**Proceedings of the IASTED International Conference**
**ARTIFICIAL INTELLIGENCE AND APPLICATIONS**
**February 16-18, 2004, Innsbruck, Austria**

# OLERA: OnLine Extraction Rule Analysis for 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**
Information extraction (IE) from semi-structured Web documents plays an important role for a variety of information agents. Over the past decade, researchers have developed a rich family of generic IE techniques based on supervised approach which learn extraction rules from user-labelled training examples. However, annotating training data can be expensive when a lot of data sources need to be extracted. In this article, we introduce annotation-free IE using pattern mining and string alignment techniques. We describe OLERA, a semi-supervised IE system that produces extraction rules by aligning similar contents of multiple input records together and presents the result in a spreadsheet-like table. Therefore, users do not need to annotate the input documents but only to specify the scheme for the extracted data after the extraction pattern is discovered. Another plus is that this approach works not only for multi-record Web pages (as a limitation of some unsupervised IE approaches) but also single-record Web pages.

**KEY WORDS**
information extraction, semi-structured documents, string alignment, approximate matching

## 1 Introduction

The explosive growth and popularity of the world-wide web has resulted in a huge number of information sources on the Internet. However, due to the heterogeneity and the lack of structure of Web information sources, information integration systems and software agents often require a lot of efforts for manipulation among various data formats. The problem to translate the contents of input documents into structured data is called **information extraction** (IE), and the problem of information extraction *from a Web page* is to apply information extraction to Web pages. Unlike information retrieval (IR), which concerns how to identify relevant documents from a collection, information extraction produces structured data ready for post-processing, which is crucial to many applications of Web mining and searching tools.

Tailoring an IE system to new requirements is a task that varies in scale dependent on the text type, domain, and scenario [5]. 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 [8]. Traditional IE involves free texts that are written in natural language and fully unstructured. Web IE, on the other hand, process online documents that are marked by HTML tags and presented in semi-structured format (e.g. Web pages generated by CGI scripts). Free-text information extraction roots from linguistic analysis and message understanding [15], whereas Web IE often relies on landmark identification marked by HTML tags and other delimiters.

Programs that perform the tasks of information extraction are referred to as wrappers. A wrapper is a procedure, specific to a single information resource, that translate the input into relational form. Wrappers can be hand-coded in general programming language or specialized languages, or they can be produced via wrapper generators. Wrapper generators are software tools that generate wrappers via induction. A typical wrapper induction system receives labelled training examples which "tell" the IE system what to extract. Previous researches, e.g. WIEN [10], Softmealy [9], Stalker [13], focus on rule generalization and wrapper architecture design, and leave the problem of obtaining labelled training examples to some oracles. As labelling training examples are tedious, recent researches have focused on developing tools that can reduce labelling effort. For instance, several researches propose supervised interactive wrapper generation tools for rule generalization and writing [14, 11, 1]. Chidlovskii, et al, design a wrapper generation system that requires a small amount (one training record) of labelling by the user [4]. Some researches develop object/record extraction systems that recognize record boundaries without labelling, e.g. [6, 2]. However, the assumption of such systems is that the input is a multi-record Web page. If the training page contains only one record (called singular pages), these approaches simply fail.

This paper provides a novel framework – OLERA, for training extraction rules from Web pages with arbitrary number of records. Most of all, labelling 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

attribute of the record. Extraction rules are learned by approximate pattern matching and string alignment techniques in a way similar to how people generalize rules. The experiments are conducted for both multi-record pages and single-record pages. In addition, OLERA can be easily adapted to different document sets.

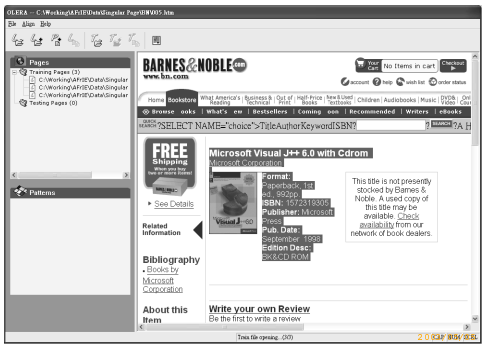
The remainder of the paper is organized as follows. Section 2 reviews background material on information extraction. Section 3 describe OLERA's system framework and the implementation details. Section 4 gives the implementation of the OLERA extractors. Experimental results are shown in Section 5. Finally, Section 6 concludes the paper.

