

$$V(i,0) = i * d; \tag{1}$$

and

$$V(0,j) = j * d; \tag{2}$$

where $V(i,j)$ be the value of the optimal alignment of prefixes $S_1[1..i]$ and $S_2[1..j]$. For i and j, the general recurrence is

$$V(i,j) = \max\{ \begin{array}{l} V(i-1,j-1) + match(S_1[i], S_2[j]); \\ V(i-1,j) + d; \\ V(i,j-1) + d; \end{array} . \tag{3}$$

If S_1 and S_2 are of length n and m, respectively, then the value of their optimal alignment is given by $V(n,m)$. The value, and the entire dynamic programming table, can be obtained in $O(nm)$ time, since only three comparisons and arithmetic operations are needed per cell. String similarity is one of the ways to formalize the relatedness of two strings. An alternate and more common way of formalizing the relatedness is to measure their `edit distance` as used in bio-informatics literatures.

Now, to discover approximate occurrences of a string P in a training page T (an encoded token string), we use a variant of global alignment which are defined as follows:

Definition 3.4 *Given a parameter θ ($0 < \theta < 1$), a substring T' of T is said to be an approximate occurrence of P if and only if the similarity ratio of P and T' is at least θ, or the similarity score is at least $\delta = \theta \cdot s \cdot |P|$, where $s \cdot |P|$ denotes the largest value matching pattern P.*

The problem of determining if there is an approximate occurrence of P in T can also be solved using dynamic programming with the same recurrences for global alignment but change the base conditions for $V(0,j)$ to $V(0,j) = 0$ for all j. Using this variant of global alignment, we have the following theorem.

Theorem 3.5 *There is an approximate occurrence of P in T ending at position j of T if and only if $V(n,j) \geq \delta$, where n is the length of P. Moreover, $T[k,j]$ is an approximate occurrence of P in T if and only if $V(n,j) \geq \delta$ and there is a path of backpointers from cell (n,j) to cell $(0,k)$.*

To discover all approximate occurrence of P in T, we first identify the position j' in T such that $V(n,j')$ has the largest value among all $V(n,j)$, and $V(n,j') \geq \delta$. For this j', output the shortest

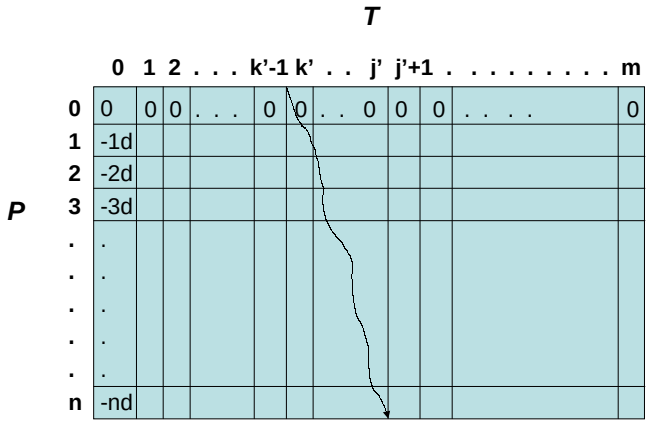


Figure 2: Dynamic Programming for approximate pattern matching.

approximate occurrence, $T[k', j']$. The substring $T[k', j']$ is found by traversing the backpointers from (n, j') until a cell in row zero $(0, k')$ is reached. We then apply this procedure to $T[1, k' - 1]$ and $T[j' + 1, m]$ recursively to find all approximate occurrence of P in T. (See Figure 2)

Though this definition of similarity seems to work for approximate string matching, there are problems in real applications. The major problem is that several alignments may have the same optimal similarity score. Even if we can enforce rules to reduce the number of alignments, e.g. breaking ties by choosing a vertical pointer over a diagonal one and a diagonal one over a horizontal one, the most correct alignment may be lost. Take the following for example, all of the following alignments for "*dtbt*" and "*dtbtbt*" have the same similarity score.

$$\begin{array}{cccccc} d & t & b & t & b & t \\ d & t & b & t & - & - \end{array} \quad , \quad \begin{array}{cccccc} d & t & b & t & b & t \\ d & t & - & - & b & t \end{array} \quad \text{and} \quad \begin{array}{cccccc} d & t & b & t & b & t \\ d & t & b & - & - & t \end{array}$$

Instead of making rules for breaking ties, we propose a calculation for the value of an alignment to distinquish various alignment. Recall that in delimiter-based encoding schemes, a segment of text is encoded as a delimiter token or `<TEXT>` token. Matching two characters of the same delimiter token is quite different than matching two characters of `<TEXT>` token, since the primitive data of a `<TEXT>` token can be vastly different. Therefore, we make a comparison between the primitive data of each matched `<TEXT>` tokens to ensure the correctness of the matching. This can compensate the lost of information during the encoding process and is better than simply gives a value of s for matching two `<TEXT>` characters.

