OLERA: Semisupervised Web-Data Extraction with Visual Support

Chia-Hui Chang, *National Central University, Taiwan*

Shih-Chien Kuo, *Trend Micro, Taiwan*

The World Wide Web's explosive growth and popularity has resulted in countless information sources on the Internet. However, due to the heterogeneity and lack of structure in Web information sources, information-integration systems and software agents—and sometimes humans as well—must expend a great deal of effort when manipulating various data formats. The problem of translating the content of input documents into structured data is called information extraction.

OLERA is a semisupervised information-extraction system that produces extraction rules from semistructured Web documents without requiring detailed annotation of the training documents. It performs well for program-generated Web pages with few training pages and limited user intervention.

An IE task is defined by its extraction target and input. Its extraction target is generally considered a relation of *k*-tuple, where *k* is the number of attributes in a record of the (desired, expected) data. An attribute may have zero (missing) or multiple instantiations in a record, and the extraction task will fill either a single slot (where *k* equals 1) or multiple slots. Programs that perform IE tasks are referred to as *extractors* or *wrappers*. A wrapper is generally a pattern-matching procedure that relies on a set of extraction rules. The simplest way to produce extractors is to have a human observe the input documents and write extraction rules, but this requires a certain degree of programming expertise. It's also time consuming, error prone, and not scalable.

IE systems can generate wrappers that can receive input documents and convert them into structured data. We can categorize most IE systems (such as WIEN (*W*rapper *I*nduction *En*vironment),[1] Softmealy,[2] and Stalker[3]) as supervised machine learning, because they require "labeled training examples" to tell the IE system what constitutes a record. By comparing the preceding and succeeding strings of several extraction examples, IE systems can learn the common landmarks as extraction patterns for each attribute and the record boundary. However, the labeled training examples require users to annotate the input documents, which can be tedious even for a small corpus of training documents. IE systems that use unlabeled training examples are comparatively interesting but can only accept specific kinds of input such as program-generated pages under certain assumptions.· (See the sidebar for a more detailed comparison of these approaches.) We propose a semisupervised IE system—*O*n-*L*ine *E*xtraction *R*ule *A*nalysis—that lets users, with minimal effort, train extraction rules from Web pages. OLERA offers visual interaction by displaying discovered records in a spreadsheet-like table for schema assignment.

System framework

We introduce OLERA from the users' viewpoint—that is, we explain how users interact with OLERA to generate extraction rules for their interested targets. Instead of labeling training pages, users enclose an information block of interest and then specify relevant information slots for each field in the record (see Figure 1).

Enclosing a data block

Given a set of training pages, an OLERA user first encloses a block that's large enough to contain one record of interest as an example. The user doesn't need to label the block's detailed subsegments to indicate the locations of titles, authors, or prices. The labeling work is delayed until OLERA generates the extraction pattern. In addition, the user needn't enclose every record of interest in the training page. The system can automatically discover other records that resemble the enclosed example and present the data in a spreadsheet for attribute designation.

To illustrate, suppose we're interested in the main search result for Christmas songs. We can enclose

Comparing Various Approaches

Web *information extraction* is an important problem for information integration, and many approaches have been proposed. We can categorize these approaches based on the IE task that various IE systems address or according to the techniques they use.[1] In this article, we compare the approaches from users' viewpoints and explore what background knowledge different IE systems require.

One category refers to IE systems that require users who have programming expertise. These include languages or toolkits designed for wrapper development—for example, W4F and XWrap. Researchers proposed such languages or toolkits as alternatives to general-purpose languages because users could concentrate on formulating the extraction rules without concerning the input string's process. In other words, users of these IE systems must be trained to understand the language and be able to generalize extraction rules by inspecting and writing them using the designed languages or tools.

Another category refers to IE systems that require users to label extraction targets as examples for IE systems to construct extraction rules. Therefore, such IE systems don't require any programming. Many IE systems—including WIEN, Softmealy, and Stalker—belong to this category. Compared to the first category, these IE systems are preferable because general users, instead of programmers, can be trained to use the IE systems for wrapper construction.

A third category refers to IE systems that don't require users to preprocess the input documents. We call them annotation-free IE systems. Developing such systems is based on one important characteristic—that these input pages are generated using a common template by "plugging in" values from an underlying structured source such as a relational database (for example, an advertisements database or book database). Example systems include IEPAD,[2] RoadRunner,[3] and EXALG.[4] Because users don't specify extraction targets, these systems make presumptions about the data to be extracted. For example, IEPAD assumes the existence of multiple tuples to be extracted in one page, so its approach is to discover repeated patterns with regularity and vicinity. With such an assumption, IEPAD can only process multirecord pages. RoadRunner and EXALG assume that any plug-in values are the extraction targets. However, because commercial Web pages often contain multiple topics where a great deal of information is embedded for navigation, interaction, and advertisement, irrelevant information as well as relevant information will be extracted. In other words, it's difficult to judge whether a token (or a piece of information) is a data value or template. The same problem has caused the intervention of IEPAD users for selecting relevant patterns. After all, what's relevant or of interest is quite subjective. So, annotation-free IE systems, although they remove the labeling work before training, require postprocessing because their assumptions might not sustain for all cases.

References

1. A.H.F. Laender et al., "A Brief Survey of Web Data Extraction Tools," *SIGMOD Record*, vol. 31, no. 2, 2002, pp. 84–93.
2. C.-H. Chang and S.-C. Lui, "IEPAD: Information Extraction Based on Pattern Discovery," *Proc. 10th Int'l Conf. World Wide Web* (WWW 01), ACM Press, 2001, pp. 681–688.
3. V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: Towards Automatic Data Extraction from Large Web Sites," *Proc. 27th Int'l Conf. Very Large Data Bases* (VLDB 01), Morgan Kaufmann, 2001, pp. 109–118.
4. A. Arasu and H. Garcia-Molina, "Extracting Structured Data from Web Pages," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 03), ACM Press, 2003, pp. 337–348.

one record block and add it as an example (see the highlighted area in Figure 2a). Then the system discovers 10 records from the training pages and presents them in rows in a spreadsheet such that it aligns similar information—such as shipping information and list price—in the same column (see Figure 2b). For CDs with no in-store pickups, null strings are presented.