## 2 Background and Motivation

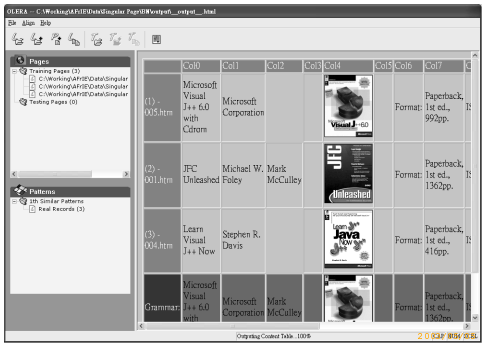
Information Extraction (IE) is concerned with extracting relevant data from a collection of documents. The goal is not necessarily to produce a general-purpose IE system, but to create tools that would allow users to build customized IE systems quickly. Web IE is an important problem for information integration and many approaches have been proposed with various degree of automation. Related work on IE has been studied and compared in [12], where the author discusses IE systems for free text and online documents. One of the comparing metrics discussed in this paper is whether the IE systems generate single-slot or multi-slot extraction rules. Multi-slot extraction rules are more difficult to generate because the input can contain missing attributes or several attribute permutations. If the input is a single-record document, we can simply design IE systems that generate single-slot extraction rules for each attribute. For multi-record documents, we need multi-slot extraction rules or the documents have to be segmented in advance. Therefore, multi-record input can be a problem for IE systems that generate only single-slot rules. However, multi-record input can be an advantage for some IE systems [2, 6], for they rely on this characteristic for record/object identification.

Another metric is rule expression: some use extraction patterns based on syntactic/semantic constraints, some use delimiter-based constraints, and still some use both constraints. For example, most semi-structured IE systems use delimiter-based constraints from HTML tags. Some IE systems, e.g. SRV [7] imposes content constraints based on orthographic features, part-of-speech tags, WordNet semantic classes, etc. These various constraints reflect the common features that present in the input and can be generalized by rule learners. Since it relies on system developers' knowledge to add such features, it is desirable that such background knowledge is separate from the underlying learning mechanism for rapid adaptation.

In addition to these metrics discussed in [12], we can also compare IE systems from their input. Most machine-learning based IE systems require precise annotation (including the boundaries for each field in the object/record) of the extraction targets as training examples because they



(a)



(b)

Figure 1. Singular Web pages – Barnes&Noble (a) enclosing one record and (b) the corresponding result.

are supervised learning approaches. However, labelling is often a burdensome task even with the help of graphic user interface design. Therefore, some approaches have been proposed to alleviate the labelling task. For example, Kushmerick, et al. use heuristic knowledge for automatic labelling [10]. Chidlovskii, et al. design a wrapper generation system that requires a small amount of labelling by the user [4]. Besides these supervised IE systems, more interesting works are unsupervised IE systems that recognize record boundaries without labelling, e.g. [2, 6]. For such systems, relevant slots will be chosen by the user through an `attribute designation` operation [3]. Note that attribute designation is different from labelling and requires less human effort since it is operated after the extraction pattern is discovered, while labelling is conducted before the extraction rule is generated. However, current unsupervised IE systems can only work on multi-record documents. If the training page contains only one record, these approaches simply fail.

In summary, designing IE systems that can reduce labelling efforts is the main research stream. Therefore, it

is our goal to build an IE system that requires minimum labelling effort and works for both single record or multi-record HTML pages. In addition, it is desirable that the interpretation of the input document sets can be separate from the underlying learning mechanism for rapid adaptation.

## 3 System Framework

In this section, we present the framework of our novel approach, called OLERA. The main goal of this system is to learn extraction rules for pages containing single records, but with no annotation information. One possible way to achieve this goal is to include multiple pages and align them for attribute designation as in IEPAD. However, aligning the whole pages is much difficult and it often results in too many slots which is inconvenient for attribute designation. Therefore, we adopt an eclectic approach where users enclose a data block to indicate his/her interest. Unlike traditional learning approaches, users do not need to annotate each interested data block in the training pages or the exact locations for each attribute in the record. Instead, the system would automatically discover other records that are similar to the enclosed example and present the data in form of a spreadsheet for attribute designation.

To illustrate, in Figure 1(a), three singular pages (each containing one record) are given as the training set. Instead of enclose each data block from the training pages, we only enclose a block of book information from one training page for analysis. The corresponding result of this operation is as shown in Figure 1(b), where two other books from the training pages are discovered automatically. The records are then presented in rows of a spreadsheet where information of the same kind, say, book image, is aligned at the same column. The spreadsheet presentation is designed for easy understanding such that users can designate relevant slots for each attribute in a record.