Therefore, string alignment in this paper is implemented by using two encoding schemes. The first encoding scheme presents the structure of the input texts and the second encoding scheme

abstracts the primitive data for <TEXT> tokens encoded previously. The reason is that it is usually hard to compare pure texts character by character, especially for long segments of primitive data. Hence, we need some abstraction of such data for comparison. The value of matching two <TEXT> tokens is given by the similarity score of the encoded token strings of two primitive data divided by the shorter length of the two token strings. This will return a value between 0 and s (If two primitive data are identical, we get a score of s.). Matching two delimiter tokens still get a score of s. The test on primitive data is enforced through out the paper so that the matching is correct not only for a exterior resemblance but also an interior resemblance.

For example, for a <TEXT> token from block-level-tag encoding scheme, we would consider its encoded representation at text-level-tag encoding scheme. The matching score of two <TEXT> tokens in the second level alignment will simply be given s since we only cares the similarity score instead of the alignment result. The similarity score will then be normalized by dividing the minimum length of two token strings.

3.3 Multiple String Alignment

For multiple records identified from the approximate occurrence identification, we need a generalization over these instances. Let's say k token strings are discovered after occurrence identification. We will apply (global) multiple string alignment procedure to the k token strings to generalize the record extraction rule. A formal definition is given below (quoted from [14]):

Definition 3.6 *A* `global multiple alignment` *of $k > 2$ strings $S = S_1, S_2, ..., S_k$ is a natural generalization of alignment for two strings. Chosen spaces are inserted into or at either end of each of the k strings so that the resulting strings have the same length, defined to be l. Then the strings are arrayed in k rows of l columns each, so that each character and space of each string is in a unique column.*

For example, Figure 3 shows a multiple alignment of three token strings "`dtbtbt`", "`dtbt`" and "`dtbat`". With multiple string alignment, we can represent a set of records in *profile* representation or *signature* representation, which can be used for extraction in the test pages. In this paper, we adopt signature representation for extraction rules since the symbol set for each column should not vary a lot. The signature representation of Figure 3 will be expressed by "`dtb[a|-]t[b|-][t|-]`", which divides the texts into 7 columns.

The goodness of a multiple alignment can be estimated using an extension from two string alignment, which is called *sum-of-pairs* objective function.

d	t	b	-	t	b	t
d	t	b	-	t	-	-
d	t	b	a	t	-	-

Figure 3: A multiple alignment for three token strings

Definition 3.7 *The* `sum of pairs (SP)` *score of a multiple alignment M is the sum of the similarity scores of pairwise global alignments induced by M, where the induced pairwise alignment is defined as follows.*

Definition 3.8 *Given a multiple alignment M, the* `induced pairwise alignment` *of two strings S_i and S_j is obtained from M by retaining S_i and S_j and removing any two opposing spaces in these two rows.*

The `SP` problem can be solved exactly via dynamic programming for a small number of strings with $O(n^k)$ time for k strings of length n. Bound-error approximation algorithm for large numbers of strings is also available from [13]. The approximation algorithm, called *center star* method, proceeds by constructing the multiple alignment one string at a time. Given a set of k strings S, first find the `center` string $C_S \in S$ which maximizes $\sum_{S_j \in S, S_j \neq C_S} SC(C_S, S_j)$, where function SC denotes the similarity score of two strings. Renumber the strings $S - \{C_S\}$ from 1 to $k-1$. Next, select S_1 and optimally align S_1 with C_S. Let C_j and S'_j denote the center string and S_j after alignment (with spaces inserted). The existing multiple alignment will have two strings C_1 and S'_1.

To continue, for each S_j, $j = 2, \ldots, k-1$, form a two-string lignment of S_j and the center string, C_{j-1}, in the existing multiple alignment using the above mentioned scoring scheme with the added rule that two opposing spaces have zero value. If the pairwise alignment alignment does not insert any new spaces into C_{j-1}, then append S'_j to the existing multiple alignment. If the pairwise alignment alignment does insert a new space in C_{j-1} between some character l and $l+1$, then insert a space between characters l and $l+1$ in every string in the existing multiple alignment. Then replace C_{j-1} with C_j and append S'_j to that multiple alignment to add one more string. The time for this center star method is $O(k^2 n^2)$ [1].

Multiple string alignment has been applied in IEPAD to generalize the contextual presentation of the critical common features and tolerate exceptions induced by missing attributes. The major problem in previous work is the decision of an alignment when multiple alignments have the same optimal edit distance or similarity score. In order to find a correct alignment, the match function

[1] Most of the material on string comparison can be obtained in "Algorithms on Strings, Trees, and Sequences" by Dan Gusfield, Cambridge, 1997, Chapter 11 and 14.

needs to consider more than just facile tokens, especially on matching non-delimiter tokens as described in Section 3.2. With this new matching function, the system can make a better alignment.