Enclosing a data block for extraction-rule analysis works not only for Web pages but also for non-HTML semistructured documents such as those from dbEST (Expressed Sequence Tags database—one of the Genbank databases hosted by the National Center for Biotechnology Information). Figure 3a shows an example of the dbEST files, which are pure text files formulated by delimiters such as tabs and new lines. Figure 3b shows the result of enclosing one whole file for analysis. As we

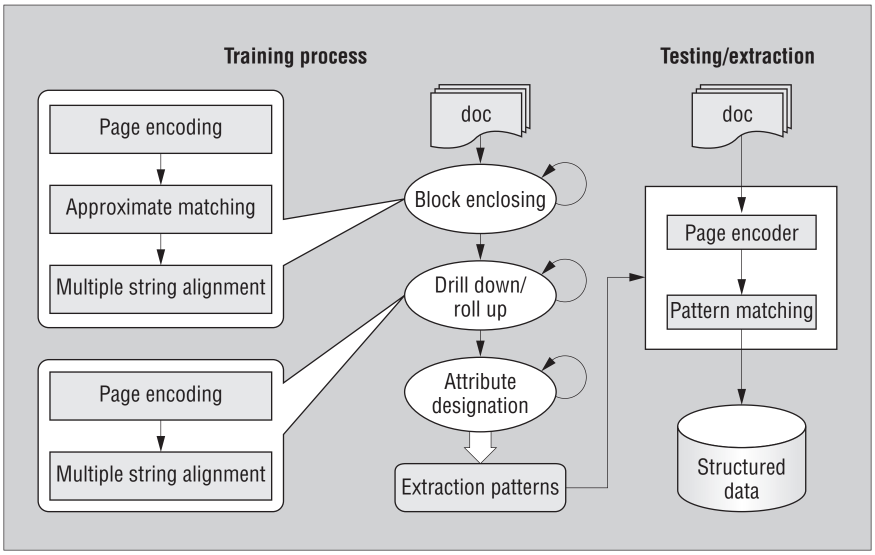
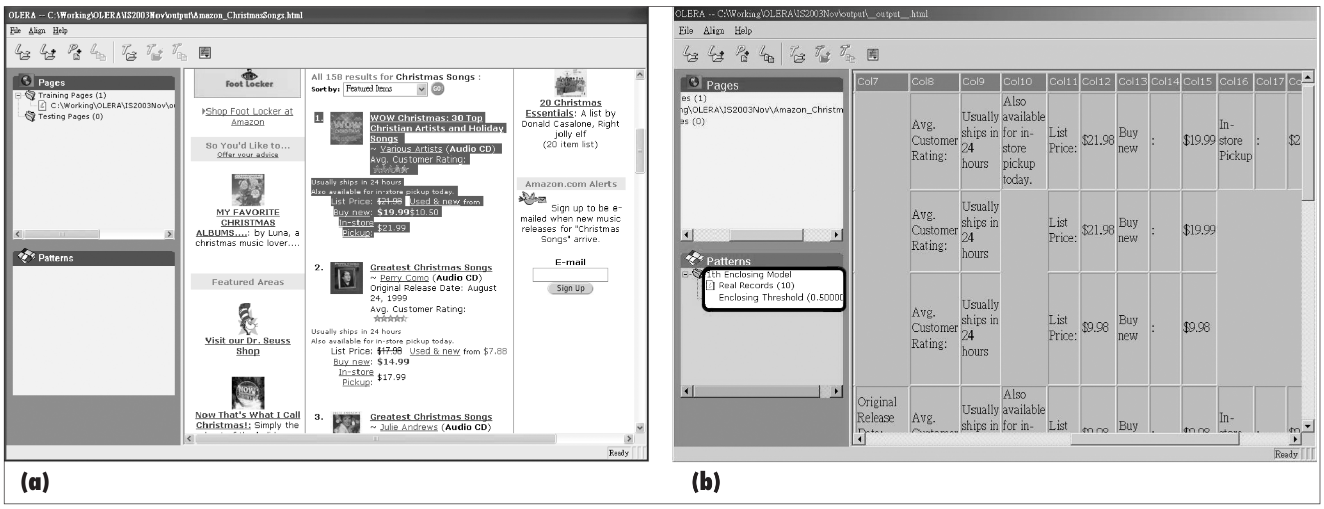


Figure 1. The OLERA framework. Ovals denote the operations that users perform during training; rectangles denote OLERA's actions in response to the users' operations.



Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14	Col15	Col16	Col17	Co
	Avg. Customer Rating:	Usually ships in 24 hours	Also available for in-store pickup today.	List Price:	$21.98	Buy new	:	$19.99	In-store Pickup	:	$2
	Avg. Customer Rating:	Usually ships in 24 hours		List Price:	$21.98	Buy new	:	$19.99			
	Avg. Customer Rating:	Usually ships in 24 hours		List Price:	$9.98	Buy new	:	$9.98			
Original Release	Avg.	Usually ships in	Also available for in-	List		Buy			In-		

(a) **(b)**

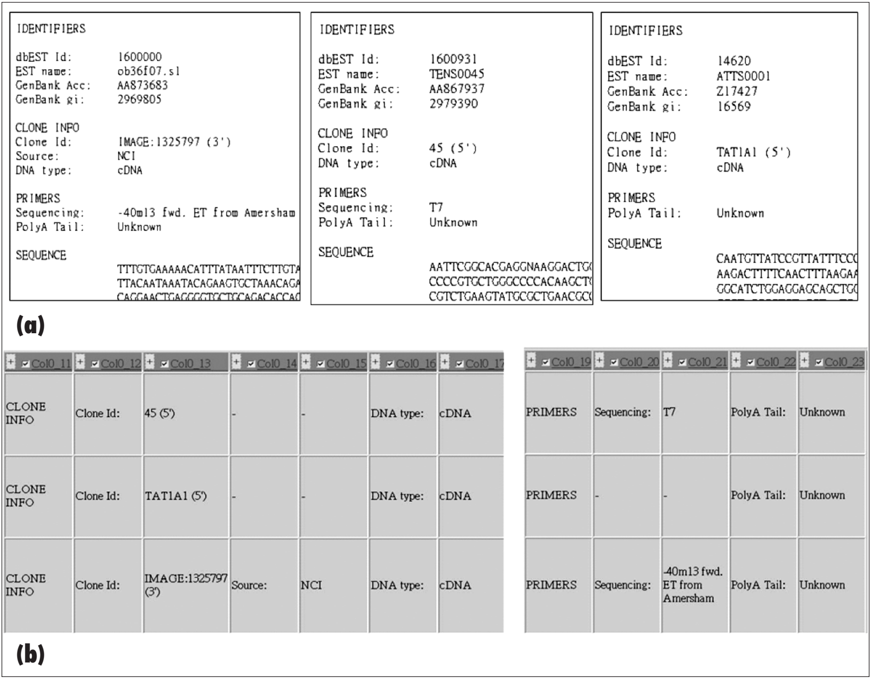
Figure 2. (a) Enclosing one record for analysis; (b) OLERA finds 10 records and aligns them in a spreadsheet (only the top four are shown).

can see, the training pages can contain a single record or multiple records (a page containing one tuple of interest is a single-record page and a page containing a list of tuples is a multirecord page). As long as the user supplies one example record, OLERA can find other similar records through approximate occurrence matching in the training page.