The idea of enclosing a data block for extraction rule analysis not only works for singular pages but also for multi-record pages. In addition, the training pages are not necessarily all singular or multi-record pages. We may choose one singular page for the enclosing operation, and take other multiple-record 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 multi-record, to recognize other records in the training set.

Spreadsheet presentation and the concept of attribute designation were introduced in IEPAD [3]. In this paper, we further introduce two operations: drill-down and roll-up operations for users to manipulate the information of interest. The inclusion of these two operations offers a summarized or detailed view of the data. For example, Figure 2 demonstrates the drill-down operation on the 7th column of Figure 1, where the text "Paperback, 1st ed., 992pp." is further divided into three components "Paperback", "1st ed." and "992pp." by delimiter ",". In addition, column 3 to column 5 are combined to demonstrate the roll-up operation. Finally, information slots of interest can then be specified by the checkbox above each column and saved for later use by OLERA extractors.

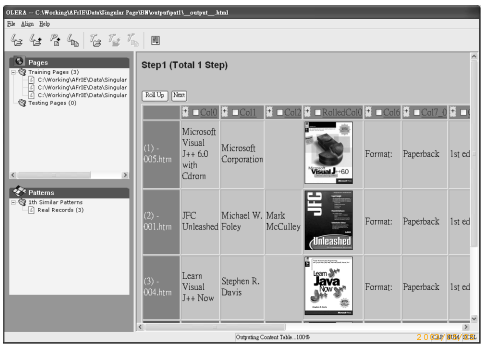



Figure 2. Drill down operation on column 7 from Figure 1

In summary, OLERA provides three kinds of operations: enclosing, drill-down/roll-up, and attribute designation for users to manipulate the information of interest. Formal description of these operations and the corresponding actions by OLERA are given below:

- Enclosing an information block of interest
  Users can mark a block to indicate the boundaries of a record. The enclosed block is the target where extraction rule analysis is carried out. In response to this operation, OLERA automatically discover other records that are similar to the enclosed one and arrange information of the same kind in the same column, finally present the data in a spreadsheet for users.

- Drilling-down/rolling-up an information slot
  Drill-down operations allow users to navigate from a text fragment to more detailed components, while roll-up operations allow the combinations of several slots to form a meaningful information unit. Drill-down operations are realized by translating the text fragments of the same slot according to a specified encoding scheme and aligning all encoded strings by multiple string alignment.

- Designating relevant information slots
  The result of the above operations is presented by a spreadsheet with multiple slots decomposed from the enclosed information block. Users can then specify relevant slot for each attribute to specify the scheme of the extraction target.

Figure 3 shows the framework of the system OLERA. The ellipses denote the operations performed by the users in the training process and the rectangles denotes the corresponding actions taken by OLERA in response to users'

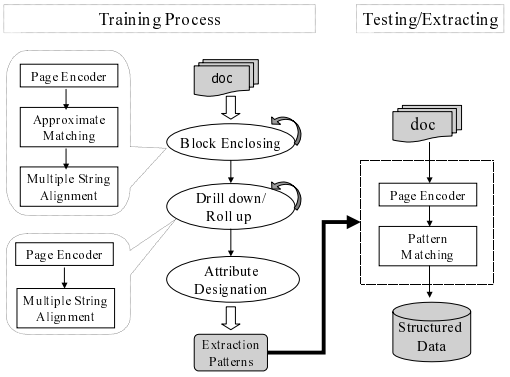



Figure 3. The OLERA's framework

operations. For example, the corresponding procedures for the enclosing operation 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/similar records in training pages by **approximate matching**. Third, align matched records by **multiple string alignment** and present the result in a spreadsheet with $m$ rows (records) and $n$ columns (slots). In the following, we will describe these three procedures one by one.

## 3.1 Encoding Hierarchy

The core technique of OLERA is a well-known algorithm called string alignment. However, aligning different book titles in the same column requires more than comparing characters by characters. To construct the alignment for HTML documents, translation of the pages is performed where each HTML tag is regarded as an individual **delimiter** token and any texts between two delimiter tokens are ignored and represented as a special token called **¡TEXT¿** token. Formally, 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¿**.