3.4 Summary

In the above, we describe the three main techniques used in OLERA. It will be clear now how they are used in three operations. For example, the enclosing operation is the most complex one which involves three steps: it triggers the training procedure which identifies approximate occurrences (after encoding) and generalize them to an extraction pattern by multiple string alignment. A drill-down operation on an information slot involves two steps including encoding and multiple string alignment over the encoded token strings of that column. Finally, information slots of interest can be specified and given proper field names. The corresponding encoding scheme used by each drill-down operations as well as the positions of the specified slots will be record in the extraction rule such that the extractors generated by OLERA can perform the same encoding and extraction of the specified slots as recorded. The details of the extractors will be described in Section 5.

In Section 3.2, we describe the situation when one training page is enclosed and the number of expected records k equals one. We now consider two other scenarios when k is greater than 1 or there are more than one enclosed block. When k is greater than one, we can apply IEPAD's algorithms to discover patterns that occurs k times in the enclosed block, and then generalizes the segments between two adjacent occurrences by multiple string alignment. The record pattern is then expressed as a signature representation R which contains alternatives. Note that the method described in Section 3.2 finds approximate occurrences of a string P in a training page T. Here, we have a regular expression R instead of a simple string P. Therefore, we extend R to the longest expression P (without spaces and alternatives) and apply the technique of in Section 3.2 to identify approximate occurrences of P. For example, for a regular expression $R =$"`dtb[a|-]t[b|-][t|-]`", the longest expression is $P =$"`dtbatbt`".

In another scenario, the users may enclose more than one blocks. In this case, we first discover the record pattern for each block and extend any regular pattern to its longest expression as described above. Next, we conduct a simple (single pass) clustering to decide whether these patterns should be aligned. First, we compute the similarity ratio of two patterns P_i and P_j. If the similarity ratio of these two patterns is larger than a predefined parameter θ, these two patterns are aligned. Otherwise, we simply keep each individual pattern as a separate cluster. Other pattern P_k will be compared to each of these two clusters, respectively. The similarity ratio will decide whether

to align P_k to one of the existing cluster or to form a independent cluster. In other words, if two enclosed (training) records are too different, we would rather keep two regular expression for them to prevent odd alignment. This will help deal with situations when a CGI script has several display templates to choose from such as Amazon and Barnes&Noble. The principle is the same as in Section 3.2, we only apply multiple string alignment to approximate occurrences that are similar to P. Therefore, only patterns with similarity ratio greater than θ are aligned for record representation.

Finally, this process of training an extraction rule is similar to the process of generalize an extraction rule by people. The encoding scheme (decomposition) and multiple alignment enables the program to simulate the work and express the record pattern as an regular expression. As long as the matching score of two abstracted tokens are properly defined, the alignment will give a result similar to what we expect.

4 Examples

In the section, we show three examples of this approach to solve the extraction of singular pages, multiple-record page, and non-HTML documents. The first example regards the application to the extraction from multiple Web pages, each containing one record information. Figure 1(a) in Section 3 shows an example of singular pages, where the information of interest is enclosed and copied for analysis. As described above for $k = 1$, OLERA's procedure is to matche the encoded token string of the enclosed block (which is considered as the record pattern) against other training pages. The alignment result of the tree training pages is as shown in Figure 1(b).

The second application is about the extraction from Web pages, each containing multiple records. The difficult part is to discover the boundary for each record in the Web page which has been addressed by IEPAD using PAT-tree construction and multiple string alignment [4]. To distinguish OLERA's approach to IEPAD's, we enclose only one record as shown in Figure 4 and let OLERA identify other approximate records. The result of this approximation is shown in Figure 5, where we show the primitive data of `<TEXT>` tokens as well as the link information for `<A>` tokens in the record grammar. Figure 5(a) shows the first three columns. Figure 5(b) demonstrates the drill-down operation on the last `<TEXT>` token with delimiter ":" to separate the string "URL" with the link. Since colons also exist as part of the link, roll-up operations can be applied to restore the links.

The third example is a non-HTML semi-structured document from dbEST (one of many Genebank databases hosted by NCBI). Figure 6 shown an example of the dbEST files, which