Specifying relevant information slots

Spreadsheet presentation helps users give attribute names to relevant slots in a record. Two additional operations, *drill down* and *roll up*, let users manipulate this information, providing a summarized or detailed view of the data. For example, Figure 4 demonstrates a drill-down operation where a user-specified comma delimiter divides the text strings under column 18 (see Figure 4a) and aligns them so that the first two text strings ("Paperback, 1000pp." and "Paperback, 750pp.") are divided into two text segments and "Hardcover, 1st ed., 488pp." is divided into three text segments (see Figure 4b). This aligns information of the same kind for attribute designation. You implement roll up, the inverse of drill down, by simply concatenating strings from selected columns.

Finally, the user can then specify information slots of interest using the checkbox above each column (see Figure 4b), saving it for later use by OLERA extractors.



```
IDENTIFIERS

dbEST Id:       1600000
EST name:       ob36f07.s1
GenBank Acc:    AA873683
GenBank gi:     2969805

CLONE INFO
Clone Id:       IMAGE:1325797 (3')
Source:         NCI
DNA type:       cDNA

PRIMERS
Sequencing:     -40m13 fwd. ET from Amersham
PolyA Tail:     Unknown

SEQUENCE
                TTTGTGAAAAACATTTATAATTTCTTGTA
                TTACAATAAATACAGAAGTGCTAAACAGA
                CAGGAACTGAGGGGTGCTGCAGACACCAC
```

```
IDENTIFIERS

dbEST Id:       1600931
EST name:       TENS0045
GenBank Acc:    AA867937
GenBank gi:     2979390

CLONE INFO
Clone Id:       45 (5')
DNA type:       cDNA

PRIMERS
Sequencing:     T7
PolyA Tail:     Unknown

SEQUENCE
                AATTCGGCACGAGGNAAGGACTGC
                CCCCGTGCTGGGCCCCACAAGCTC
                CGTCTGAAGTATGCGCTGAACGCC
```

```
IDENTIFIERS

dbEST Id:       14620
EST name:       ATTS0001
GenBank Acc:    Z17427
GenBank gi:     16569

CLONE INFO
Clone Id:       TAT1A1 (5')
DNA type:       cDNA

PRIMERS
PolyA Tail:     Unknown

SEQUENCE
                CAATGTTATCCGTTATTTCCC
                AAGACTTTTCAACTTTAAGAA
                GGCATCTGGAGGAGCAGCTGC
```

(a)

+ ✓ Col0_11	+ ✓ Col0_12	+ ✓ Col0_13	+ ✓ Col0_14	+ ✓ Col0_15	+ ✓ Col0_16	+ ✓ Col0_17
CLONE INFO	Clone Id:	45 (5')	-	-	DNA type:	cDNA
CLONE INFO	Clone Id:	TAT1A1 (5')	-	-	DNA type:	cDNA
CLONE INFO	Clone Id:	IMAGE:1325797 (3')	Source:	NCI	DNA type:	cDNA

+ ✓ Col0_19	+ ✓ Col0_20	+ ✓ Col0_21	+ ✓ Col0_22	+ ✓ Col0_23
PRIMERS	Sequencing:	T7	PolyA Tail:	Unknown
PRIMERS	-	-	PolyA Tail:	Unknown
PRIMERS	Sequencing:	-40m13 fwd. ET from Amersham	PolyA Tail:	Unknown

(b)

Figure 3. (a) Three non-HTML documents (dbEST files) and (b) OLERA's alignment result.

The algorithms

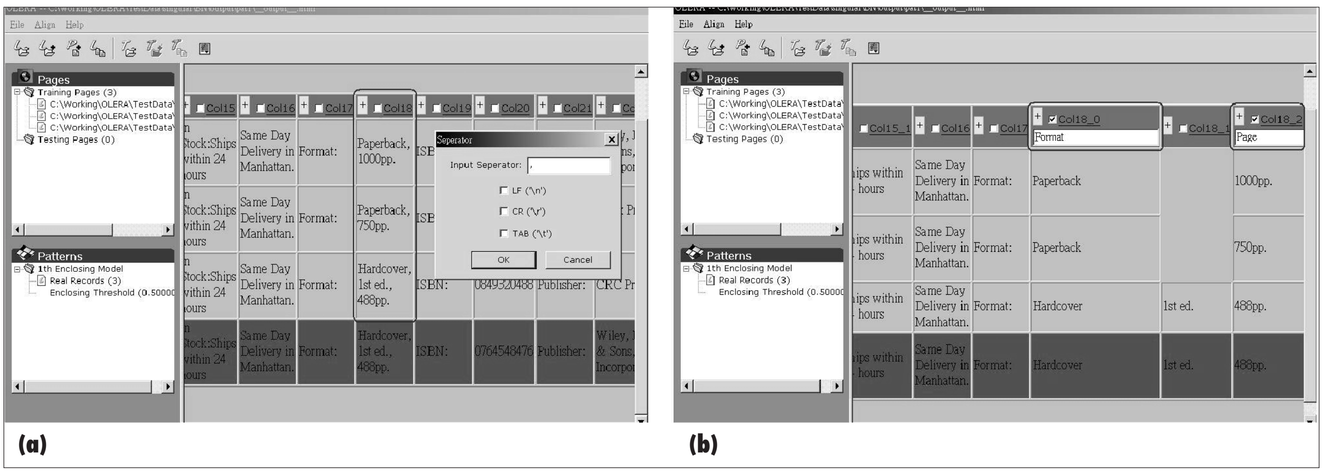
The procedures for enclosing a block comprise three steps (see Figure 1):

1. Translate the training page using an encoding scheme in the encoding hierarchy.
2. Match the pattern of the enclosed block to discover similar records in training pages by approximate matching.
3. Align matched records through multiple string alignment and present the results in a spreadsheet with *m* rows (records) and *n* columns (slots).

These steps are the main procedures needed in OLERA's training process because drill-down operations also use encoding and multiple string alignment. Here, we describe the algorithms for these three steps.