In order to construct a desired alignment, it often requires a higher level encoding scheme to abstract the input pages. However, a high level abstraction may not be appropriate for extraction of finer information. Hence, we further introduce the drill-down operation by incorporating an encoding hierarchy as that for multi-dimensional models in OLAP. 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** (see Figure 4). The greater-than sign indicates that the left encoding is a higher level abstraction of the right one. 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

| Level | Encoding scheme | Delimiters |
|---|---|---|
| Markup | Block-level tag | block-level tags |
| | text-level tag | text-level tags |
| Text | Paragraph | NL, CR, Tab |
| | Sentence | . ? ! |
| Word | Phrase | : , ; [ ] ( ) { } |
| | Others | Blank @ $ - / |

Figure 4. The Encoding Hierarchy for Web Documents

scheme which are introduced in IEPAD. For word-level encoding schemes, we concern the constituents of sentences: phrase separated by quotation marks, parenthesis, brackets; words separated by blank spaces, and other symbols such as dollar signs, dashes, slash, etc. Take the drill-down operation in Figure 2 for example. We employ the phrase encoding scheme to encode the three text segments "Paperback, 1st ed., 992pp.", "Paperback, 1st ed., 1362pp.", and "Paperback, 1st ed., 416pp." into the same token string "<TEXT>, <TEXT>, <TEXT>" for alignment.

## 3.2 Approximate Matching

To automatically discover other similar records of the enclosed block in the training set, we use a variant of string comparison called approximate matching. Approximate matching of a pattern $P$ in a text $T$ can be computed by dynamic programming with base conditions

$$\begin{aligned} V(i,0) &= \quad 0; \\ V(0,j) &= \quad -d * j; \end{aligned} \tag{1}$$

and general recurrence

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + match(P[i], T[j]); \\ V(i-1,j) - d; \\ V(i,j-1) - d; \end{cases} \tag{2}$$

where the value $-d$ ($d > 0$) denotes the score of aligning a character with a space and the function $match(x, y)$ denotes the score of aligning two characters. Traditionally, a match ($x = y$) of two characters get a value of $s$ ($> 0$) and a mismatch ($x \neq y$) a value of $-s$. However, it is hard to break ties when several alignments have the same optimal similarity score. Therefore, we enforce the comparison of the primitive data of two <TEXT> tokens to quantify the similarity between 0 to $s$. Here, we define the similarity score as the value of the optimal global alignment of the primitive data. For long primitive data, they are further translated by another encoding scheme before comparison.

We say a substring $T'$ of $T$ is an approximate occurrence of $P$ if and only if the similarity ratio of $P$ and $T'$ is greater than a given threshold $\theta$. Therefore, to discover all approximate occurrence of $P$ in $T$, we first identify the position $j'$ in $T$ such that $V(m, j')$ has the largest

value among all $V(m, j)$, and $V(m, j')$ is greater than $\delta = \theta * s * m$. For this $j'$, output the approximate occurrence, $T[k', j']$ by backtracking from $(m, j')$ until a cell in row zero $(0, k')$ is reached. We then apply this procedure to $T[1, k' - 1]$ and $T[j' + 1, |T|]$ recursively to find all approximate occurrence of $P$ in $T$.

### 3.3 Multiple String Alignment

For those records identified by approximate matching, we need a generalization over these instances. Let's say $k$ token strings are discovered after occurrence identification. We will apply multiple string alignment procedure to the $k$ token strings to generalize the record extraction rule. Multiple string alignment of $k(> 2)$ strings $S_1, S_2, \ldots, S_k$ 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 so that each character and space of each string is in a unique column. For example, suppose we have three token strings "`dtbtbt`", "`dtbt`" and "`dtbat`", a multiple string alignment for them can be illustrated as following:

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

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. For example, the signature representation of the above alignment will be expressed by "`dtb[a|-]t[b|-][t|-]`", which contains 7 columns. Note that the score function can affect the optimal alignment of two strings. For example, if $s$ is greater than $2d$, alignment with spaces is preferable than aligning two different characters. In OLERA, $s$ is always given a value greater than $2 * d$ to prevent a mismatch of two different tokens.

Multiple string alignment has been applied in IEPAD to compensate the insufficiency of PAT trees. The major problem in previous work is the decision of an alignment when multiple alignments have the same similarity score. Therefore, a large number of patterns can be produced. In this paper, the proposed matching function for comparing the primitive data of <TEXT> tokes is used to avoid such a problem and find a best alignment.