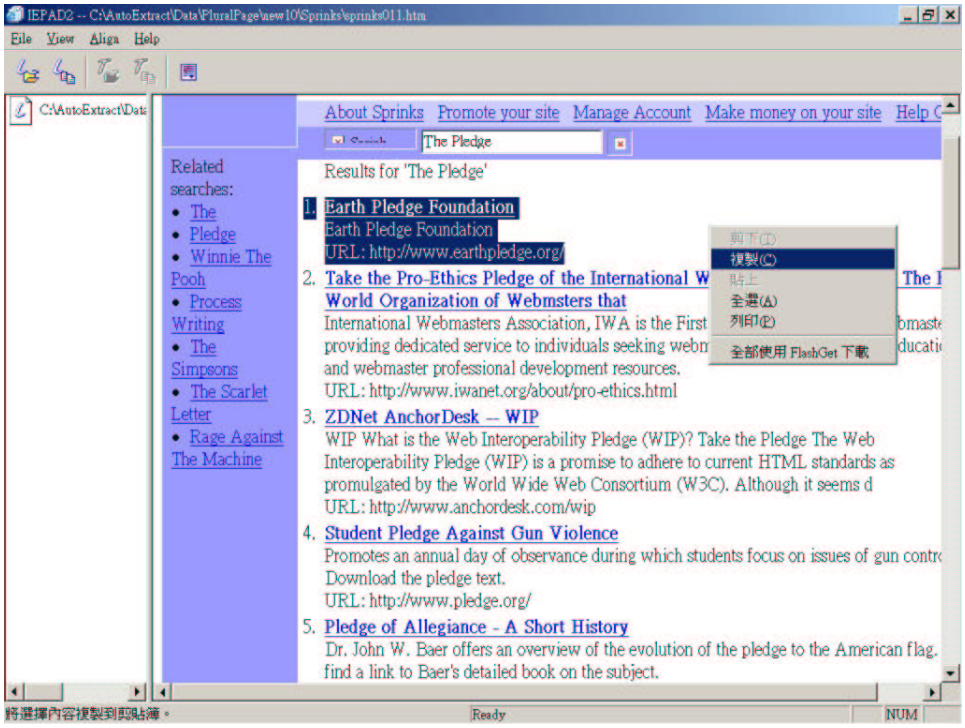
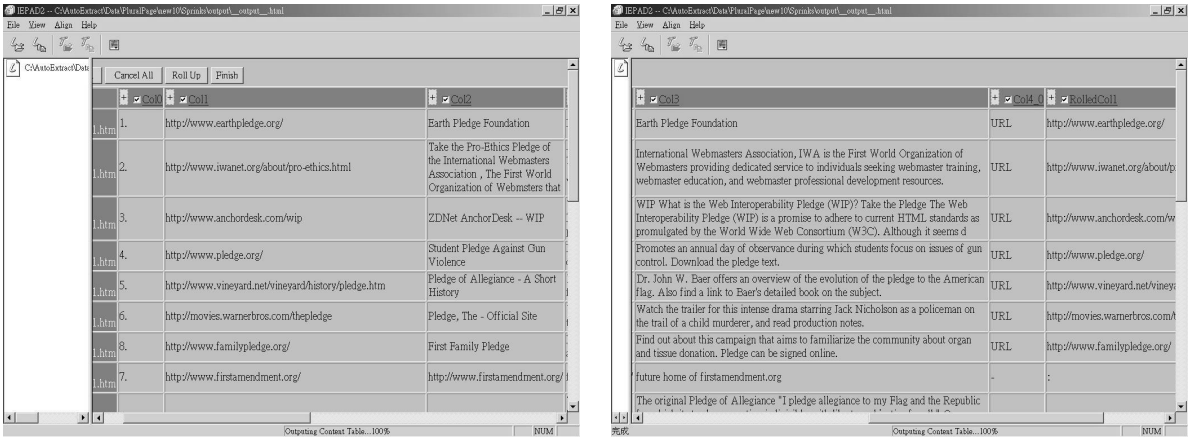


Figure 4: Multiple-record page – Sprinks.



(a) (b)

Figure 5: Aligned records from Sprinks: (a) the first 3 columns (b) drill-down and roll-up operations.