Translating the page

OLERA's core technique is a well-known technique called string alignment. However, aligning different book titles, for example, in the same column requires more than comparing characters with other characters. To align HTML documents, the system first translates the pages, regarding each HTML



Input Seperator: ,

☐ LF ('\n')
☐ CR ('\r')
☐ TAB ('\t')

Col15	Col16	Col17	Col18	Col19	Col20	Col21	Co
n Stock:Ships within 24 hours	Same Day Delivery in Manhattan.	Format:	Paperback, 1000pp.	ISB			y, J ns, por
n Stock:Ships within 24 hours	Same Day Delivery in Manhattan.	Format:	Paperback, 750pp.	ISB			Pr
n Stock:Ships within 24 hours	Same Day Delivery in Manhattan.	Format:	Hardcover, 1st ed., 488pp.	ISBN:	0849320488	Publisher:	CRC Pr
n Stock:Ships within 24 hours	Same Day Delivery in Manhattan.	Format:	Hardcover, 1st ed., 488pp.	ISBN:	0764548476	Publisher:	Wiley, J & Sons, Incorpo

Col15_1	Col16	Col17	☑ Col18_0 Format	Col18_1	☑ Col18_2 Page
ips within hours	Same Day Delivery in Manhattan.	Format:	Paperback		1000pp.
ips within hours	Same Day Delivery in Manhattan.	Format:	Paperback		750pp.
ips within hours	Same Day Delivery in Manhattan.	Format:	Hardcover	1st ed.	488pp.
ips within hours	Same Day Delivery in Manhattan.	Format:	Hardcover	1st ed.	488pp.

Figure 4. Analyzing Barnes & Noble pages with three training pages, each containing one record to be extracted: (a) a drill-down operation on the 18th column with a user-specified delimiter (a comma) and (b) attribute name assignment.

tag as an individual delimiter token. It represents any text between two delimiter tokens as a special token called <*TEXT*>. 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>. A <TEXT> token's original text string is called its primitive data.

For example, suppose the user encloses block B1 in Figure 5. Using HTML tags as delimiters, OLERA will encode block B1 into a token string of 18 tokens (13 tag tokens and five <TEXT> tokens) like the following: <B>T</B><BR><FONT>T<A>T</A><BR></FONT><SPAN><B>T</B>T<BR></SPAN>, where each T represents a <TEXT> token. This encoding scheme converts the training page into a better format for pattern discovery and alignment. However, pattern discovery often requires a high level of abstraction, which might not be appropriate for extracting finer information. So, we introduce the drill-down operation by incorporating an encoding hierarchy, such as that for multidimensional models in OLAP (online analytical processing).

The concept hierarchy for OLERA's drill-down operation comprises a set of encoding schemes classified into three layers: *markup-level encoding > text-level encoding > word-level encoding*. The greater-than sign indicates that each encoding is a higher-level abstraction of the encoding to the right. Each level of the encoding hierarchy contains finer classification of several encoding schemes. For example, the markup-level encoding contains both the block-level encoding scheme and text-level (tag) encoding scheme.[4] Word-level encoding schemes focus on the elements of sentences: phrases separated by quotation marks, parentheses, and brackets; words separated by blank spaces; and other symbols such as dollar signs, dashes, slashes, and so forth.

Users can also specify an encoding scheme (or delimiters) as shown in Figure 4a, where the first two strings are encoded into token string "<TEXT>,<TEXT>" and the third is encoded into "<TEXT>,<TEXT>,<TEXT>." In addition to delimiter-based encoding schemes, users can also apply language-specific information such as part-of-speech tagging, semantic-class information, and so forth. For instance, OLERA can parse sentences into proper grammatical notations such as <subject><verb><dobj>. It can also recognize date-related strings, such as "2002/9/1" or "Apr. 4, 2002," as a <DATE> token. Sometimes, you can use a tag's level information in the parse tree in the encoding to avoid misaligning the same tags in different levels.

In the context of semistructured Web IE, we focus on simpler text data segmented from Web pages using delimiter-based encoding schemes. Unless noted otherwise, all the following algorithms operate on the encoded token strings.