### 3.4 Summary

In summary, for each enclosing operation, OLERA identifies approximate occurrences (after encoding) and generalizes them to an extraction pattern by multiple string alignment. A drill-down operation on a column involves two steps including encoding and multiple string alignment over the contents 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 recorded in the extraction rule such that the extractors of OLERA can perform the same encoding and extraction as recorded.

## 4 The OLERA Extractors

Similar to IEPAD, 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 and a signature representation for the pattern.

An OLERA extractor follows the encoding schemes specified in the record grammar to translate a testing page. It then searches for an instance of a record pattern $R$ in the encoded token string. For example, signature "$a[b|-]c$" can match only "abc" or "ac". The search for an instance of the record pattern $R$ can be solved by *regular expression* pattern matching. However, regular expression matching lacks the comparison of the primitive data for `TEXT` tokens and requires additional alignment after matching, therefore we do not use this approach. Instead, we adopt a variation of approximate matching with two differences. First, each slot $R[i]$ can contain both tokens and spaces since the record pattern $R$ is now a signature representation. Second, no insertion or deletion can be made to the signature patterns except indicated in the signature patterns. To enforce the restriction, the recurrence for the dynamic programming is revised as

$$V(i,j) = \max \begin{cases} V(i-1, j-1) + s; & \texttt{if} \quad T[j] \in R[i] \\ V(i-1, j) + s; & \texttt{if} \quad space \in R[i] \\ -Max; & \texttt{o/w} \end{cases} \tag{3}$$

with base condition

$$\begin{aligned} &V(0,j) = 0; \\ &V(i,0) = \max \begin{cases} V(i-1, 0) + s; & \texttt{if} \quad space \in R[i] \\ -Max; & \texttt{o/w} \end{cases} \end{aligned} \tag{4}$$

Note that $Max$ is a large value which is used to indicate no proper token is matched with $R[i]$ and any value added to $-Max$ still results in $-Max$. 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.

## 5 Experiments

OLERA has been tested on a set of 24 real world Web sites. Ten of them are single-record pages. The others are multi-record pages. Some of the information sources have been used in WIEN and STALKER (like Bigbook, IAF, OKRA, QuoteServer). A total of 2887 pages are collected for experiments. The average number of attributes in a record is 4.8 and 8.0 for multi-record and single-record pages, respectively. Some of the data source contains missing attributes, or several permutations, or list attributes (e.g. book's authors). A total of 2964 pages are collected for experiments.

To wrap a data source, we start with one randomly chosen page (two for single-record data sources), and enclose one example record to approximate other records in the training set. If not all records in the training set are discovered, similarity threshold can be reduced to approximate more records. When all records in the training set are correctly extracted, the extraction rule is then applied to other unseen pages for testing. If not all records in the testing pages are extracted, another page that contains such records is added in the training set for training. Repeat the same procedure for 3, 4, ... training pages until all testing records are successfully extracted. This procedure is repeated for 3 times and the numbers of training pages are averaged for each data source. Extraction performances are then evaluated by retrieval rate (recall) and accuracy rate (precision).

For some cases, a CGI script can have several display templates to enhance the visualization for different products (e.g. ByuBook). In this case, it is impossible for one enclosed example to match all records presented in different formats with reasonable similarity threshold $\theta$. Therefore, OLERA allows multiple enclosing to solve this problem. For each enclosed example, the system identifies its approximate occurrences in the training example. If a text segment is similar to several enclosed examples, the most similar one will be chosen. Similarly, if two enclosed examples match two text segments that are mostly overlapped, such text segments are considered the same and the above rule applies as well.

### 5.1 Singular pages

Table 1 shows the number of training pages required to achieve best retrieval rate or accuracy rate for single-record pages. For some data source (e.g. AlBook, IUniverse), only a few training pages are needed to achieve 100% retrieval rate while some (e.g. Amazon, Barnes&Noble, E-bay) require more training pages. The number of enclosing operations (Pat) used for the last training and the similarity threshold (Sim) are also recorded. We found that 0.5 (default) is a good similarity threshold for most data sources, since it can most identify approximate occurrences of the example enclosed. Therefore, only one enclosing operation is used for most data sources. The data source, ByuBook, contains three presentation formats where missing attributes might occur. Therefore, three enclosing operations and average 4.3 training pages are needed to extract all instances.