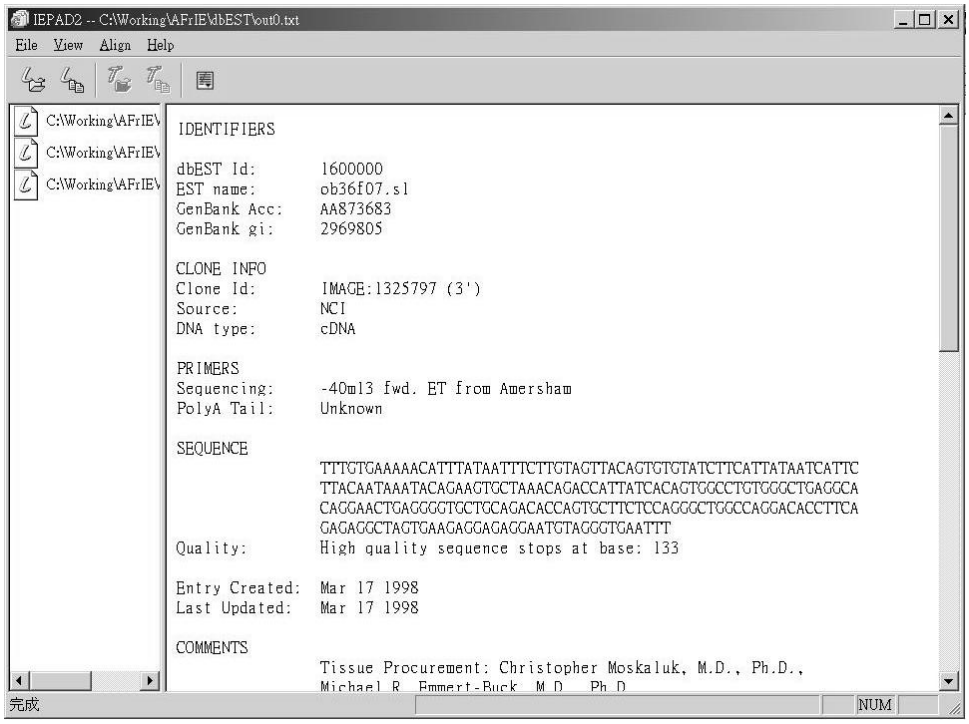
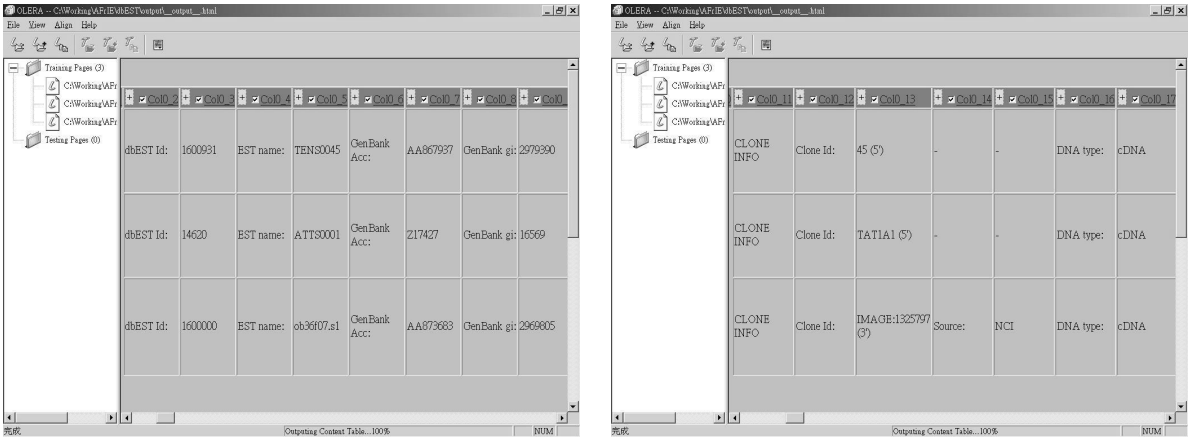


Figure 6: Non-HTML document – dbEST.



(a) (b)

Figure 7: Aligned records from dbEST

can be encoded by delimiters: tab-stops and new-lines from users' point of view. We must stress again that primitive data comparison is important in this example, since it controls how `TEXT` tokens are aligned. Figure 7 shows the result of this alignment.

Compare OLERA with IEPAD, OLERA can solve the problem of singular page extraction which can't be solved by IEPAD. In addition, it does not generate many patterns as IEPAD does for the extraction of multiple-record pages, since OLERA focused the analysis on an information block of interest, while IEPAD analyzed the whole training page. In order to tolerate variance in records, OLERA further provides approximate matching of training example and aligns similar instances to a general expression. In addition, the definition of matching score between two `<TEXT>` tokens has also improved the alignment result and reduced the number of possible alignments. Finally, by the drill-down operation along the encoding hierarchy, it allows the extraction of very fine information.

It is worth mentioning that the training pages are not necessarily all singular or multi-record pages. We may choose one singular page for block enclosing, and take other plural pages for approximate pattern matching. As long as one record pattern can be identified, approximate occurrence identification can be applied to other training page, either singular or plural, to recognize other records in the training examples.

5 The OLERA Extractors

Similar to most IE systems, the OLERA extractor is essentially a pattern matching program which finds the occurrences of the record pattern and then extracts information for each designated slot. An OLERA extraction rule contains three constituents: a grammar representation for record pattern, the drill-down and roll-up operation applied, and the specified slots for output. For each drill-down operation, a grammar representation is used for that column. A grammar contains two encoding schemes, a signature representation for the record pattern. In addition to the signature representation, the grammar also records and the longest token string representation for each `<TEXT>` token to reconstruct the alignment.

An OLERA extractor follows the encoding schemes specified in the extraction rule to translate a testing page. It then searches for an instance of the record pattern R in the encoded token string. Once the instances are identified, the extractor follows the drill-down operations to the specified columns as indicated in the extraction rule. Finally, information slots of interest are extracted accordingly. Among these steps, instance identification is the most difficult one. Therefore, we discuss the related choices below.