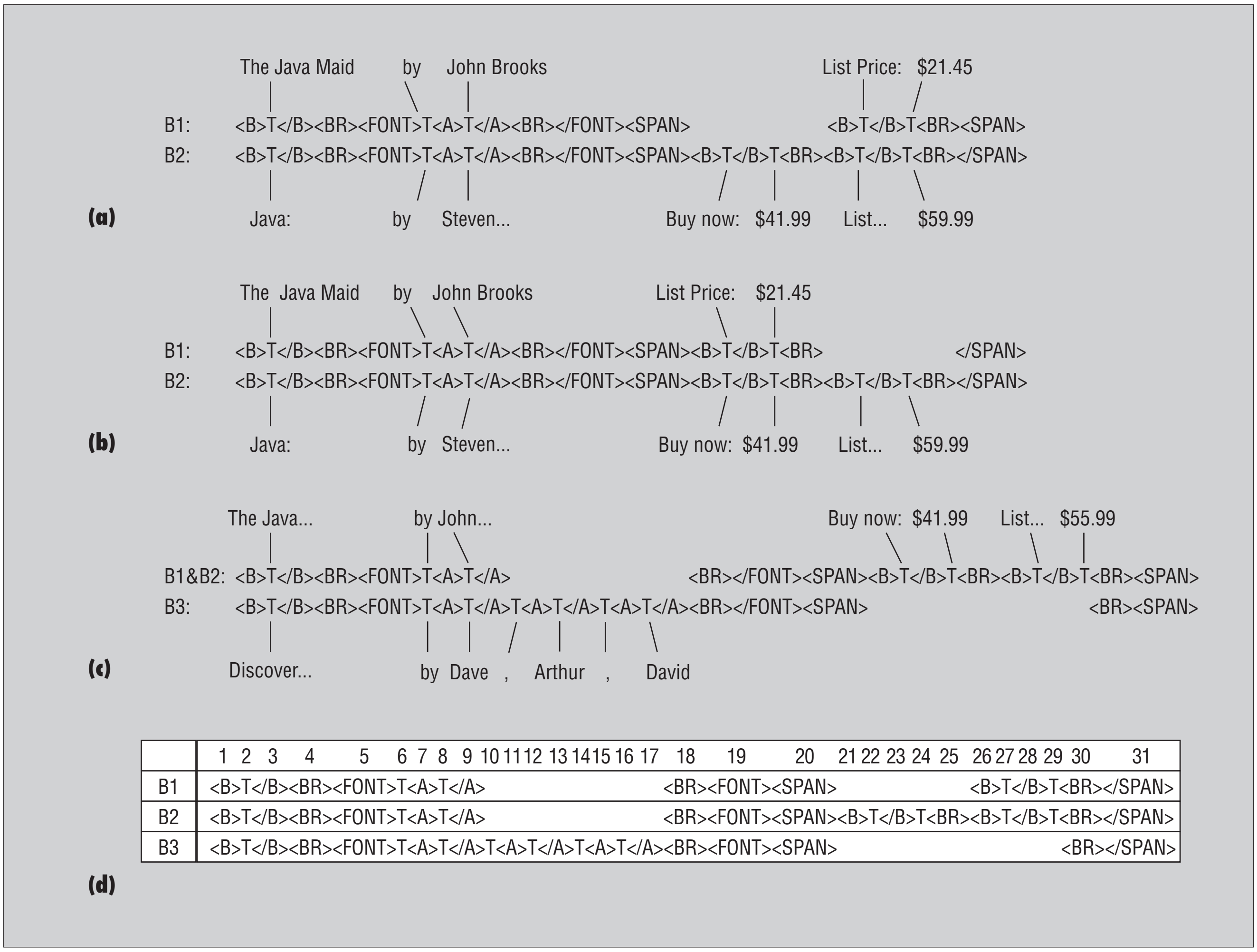
Using approximate matching

Given the encoded token string P of the enclosed example, OLERA will discover other similar records in the training set. Let T denote a training page's encoded token string. We say a substring T' of T is similar to P if their similarity is greater than a given threshold. In this article, we use two string alignments to define the similarity between two strings.

We can align two strings S_1 and S_2 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. We define the value of such an alignment as $\sum_{i=1}^{l} s(S_1'[i], S_2'[i])$, where l denotes the (equal) length of the two strings S_1' and S_2' in the alignment, and $s(x, y)$ denotes the value

B1	The Java Maid by John Brooks List price: $21.45	<b class=sans>The Java Maid</b><br> <font size=–1>by <a href=link>John Brooks</a><br></font> <span class=small><B>List Price:</B>$21.45<BR></span>
B2	Java Black Book by Steven Holzer Buy now: $41.99 List price: $59.99	<b class=sans>Java Black Book</b><br> <font size=–1>by <a href=link>Steven Holzer</a><br></font> <span class=small><B>Buy now:</B>$41.99<BR> <B>List Price:</B>$59.99<br></span>
B2	Discover Visual Cafe by Dave Wall Arthur Griffith, David A. Wall	<b class=sans>Discover Visual Cafe<b><br> <font size=–1>by <a href=link>Dave Wall</a>, <a>Arthur Griffith<a/>, <a href=link>David A. Wall <a/><br></font><span class+small><be></span>

Figure 5. Example HTML sources used in block enclosing.



(a)
The Java Maid by John Brooks List Price: $21.45
B1: <B>T</B><BR><FONT>T<A>T</A><BR></FONT><SPAN> <B>T</B>T<BR><SPAN>
B2: <B>T</B><BR><FONT>T<A>T</A><BR></FONT><SPAN><B>T</B>T<BR><B>T</B>T<BR></SPAN>
Java: by Steven... Buy now: $41.99 List... $59.99
(b)
The Java Maid by John Brooks List Price: $21.45
B1: <B>T</B><BR><FONT>T<A>T</A><BR></FONT><SPAN><B>T</B>T<BR> </SPAN>
B2: <B>T</B><BR><FONT>T<A>T</A><BR></FONT><SPAN><B>T</B>T<BR><B>T</B>T<BR></SPAN>
Java: by Steven... Buy now: $41.99 List... $59.99
(c)
The Java... by John... Buy now: $41.99 List... $55.99
B1&B2: <B>T</B><BR><FONT>T<A>T</A> <BR></FONT><SPAN><B>T</B>T<BR><B>T</B>T<BR><SPAN>
B3: <B>T</B><BR><FONT>T<A>T</A>T<A>T</A>T<A>T</A><BR></FONT><SPAN> <BR><SPAN>
Discover... by Dave , Arthur , David
(d)

	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
B1	<B>T</B><BR><FONT>T<A>T</A> <BR><FONT><SPAN> <B>T</B>T<BR></SPAN>
B2	<B>T</B><BR><FONT>T<A>T</A> <BR><FONT><SPAN><B>T</B>T<BR><B>T</B>T<BR></SPAN>
B3	<B>T</B><BR><FONT>T<A>T</A>T<A>T</A>T<A>T</A><BR><FONT><SPAN> <BR></SPAN>

Figure 6. Aligning the encoded token strings from Figure 5: (a) the optimal alignment of B1 and B2 (with 18 matched tokens and five mismatches with spaces); (b) Another alignment of B1 and B2; (c) the alignment of B3 with the alignment result of B1 and B2; (d) the multiple alignment of B1, B2, and B3.

obtained by aligning two characters x and y. Traditionally, a match ($x = y$) of two characters is assigned a value of s (> 0), a mismatch ($x \neq y$) is assigned a value of s, and an alignment against spaces is assigned a value of d.

We define the similarity score, $sim(S_1, S_2)$, of two strings S_1 and S_2 as the optimal value of all alignments between S_1 and S_2. We can compute the optimal alignment of two strings, S_1 and S_2, using dynamic programming with base conditions

$$V(i, 0) = -i * d; \quad i = 0, 1, \ldots, |S_1|$$
$$V(0, j) = -j * d; \quad j = 0, 1, \ldots, |S_2|$$

and general recurrence

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

where $V(i, j)$ denotes the value of the optimal alignment of prefixes $S_1[1...i]$ and $S_2[1...j]$.

With the definition of two strings' similarity, we can now use approximate matching, a variant of two-string alignment, to discover all similar records of P in T. Given a threshold $\theta\,(0 < \theta < 1)$, we say a substring T' of T is an approximate of P if and only if the similarity ratio of P and T' is greater than θ. In other words, the optimal alignment of T' with P has a value greater than $\theta * s * |P|$, where $s * |P|$ denotes the largest value matching P. We can solve the problem of determining if there's an approximate occurrence of P in T using the same recurrences as for two-string (global) alignment between P and T and change only the base condition for $V(0, j)$ to $V(0, j) = 0$ for all j.

For example, block B2 in Figure 6a is an approximate matching of block B1 within threshold 0.5 because the alignment of the encoded token strings for B1 and B2 shown in Figure 6a has a value of $18 * s - 5$ (18 matches and five missing) greater than $0.5 * s * 19$, for $s = 3$. Similarly, B3 is an approximate occurrence of B1, with 14 matches and 12 mismatches with spaces.