| Source | Pre | Rec | Pages | Pat | Sim | Len |
|---|---|---|---|---|---|---|
| Amazon | 100 | 100 | 12.0 | 1 | 0.5 | 158 |
| AlBooks | 95.7 | 100 | 3.3 | 1 | 0.5 | 168 |
| BarnesNoble | 100 | 100 | 17.0 | 1 | 0.45 | 116 |
| BookPool | 100 | 100 | 6.0 | 1 | 0.5 | 61 |
| ByuBook | 100 | 100 | 4.3 | 3 | 0.7 | 36 |
| Ebay | 100 | 100 | 10.7 | 1 | 0.5 | 466 |
| IUniverse | 100 | 100 | 1.7 | 1 | 0.5 | 85 |
| JulliardBook | 100 | 100 | 6.3 | 1 | 0.5 | 72 |
| PMIBook | 100 | 100 | 1.0 | 10 | 0.5 | 4 |
| Zagat | 100 | 100 | 5.0 | 1 | 0.5 | 44 |
| Average | 99.6 | 100 | 6.7 | 2.1 | 0.52 | 121 |

Table 1. Performance for single-record pages

The last column of Table 1 records the pattern length which is defined as the number of tokens for the encoded block. This information is presented here as a comparison to the number of attributes that we want to extract and the pattern length for multi-record pages presented next. The pattern length for Barnes&Noble is 116 tokens even though we are interested only in 8 slots. For longer patterns, they often present more changes in the data structure. Therefore, they require a lot more training pages than other data sources. Amazon, Barnes&Noble, and E-Bay are three of such examples.

### 5.2 Multi-record pages

Table 2 shows the parameter setting of the final experiment for multi-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 multi-record pages is comparably smaller than that for singular pages, there are less variations, too. This can be seen from pattern length too. To illustrate, the average length of a record is 121 tokens for singular pages, while the average length for multi-record pages is 20 tokens. For these 14 information sources, thirteen can all be perfectly extracted as in IEPAD [3] except for IAF.

## 6 Conclusion

In this paper, we propose a new approach – OLERA for information extraction from unlabelled training set. This approach not only deal with single-record pages but also multi-record pages. It can also be easily adapted for non-HTML documents by adding new encoding schemes since the underlying learning mechanism is independent of the

| Source | Pre | Rec | Pages | Pat | Sim | Len |
|---|---|---|---|---|---|---|
| ∗AltaVista | 100 | 100 | 2 | 1 | 0.6 | 19 |
| DirectHit | 100 | 100 | 3 | 1 | 0.45 | 11 |
| ∗Excite | 100 | 100 | 1 | 1 | 0.5 | 10 |
| HotBot | 100 | 100 | 7 | 1 | 0.45 | 17 |
| Infoseek | 100 | 100 | 4 | 1 | 0.5 | 27 |
| MSN | 100 | 99 | 1 | 1 | 0.5 | 9 |
| ∗NorthernLight | 100 | 100 | 1 | 1 | 0.5 | 34 |
| Sprinks | 100 | 99 | 4 | 1 | 0.45 | 16 |
| Webcrawler | 100 | 100 | 1 | 1 | 0.7 | 13 |
| ∗Yahoo | 100 | 97 | 1 | 1 | 0.5 | 12 |
| OKRA | 100 | 100 | 1 | 1 | 0.5 | 35 |
| BigBook | 100 | 100 | 1 | 1 | 0.5 | 30 |
| IAF | 86 | 86 | 4 | 1 | 0.5 | 9 |
| Quote Server | 100 | 100 | 3 | 1 | 0.5 | 53 |
| Average | 99 | 99 | 2.4 | 1 | 0.51 | 21 |

* denotes block-level encodings are used.

Table 2. Performance for multi-record pages

encoding schemes used. OLERA distinguishes itself from other IE systems by its simplicity for extraction rule learning. The proposed operations: enclosing relevant blocks, drilling down and specifying relevant information slots can greatly reduce users' burden for annotation. Because of this simplicity, OLERA is very efficient, which makes it perfect for online (CGI-generated) document extraction.

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 and Education Bureau, Taiwan under grant 甲 92-H-FA07-1-4.

## References

[1] R. Baumgartner, S. Flesca, and G. Gottlob. Supervised wrapper generation with lixto. In *Proceedings of VLDB Demo*, 2001.

[2] 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, 2001.

[3] 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*, 35(1):129–147, 2003.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] L. Liu, C. Pu, and W. Han. Xwrap: An xml-enabled wrapper construction system for web information sources. In *Proceedings of ICDE*, 2000.

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

[13] 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.

[14] A. Sahuguet and F. Azavant. Building light-weight wrappers for legacy web data-sources using w4f. In *Proceedings of VLDB*, 1999.

[15] 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.