The first choice is reusing the technique of *approximate occurrence identification* by extending

R to the longest expression without spaces and alternatives. Since we have a regular expression R instead of a simple string P, we extend a regular expression to its longest expression by removing all spaces. For example, for a regular expression "<P><Text><P>[<Text>|-]", the longest expression is "<P><Text><P><Text>". This method is good for it not only identify an instance but also align it with the pattern. However, it might discover many similar strings beyond the generalized pattern. For example, allowing approximate occurrences might identify "<P><TEXT><P>" as well as "<TEXT><P><TEXT>", where the later is unexpected. Since this might decrease precision, we do not use this approach.

The second choice is to apply *regular expression* pattern matching, since the record pattern is presented by regular expression. This should be followed by a global alignment with the discovered substring since we also need an alignment of the instance to the record pattern such that correct data segments be extracted. The main problem is that regular expression matching does not consider the difference in matching <TEXT> tokens and delimiter tokens. Therefore, it looses the chance to identify the correct instance when multiple (co-located) substrings are all candidates. For example, two co-located substring can be matched by one pattern. For example, the above regular expression "<P><Text><P>[<Text>|-]" can match two co-located substrings: "<P><TEXT><P>" and "<P><TEXT><P><TEXT>", of a text "...<P><TEXT><P><TEXT>...". It will depend on alignment result to decide which one is correct.

6 Experiments

In order to evaluate OLERA's capabilities, we tested it on two data sets. The first data set contains 10 information sources that are singular Web pages. The second data set contains 14 information sources that are multiple-record pages used by IEPAD (http://www.csie.ncu.edu.tw/∼chia/webiepad/) [6]. Some of the information sources have been used in WIEN and STALKER, while others are collected for singular page extraction. Table 2 shows a detail description on these information sources including the average number of records in a page, the number of attributes in a record, the number of collected pages, and if the source contains missing or unordered or list attributes, etc.

We use an experimental setup that is similar to STALKER's: we start with one randomly chosen training page, learn an extraction rule, and test it against all the unseen pages. If not all records in an unseen pages are extracted, the page is added in the training set. For two training examples, we purposely enclose the most complex example, and train a new rule to extract other unseen pages. Repeat the same procedure for 3, 4, ... training pages until all testing records are

Source	# of records	k-tuple	# of pages	Missing	Unordered	List
A1books	1	8	100	Yes	No	Yes
Amazon	1	9	100	Yes	Yes	Yes
Barnes&Noble	1	8	100	Yes	No	Yes
Ebay	1	8	100	Yes	No	No
BookPool	1	6	100	Yes	No	Yes
iUniverse	1	8	100	No	No	Yes
PMIBook	1	11	100	Yes	Yes	No
JuilliardBook	1	5	100	Yes	No	No
ByuBook	1	3	100	Yes	No	No
Zagat's Guide	1	4	91	No	No	Yes
Quote Sever	3.7	11/18	209	Yes	No	No
Internet Address Finder	5.9	6	195	Yes	Yes	No
BigBook	14.2	6	235	No	No	No
OKRA	18.5	4	252	No	No	No
AltaVista	10	4	100	Yes	No	No
DirectHit	10	4	100	Yes	No	No
Excite	10	4	100	Yes	No	No
HotBot	10.2	4	100	Yes	No	No
Infoseek	15	3	100	Yes	No	No
MSN	10	3	100	No	No	No
NorthernLight	10	3	100	Yes	No	No
Sprinks	20	4	100	Yes	No	No
Webcrawler	25	3	100	No	No	No
Yahoo	20	4	100	Yes	No	No

Table 2: Information source description

Source	2	3	4	5	6	7	8	9	10	11
A1books	82.7	95.9	100	-	-	-	-	-	-	-
Amazon(1)	13.4	25.3	43.6	58.7	68.9	89.4	95.4	96.5	97.7	98.8
Amazon(m)	6.1	13.4	21.9	25.3	43.6	51.6	58.7	59.3	61.1	68.5
Barnes&Noble	14.3	20.6	24.0	35.8	60.6	67.7	81.5	84.6	-	-
Ebay	26.5	54.6	55.2	86.3	87.2	96.8	98.9	-	-	-
BookPool	79.6	84.5	91.7	93.7	97.9	100.0	-	-	-	-
iUniverse	92.8	100	-	-	-	-	-	-	-	-
PMIBook	-	-	-	-	-	-	-	-	-	-
ByuBook	70.1	100.0	-	-	-	-	-	-	-	-
JuilliardBook	87.8	88.7	94.6	95.8	96.8	97.85	98.9	-	-	-
Zagat	78.5	96.6	100	-	-	-	-	-	-	-

Table 3: Retrieval rate for singular pages

Source	2	3	4	5	6	7	8	9	10	11
A1books	100	100	100	100	-	-	-	-	-	-
Amazon(1)	100	100	100	100	100	100	100	100	100	100
Amazon(m)	100	100	100	100	100	95.8	96.3	96.3	96.5	92.4
Barnes&Noble	93.3	95.2	88.5	85.0	91.9	90.0	91.5	93.9	-	-
Ebay	100	100	100	100	100	100	100	-	-	-
BookPool	100	100	100	100	100	100	-	-	-	-
iUniverse	100	100	-	-	-	-	-	-	-	-
PMIBook	-	-	-	-	-	-	-	-	-	-
ByuBook	100	100	100	-	-	-	-	-	-	-
JuilliardBook	100	100	100	100	100	100	100	-	-	-
Zagat	100	100	100	-	-	-	-	-	-	-

Table 4: Accuracy rate for singular pages

successfully extracted. This procedure is repeated for 3 times and the numbers of test examples that are correctly extracted are averaged.