To discover all approximate occurrences of P in T, we first identify the position j' in T such that $V(|P|, j')$ has the largest value among all $V(|P|, j)$, and $V(|P|, j')$ is greater than $\theta * s * |P|$. For this j', we can output the approximate occurrence $T[k', j']$ by backtracking from (m, j') until we reach a cell in row zero $(0, k')$. We then apply this procedure to $T[1, k' - 1]$ and $T[j' + 1, |T|]$ recursively to find all approximate occurrences of P in T.

Using approximate matching, we can discover records similar (with at least θ) to the enclosed example in the training set. If the system doesn't discover all desired records, the user can decrease the threshold θ to discover more records. For some cases, one enclosed example can't approximate all records in the training set with reasonable threshold. For example, a CGI script can have several display templates to enhance visualization for different products. 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 resembles several enclosed examples, the system will choose the most similar one. Similarly, if two enclosed examples match two overlapping text segments, the system considers such text segments to be the same and applies the most similar rule or example.

Using multiple-string alignment

For those records identified by approximate matching, we need a generalization over these instances. Let's say we discover k token strings after approximate matching. We'll apply multiple-string alignment 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 as l. Then, the strings are arrayed in a matrix $\boldsymbol{M}$ with k rows of l columns so that each token and space of each string is in a unique column. For example, Figure 6d shows the alignment of the three blocks arranged in a matrix of three rows and 31 columns.

With multiple-string alignment, we can represent a set of records in a spreadsheet. Because only <TEXT> and some special tag tokens contain the data to be extracted, we can show only the contents encoded as <TEXT> tokens and the hyperlinks embedded in <A> and <IMG> tags for visualization. Additionally, we can represent a set of records in profile representation or signature representation, which we can then use for extraction in the test phase. In this article, we adopt signature representation for extraction rules. For example, we can represent the alignment of Figure 6d in signature representation as <B>T</B><BR><FONT>T<A>T</A>[T][<A>][T][</A>][T][<A>] ..., with brackets denoting optional occurrences.

We can estimate the effectiveness of a multiple alignment using an extension from two-string alignment, the *sum-of-pairs* (SP) objective function. The SP score of a multiple alignment $\boldsymbol{M}$ is the sum of the similarity scores of pair-wise global alignments induced by $\boldsymbol{M}$.[4] We can solve the SP problem using dynamic programming for a small number of strings with $O(n^k)$ time for k strings of length n. A bounded-error approximation algorithm for large numbers of strings is also available. For example, the center-star alignment algorithm was developed as a bounded-error heuristic by iteratively aligning a new string to the growing multiple-string alignment. For example, Figure 6c shows the alignment of B3's token string to the already aligned result for B1 and B2.

Notably, the system must modify the match function for aligning <TEXT> tokens.

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.

Because encoding schemes translate text strings into single <TEXT> tokens, several alignments might have the same optimal similarity score. For example, Figure 6c shows another alignment of B1 and B2 where "List Price:" is aligned against "Buy now:" but with the same 18 matches and five mismatches with spaces as Figure 6a. To distinguish the alignments, we enforce the comparison of the primitive data for <TEXT> tokens. The following equation shows the function *match*(x, y) for three cases, $x \neq y$, $x = y \neq$ <TEXT>, and $x = y =$ <TEXT>:

$$match(x,y) = \begin{cases} s & \text{if} \quad x \neq y \\ -s & \text{if} \quad x = y \neq \ <TEXT>\ , \\ simR(x.prim, y.prim) & \text{if} \quad x = y = \ <TEXT> \end{cases}$$

where *x.prim* denotes the primitive data of a <TEXT> token x and $simR(s_1, s_2)$ is the value of the optimal global alignment of s_1 and s_2 over the longer length of the strings. Therefore, the function *simR* returns a value between 0 and s.

Note that the score function can affect two strings' optimal alignment. For example, if s is greater than 2 * d, we prefer an alignment with spaces to aligning two different characters. OLERA gives s a value greater than 2 * d to prevent an alignment of two different tokens. In other words, we purposely exclude the disjunction of different tokens because it's not convenient for attribute name assignment.

Experiments

We've tested OLERA on a set of 25 real-world Web sites (see Table 1). Sites 1 through 15 are multirecord pages and sites 16–25 are single-record pages. Some of the information sources have been used elsewhere (sites 1–14[5] and 11–16[3]). We collected a total of 2,906 pages for experiments. Table 1 lists the Web sites we tested and their schema, and Table 2 lists our experimental results.

Procedure

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 the system doesn't discover all records in the training set, we can reduce the similarity threshold to approximate more records. (We can adjust the similarity threshold by double-clicking "Enclosing threshold" as shown in the black box in Figure 2b.) If the discovered records are misaligned due to attribute permutation, we'll enclose one misaligned record as another example.

When all records in the training set are correctly extracted, the system applies the extraction rule to other unseen pages for testing. If the system doesn't extract all records in the testing pages, we add another page that contains such records to the training set. (Another way is to add all misextracted pages into the training set, which speeds up the training process. However, to identify the least number of pages needed for one site, we add pages to the training set one page at a time.) We repeat the same procedure for the training pages until the system successfully extracts all testing records.

We repeated this procedure three times and averaged the numbers of training pages for each data source (see Table 2). Table 2 also compares the number of operations for enclosing/drill-down/roll-up operations, the retrieval performance (precision and recall), the threshold set for the training, and the final extraction rule's pattern length.

Table 1. The set of 25 real-world Web sites used to test OLERA.