There are several parameters that can be adjusted including matching score, similarity threshold, and encoding schemes. The default values for the tree matching score are 2, -2 and -1 for s, r, and d. The ratio of $|s|/|d|$ can be increased to allow more missing attributes. The similarity threshold θ has a default value of 0.6 and can be decreased for record patterns with a lot of variations. The default encoding hierarchy is as shown in Table 1. Users can specify the encoding schemes used or simply gives a set of delimiters as an encoding scheme.

Table 3 and 4 shows the retrieval rate and accuracy rate for singular page extraction. The average retrieval rate (recall) for 5 pages is 77% with 98% accuracy rate (precision). If 10 training pages are used, the retrieval rate and the accuracy rate achieve 94% and 97%, respectively. Generally speaking, missing attributes can be handled by approximate matching with a higher value of s. Therefore, users only need to include such variations in the training set and enclose the most complex example. For information sources that containing several formats such as Amazon and

Source	Retrieval rate					Accuracy				
	1	2	3	4	5	1	2	3	4	5
LA-Weekly Restaurants	100	-	-	-	-	100	100	100	-	-
Quote Sever	39.1	100	-	-	-	100	100	-	-	-
Internet Address Finder	-	-	-	-	-	-	-	-	-	-
BigBook	98.7	99.9	100	-	-	100	100	99.9	-	-
OKRA	100	-	-	-	-	100	-	-	-	-
AltaVista	71.0	100	-	-	-	100	100	-	-	-
DirectHit	99.2	99.6	100	-	-	100	100	100	-	-
Excite	100	-	-	-	-	100	-	-	-	-
HotBot	96.1	98.7	99.5	100	-	100	100	100	100	-
Infoseek	98.7	98.9	99.7	100	-	100	100	100	100	-
MSN	99.0	99.4	99.5	-	-	100	100	100	-	-
NorthernLight	100	-	-	-	-	100	-	-	-	-
Sprinks	96.8	97.5	99.4	-	-	100	100	100	-	-
Webcrawler	100	-	-	-	-	100	-	-	-	-
Yahoo	97.0	100	-	-	-	100	100	-	-	-

Table 5: Performance for Multiple-record pages

Barnes&Nobles, or a lot of variations in one format such as JuilliardBook, users have to enclose more than one block for training.

In this experiments, Barnes&Nobles has the most complicated format, we need to enclose 3 blocks to achieve 96.6% retrieval rate with a sacrifice of precision. For Amazon, the Author attribute is a compound field which contains a list of authors, editors, associated information, etc. We consider two scenarios: if Author is considered as one field or several fields. The corresponding performance is shown in two rows. Apparently, the retrieval rate is much higher if the author is considered as one field. The signature representation contains up to 165 columns when Author is considered as multiple fields. Finally, PMIBook, due to the permutation of attributes, can not be handled by OLERA.

Table 5 shows the performance for multiple-record pages. The training pages needed for multi-record page extraction are comparably less than those for singular pages since each training page contains several records where variations can occur. In fact, since the number of attributes in a record for multiple-record pages is comparably smaller than that for singular pages (Table 2), the variations are small, too. For these 14 information sources, thirteen can all be perfectly extracted as in IEPAD [6] except for IAF due to its unordered attributes.

Based on the results above, we can draw several conclusions. First of all, compared with IEPAD, OLERA has the ability to wrap a larger variety of sources. Second, OLERA is capable of learning the extraction rules based on a few examples for most semi-structured documents. Third, by treating list attributes as missing items, OLERA is capable of learning extraction rule for list

attributes. Like IEPAD, this approach is natural language independent, since it incorporates only delimiter-based encoding schemes. Therefore, it can adapt to English, Chinese, etc. We have not yet incorporated semantic generalization, however, it can be considered as another type of encoding scheme and can be incorporated into the encoding hierarchy. Finally, OLERA is not able to handle attributes with permutation order.

7 Related Work on Web IE

The research on IE from semi-structured Web pages can be traced to the research of information agents that integrate data sources on the World Wide Web [9, 17]. A critical problem that must be addressed for these agents is how to extract structured data from the Web. Since it is not fit to write extractors for all the Web data source, machine learning approaches are proposed to solve the problem [24]. Web IE has been quite active in recent years and a lot of literature can be found [2, 19, 16, 21, 11, 10, 7, 20, 4].