Web site	Total no. of pages	Records to be extracted	Total no. of attributes	Missing attributes	Permuted attributes	List attributes
1. AltaVista	100	2,000	4	Yes	No	No
2. DirectHit	100	1,000	4	Yes	No	No
3. Excite	100	1,000	4	Yes	No	No
4. HotBot	100	1,000	4	Yes	No	No
5. Infoseek	100	1,500	3	Yes	No	No
6. MSN	100	1,500	3	No	No	No
7. NorthernLight	100	1,000	4	Yes	No	No
8. Sprinks	100	2,000	4	Yes	No	No
9. Webcrawler	100	2,500	3	No	No	No
10. Yahoo	100	2,000	4	Yes	No	No
11. BigBook	235	4,299	6	No	No	No
12. IAF (Internet Address Finder)	200	1,156	6	Yes	Yes	No
13. OKRA	252	3,335	4	No	No	No
14. Quote Server	200	743	18	Yes	No	No
15. LA Weekly	28	159	5	Yes	No	No
16. Zagat's Guide	91	91	4	No	No	Yes
17. A1Books	100	100	10	Yes	No	Yes
18. Amazon	100	100	10	Yes	Yes	Yes
19. Barnes & Noble	100	100	13	Yes	No	Yes
20. BookPool	100	100	12	Yes	No	Yes
21. ByuBook	100	100	6	Yes	No	No
22. Ebay	100	100	11	Yes	No	No
23. iUniverse	100	100	8	No	No	Yes
24. JuilliardBook	100	100	6	Yes	No	No
25. PMIBook	100	100	10	Yes	Yes	No

Results

The first thing we notice in Table 2 is that the number of training pages needed for multirecord pages is comparably less than for single pages. One reason for this is that each multirecord page contains several records where variations can occur. Another reason is that the number of attributes in a record for multirecord pages is comparably smaller than for singular pages, so fewer variations exist. This can also be seen from pattern length—the average length of a record is 21 tokens for multirecord pages and 121 tokens for single pages. Longer patterns often present more changes in the data structure, which is why singular pages usually require more training pages than multirecord pages.

Next, we compare the number of operations and the parameter adjustment during the training process. We found that 0.5 (default) is a good similarity threshold for most data sources because it can approximate most records we want to extract. For cases when 0.5 is too low—the system discovers additional segments—we can use the similarity between each discovered record and the example to increase the threshold. On the other hand, when 0.5 is too high—the system doesn't discover some records—we can use the highest similarity of the segment that's not displayed to increase the threshold. Therefore, threshold adjustment is usually done quickly.

For the number of enclosing operations, usually one suffices for generating extraction rules. We only need multiple enclosing operations when the discovered records can't be aligned properly due to the various order of attributes in one data record (sites 18 and 25) or different formats used for pages (site 19 and 21). Drill-down operations can separate text strings such as "Retail Price: $89.99," which contains no delimiters of the encoding scheme for the enclosing operation. We only apply roll-up operation once (site 24) for concatenating small fragments of the book description, which are generated during the enclosing operation. In other words, the default encoding scheme, text-level tag, is sometimes too coarse and other times too fine for the enclosing operations. In fact, sites 1, 3, 7, and 10 would have required roll-up operations if we'd used a text-level-tag encoding scheme. For a better alignment result, we use a block-level-tag encoding scheme for these four Web sites, so no roll-up operation is used.

Three Web sites require special mention. Although OLERA identified all of IAF (site 12) and LAWeekly's (site 15) records, it couldn't completely recognize some of the attributes through drill-down operations due to incorrect alignment. Also, PMIBook has considerable attribute permutations (43 permutations in total), where most of the 10 attributes can occur in any unrestricted order. In this example, we enclose 10 tag-value pairs and use the multipass architecture for OLERA.

To compare our results with those of related work, we focus on sites 11 through 16 (which have been used in other research[3,5–8]). Table 3 shows the accuracy and the number of examples used for three annotation-based approaches. For annotation-free approaches (IEPAD, RoadRunner, and EXALG), we record the number of attributes that can be correctly identified and the number of attributes considered in a record (see Table 4). This is because the other papers don't report accuracy (or they use a different definition of accuracy) for some of the approaches. Among these Web sites, IAF is the most challenging because most of the six attributes can be missing. Although annotation-free approaches have no a priori knowledge about the attributes to be extracted, OLERA and IEPAD can recognize four of the six attributes, while EXALG can only identify one of them. Comparatively, it's easier to wrap IAF with OLERA, because only one enclosing operation is required. LA Weekly is another Web site that OLERA can't perfectly wrap because one of the attributes (credit card) was misaligned during the drill-down operation.

Table 2. Experimental results from testing 25 real-world Web sites.

Site	No. of training pages	Enclosing/drill down/roll up	Precision	Recall	Threshold	Length
1	2	*1/0/0	100	100	.60	19
2	3	1/0/0	100	100	.45	11
3	1	*1/0/0	100	100	.50	10
4	7	1/0/0	100	100	.45	17
5	4	1/0/0	100	99	.50	27
6	1	1/0/0	100	100	.50	9
7	1	*1/0/0	100	100	.50	34
8	4	1/0/0	100	99	.45	16
9	1	1/0/0	100	100	.75	13
10	1	*1/0/0	100	97	.50	12
11	1	1/2/0	100	100	.50	30
12	4	1/2/0	85.6	85.6	.50	9
13	1	1/0/0	100	100	.50	35
14	3	1/0/0	100	100	.50	53
15	1	1/1/1	85.6	85.6	.50	16
16	5	1/1/0	100	100	.50	44
17	3.3	1/0/0	95.7	100	.50	158
18	12	2/4/0	100	100	.50	168
19	17	3/1/0	100	100	.45	116
20	6	1/2/0	100	100	.50	61
21	4.3	3/0/0	100	100	.70	36
22	10.7	1/0/0	100	100	.50	466
23	1.7	1/2/0	100	100	.50	85
24	6.3	1/3/1	100	100	.50	72
25	1	10/0/0	100	100	.50	4

* Block-level encoding scheme used

Table 3. The accuracy and number of examples used for three annotation-based approaches.

	Site			
	11	12	13	14
WIEN				
Accuracy (%)	100	0	100	0
No. of examples used	274	Infinite	46	Infinite
Stalker				
Accuracy (%)	97	85–100	97	79
No. of examples used	8	10	1	10
Softmealy				
Accuracy (%)	100	42–56	100	85–97
No. of examples used	6	10	1	10

Limitations

OLERA has two limitations. First, it's sensitive to the ordering of input information because it uses a string-alignment technique. Most annotation-free IE systems have this problem. One solution is to incorporate a multipass mechanism such as Stalker or Softmealy and modify the extractor architecture.

Second, extraction failure could occur when the templates for each attribute are similar. For example, the repetitive attribute-value pairs in a record could cause errors in alignment or boundary extraction. To solve this problem, users usually need to specify their own encoding scheme such as by using attribute tags as delimiters or including the DOM tree-level information for HTML tags.

We designed OLERA to fill a gap such that users can select data of interest before the training process and name the extracted data after the system generalizes the extraction rules. We call this approach *semisupervised* because the user gives a rough example—rather than an "exact and perfect" labeling—of a record. Compared to supervised approaches, the enclosing operation is one of the many steps to label a training example. In addition, OLERA further reduces the number of examples by automatically discovering other similar examples for generalizing extraction rules. We propose the enclosing, drilling down, and attributing operations because Web pages are often embedded with data for various purposes, so relevant and irrelevant data are mixed and the

Table 4. Correctly identified attributes and attributes considered in a record for four annotation-free approaches.