WIEN [18] was the first wrapper induction system. It assumes that there is a unique multi-slot rule for a given Web site. As a result it fails to handle sites with multiple instances and variant attribte permutations. Softmealy [16] uses a wrapper induction algorithm that expresses the extraction rules as finite-state transducers. The rules are more expressive than those of WIEN because they can handle missing items, multi-valued items, and items appeared in unrestricted orders. Both WIEN and Softmealy use delimiters that immediately precede and follows the actual data. STALKER [21] is a wrapper induction system that performs hierarchical information extraction. STALKER produces extraction rules for each slot. It then uses Embedded Catalog Tree (ECT) to group individual items to assemble a multi-slot record.

Fully automatic approach to information extraction is rare and often depends on some heuristics. For example, Embley et. al. [10] describe a heuristic approach that discovers 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 [10]. However, one serious problem in this one-tag separator approach is that their system cannot be applied to Web pages where two or more separator tags are the same within a record, because their system cannot distinguish them and will fail to extract data correctly. As a result, the applicability of their system is seriously limited.

A similar work of OLERA is proposed by Chidlovskii, et al., which uses grammar induction based on string alignment [7]. In fact, it is the first paper which includes alignment for rule learning in Web IE. The authors claim that their system requires a small amount of labeling by the users:

labeling only one record suffices. Other records are found by iteratively align adjacent records. However, this approach only achieves 73% accuracy since it considers only two adjacent records at a time while OLERA takes all records into consideration by multiple string alignment.

8 Conclusion and Future Work

In this paper, we propose a new approach – OLERA to handle a richer set of semi-structured documents. Our approach not only deal with single-record pages but also multi-record pages. The proposed operations: enclosing relevant block, drilling down and specifying relevant information slots can greatly reduce users' burden for annotation.

Encoding scheme plays an important in alignment. For a good alignment, we need two encoding schemes: one for representing the higher-level structure and the other for text comparison. Though we have default encoding hierarchy, it may require adjustment for individual sources. Therefore, we hope to design an algorithm to analyze the relationships of delimiters and decide what delimiters to use for high-level structure presentation and text comparison in each alignment.

Acknowledgement

This work is sponsored by National Science Council, Taiwan under grant NSC90-2213-E-008-042.

References

[1] Joint vecture template fill rules. In *Plenary Session Notebook of the TIPSTER 12-month Meeting*, 1992.

[2] N. Ashish and C. Knoblock. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):8–15, 1997.

[3] M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1997.

[4] C.-H. Chang and S.-C. Lui. Iepad: Information extraction based on pattern discovery. In *Proceedings of the 10th International Conference on World Wide Web*, pages 681–688, Hong-Kong, May 2–6 2001.

[5] C.-H. Chang, S.-C. Lui, and Y.-C. Wu. Applying pattern mining to web information extraction. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 4–15, Hong-Kong, Apr. 16–18 2001.

[6] C.H. Chang, C.N. Hsu, and S.C. Lui. Automatic information extraction from semi-structured web pages by pattern discovery. *Decision Support Systems Journal*, To appear, 2002.

[7] B. Chidlovskii, J. Ragetli, and M. Rijke. Automatic wrapper generation for web search engines. In *Proceedings of the 1st International Conference on Web-Age Information Management (WAIM'2000)*, LNCS Series, Shanghai, China, 2000.

[8] H. Cunningham. Information extraction – a user guide. Technical Report CS-97-02, Institute for Language, Speech and Hearing (ILASH) and Dept. of Computer Science, University of Sheffield, UK, 1997.

[9] R.B. Doorenbos, O. Etzioni, and D.S. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 39–48, NewYork, USA, 1997.

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

[11] D. Freitag. Information extraction from html: Application of a general machine learning approach. In *Proceedings of the Fifteenth national Conference on Artificial Intelligence*, pages 517–523, 1998.

[12] R. Grishman and B. Sundheim. Message understanding conference – 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Jun 1996.

[13] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.*, 55:141–154, 1993.

[14] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge, 1997.

[15] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

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

[17] C. Knoblock, S. Minton, and et al. J. Ambite. Modeling web sources for information integration. In *Proceedings of the 15th National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*, pages 211–218, Wisconsin, USA, 1998.

[18] N. Kushmerick. Gleaning the web. *IEEE Intelligent Systems*, 14(2):20–22, Mar/Apr 1999.

[19] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, Japan, 1997.

[20] W.-Y. Lin and W. Lam. Learning to extract hierarchical information from semi-strucutred documents. In *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management*, pages 250–257, VA, USA, 2000.

[21] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the 3rd International Conference on Autonomous Agents*, pages 190–197, Seattle, WA, 1999.

[22] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816, 1993.

[23] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thiteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1996.

[24] A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *Data and Knowledge Engineering*, 36(3):283–316, 2001.

[25] S. Soderland. Learning to extract text-based information from the world wide web. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 233–272, CA, USA, 1997.