	Site					
	11	12	13	14	15	16
IEPAD						
No. of attributes that could be correctly identified	6	4	4	18	Not tested	Not tested
No. of attributes considered	6	6	4	18	Not tested	Not tested
RoadRunner						
No. of attributes that could be correctly identified	6	0	4	Not tested	4	Not tested
No. of attributes considered	6	6	4	Not tested	5	Not tested
EXALG						
No. of attributes that could be correctly identified	5	1	8	16	1	4
No. of attributes considered	5	7	8	16	4	6
OLERA						
No. of attributes that could be correctly identified	6	4	4	18	4	4
No. of attributes considered	6	6	4	18	5	4

desired granularity of data varies from application to application.

Several research directions exist for further study. For supervised or semisupervised IE systems, is it possible to maintain high recall when high precision is pursued using more examples? For example, although some generalization methods, such as Olera's string alignment, achieve high extraction recall given a small number of examples, their recall decreases when we try to improve their precision using more examples. Such a phenomenon exists when the generalization moves to higher levels too quickly. For unsupervised IE systems, we still must ask to what level of granularity a system can differentiate a token's roles (template or data). For example, RoadRunner incorporates a markup-level encoding scheme for input-page tokenization, while Exalg uses a word-level encoding scheme. Does the granularity of tokenization affect the template's induction for a set of input pages? Ambiguity rises in the presence of nullable data attributes,[9] owing partly to incomplete sampling of the data source to be wrapped and partly to the grammar used to generate the pages for the data source. Further research is also necessary to determine when grammars can be induced and what kind of grammars can be induced.

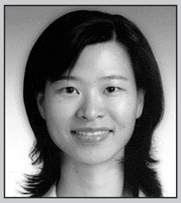
Acknowledgments

The Ministry of Education Taiwan sponsored this work under grant 92-H-FA07-1-4.

References

1. N. Kushmerick, D. Weld, and R. Doorenbos, "Wrapper Induction for Information Extraction," *Proc. 15th Int'l Joint Conf. Artificial Intelligence* (IJCAI 97), Morgan Kaufmann, 1997, pp. 729–737.
2. C.-N. Hsu and M.-T. Dung, "Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web," *Information Systems*, vol. 23, no. 8, 1998, pp. 521–538.
3. I. Muslea, S. Minton, and C. Knoblock, "A Hierarchical Approach to Wrapper Induction," *Proc. 3rd Int'l Conf. Autonomous Agents*, ACM Press, 1999, pp. 190–197.
4. D. Gusfield, "Efficient Methods for Multiple Sequence Alignment with Guaranteed Error Bounds," *Bull. Math. Biology*, vol. 55, 1993, pp. 141–154.
5. C.-H. Chang, C.-N. Hsu, and S.-C. Lui, "Automatic Information Extraction from Semi-Structured Web Pages by Pattern Discovery," *Decision Support Systems J.*, vol. 35, no. 1, 2003, pp. 129–147.
6. V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: Towards Automatic Data Extraction from Large Web Sites," *Proc. 27th Int'l Conf. Very Large Data Bases* (VLDB 01), Morgan Kaufmann, 2001, pp. 109–118.
7. A. Arasu and H. Garcia-Molina, "Extracting Structured Data from Web Pages," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 03), ACM Press, 2003, pp. 337–348.
8. C.-N. Hsu and C.-C. Chang, "Finite-State Transducers for Semi-Structured Text Mining," *Proc. 16th Int'l Joint Conf. Artificial Intelligence* (IJCAI 99), Morgan Kaufmann, 1999, pp. 38–49.
9. G. Yang, I.V. Ramakrishnan, and M. Kifer, "On the Complexity of Schema Inference from Web Pages in the Presence of Nullable Data Attributes," *Proc. 12th Int'l Conf. Information and Knowledge Management* (CIKM 03), ACM Press, 2003, pp. 224–231.

The Authors



Chia-Hui Chang is an associate professor in the Department of Computer Science and Information Engineering at the National Central University in Taiwan. Her research interests include information extraction, data mining, machine learning, and Web-related research. She received her PhD in computer science and information engineering from National Taiwan University. Contact her at chia@csie.ncu.edu.tw; www.csie.ncu.edu.tw/~chia.

Shih-Chien Kuo is an engineer in the R&D Department at Trend Micro in Taiwan. His research interests include Web information extraction and data mining. He received his MS in computer science and information engineering from National Central University, Taiwan. Contact him at bruce@db.csie.ncu.edu.tw.

PURPOSE The IEEE Computer Society is the world's largest association of computing professionals, and is the leading provider of technical information in the field.

MEMBERSHIP Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

COMPUTER SOCIETY WEB SITE
The IEEE Computer Society's Web site, at **www.computer.org**, offers information and samples from the society's publications and conferences, as well as a broad range of information about technical committees, standards, student activities, and more.

OMBUDSMAN Members experiencing problems—magazine delivery, membership status, or unresolved complaints—may write to the ombudsman at the Publications Office or send an e-mail to help@computer.org.

CHAPTERS Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

AVAILABLE INFORMATION

To obtain more information on any of the following, contact the Publications Office:

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

To check membership status or report a change of address, call the IEEE toll-free number, +1 800 678 4333. Direct all other Computer Society-related questions to the Publications Office.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, industry trends, and product reviews.

Periodicals. The society publishes 12 magazines and 10 research transactions. Refer to membership application or request information as noted at left.

Conference Proceedings, Tutorial Texts, Standards Documents.
The Computer Society Press publishes more than 160 titles every year.

Standards Working Groups. More than 200 groups produce IEEE standards used throughout the industrial world.

Technical Committees. Thirty TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

COMPUTER SOCIETY OFFICES

Headquarters Office
1730 Massachusetts Ave. NW
Washington, DC 20036-1992
Phone: +1 202 371 0101 • Fax: +1 202 728 9614
E-mail: hq.ofc@computer.org

Publications Office
10662 Los Vaqueros Cir., PO Box 3014
Los Alamitos, CA 90720-1314
Phone:+1 714 821 8380
E-mail: help@computer.org
Membership and Publication Orders:
Phone: +1 800 272 6657 Fax: +1 714 821 4641
E-mail: help@computer.org

Asia/Pacific Office
Watanabe Building
1-4-2 Minami-Aoyama,Minato-ku,
Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • Fax: +81 3 3408 3553
E-mail: tokyo.ofc@computer.org