

# Text Mining From Site Invariant and Dependent Features For Information Extraction Knowledge Adaptation\*

Tak-Lam Wong and Wai Lam  
Department of Systems Engineering And Engineering Management  
The Chinese University of Hong Kong  
Hong Kong  
{wongtl,wlam}@se.cuhk.edu.hk

**Keywords:** Text Mining, Information Extraction, Knowledge Adaptation

## Abstract

We develop a framework which can adapt previously learned information extraction knowledge from the source Web site to new unseen sites. Our framework also makes use of data items previously extracted or collected. Site invariant features are derived from the previously learned extraction knowledge and previously collected items. Multiple text mining methods are employed to automatically discover machine labeled training examples for the new site. Both site invariant and site dependent features of these machine labeled training examples are used to learn the new extraction knowledge. Extensive experiments on real-world Web sites have been conducted to demonstrate the effectiveness of our framework.

## 1 Introduction

The World Wide Web has been growing in a rapid pace. Huge amount of online documents are available on the Web. Many intelligent applications such as shopping agents and automated travel assistants rely on automatically extracting data from Web documents [7]. Some users may wish to obtain items of interest by browsing online documents page by page manually. It is desirable to develop a system which can automatically extract information from different Web pages effectively. Techniques have been proposed for extracting information from different kinds of textual documents. Some methods focus on extracting information from free texts which are largely grammatical [6, 22]. Usually, natural language processing (NLP) techniques

are employed in these systems. The other kind of textual documents is highly structured texts which are organized in a rigid format. Uniform syntactic rules are usually employed to extract information from these structured documents [3]. Unlike structured documents and free texts, there is a kind of documents called semi-structured documents. Semi-structured documents are characterized by the fact that they contain a mix of short ungrammatical (or weakly grammatical) text fragments, mark-up tags, and free texts. Web pages in HTML or XML documents belong to semi-structured documents. NLP techniques and uniform syntactic rules are not suitable for extracting information from semi-structured documents. A promising approach is to employ *wrapper* containing information extraction knowledge or pattern which can automatically extract data from Web documents. The extraction knowledge or pattern can identify the appropriate text fragments in documents. Some systems rely on human effort to construct the wrapper [12]. However, it is undesirable for writing extraction rules manually because such task is very time-consuming, error-prone, and requiring high level of expertise. Recently, many wrapper induction systems have been designed to automatically construct wrappers from training examples by using machine learning techniques [4, 21].

Figure 1 shows a sample of a Web page containing information about a book catalog collected from a Web site<sup>1</sup> denoted by  $S$ . Figure 2 depicts an excerpt of the HTML document corresponding to the Web page of Figure 1. The items of interest in each book record are *book title*, *author*, and *price*. To construct the wrapper,  $W_S$ , using a wrapper induction system, users are only required to provide, via GUI, few samples of text fragments as shown in Table 1. The wrapper induction system can then learn the wrapper automatically. The learned wrapper,  $W_S$ , is tailored to the site  $S$ . It

\*The work described in this paper was substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Nos: CUHK 4187/01E and CUHK 4179/03E) and CUHK Strategic Grant (No: 4410001).

<sup>1</sup>The URL of the Web site is [www.halfpricecomputerbooks.com](http://www.halfpricecomputerbooks.com).



new site. Both *site invariant* and *site dependent features* of the machine labeled training examples will then be considered in learning the new wrapper for the unseen target site.

We have previously developed a method called WrapMA [24] for solving the wrapper induction and wrapper adaptation. However, WrapMA still requires some human efforts to scrutinize the intermediate data during the adaptation process. The major advantage of IEKA over WrapMA is that IEKA tackles the wrapper adaptation problem in a completely automatic manner.

## 2 Related Work

Several wrapper learning systems [4, 15, 21] have been proposed. They make use of machine learning techniques to discover the wrappers from the user labeled examples. Despite the good performance of these wrapper learning methods, they share some common shortcomings. As the layout format of Web sites changes from time to time, a previously constructed wrapper may become obsolete sooner or later. Wrapper maintenance aims at re-learning a new wrapper when the current wrapper can no longer extract correct information. RAPTURE [13] uses regression testing on the data extracted by the wrapper to verify the validity of the wrapper. WebCQ [19] is designed to monitor the changes of the Web documents. Both of them can only partially solve the wrapper maintenance problem. Lerman et al. [16, 17] tried to tackle the wrapper maintenance problem by their DataPro algorithm. When the layout format of the Web site is believed to be changed, it tries to label a new set of training examples for training a new wrapper by matching the patterns learned in the old set of training examples. For example, the pattern learned for the business name “Cajun Kitchen” is  $\langle ALPHA\ UPPER \rangle$ . They assume that the items in the changed Web documents have the same pattern. This requirement poses a limitation to their approach.

When a wrapper is found to be obsolete, a new wrapper might be re-learned using previously collected training examples. Most likely, these training examples may also become invalid. Besides, the wrappers learned from a particular information source typically cannot be applied to extract items from other sources. A separate effort is required to provide a new set of training examples in the new Web site, so as to learn a new wrapper for that source. A possible solution for this problem is to address the issue of preparing the training examples. Muslea et al. [20] proposed an active learning technique which can partially reduce the human effort in preparing training examples. Brin’s DIPRE [1] tackled this problem by continuously providing some concept pairs (e.g., book title/author) to the system. DIPRE searches the

documents that contain the concept pairs and learn the extraction pattern. The extraction pattern will then be applied to other documents to find more training examples. Bootstrapping algorithms [9] aim at reducing the number of training examples. They initiate their training with a set of *seed words* and assume that the seed words will be present in the training data. These systems still require a separate effort for different Web sites. ROADRUNNER [5] attempts to solve the problem by eliminating the need for training example preparation. The idea is based on the difference and the similarity of the text content of the Web pages. DeLa [23] is a system developed for generating wrapper without using training examples. The idea of DeLa is to find repeated patterns in the Web page and discover a regular expression for the repeated patterns. However, human effort is still required in order to get the semantic meaning and the relationship between the extracted data.

The above approaches cannot effectively solve the wrapper adaptation problem. Wrapper adaptation aims at adapting a previously learned wrapper in an information source to a new, unseen information target in the same domain. In principle, wrapper adaptation can also solve the wrapper maintenance problem. Golgher et al. [10] tried to address the wrapper adaptation problem by a query-like approach. This approach searches the exact matching of items in an unseen Web page. However, exact match of items in different Web sites is ineffective.

## 3 Our Proposed Framework: IEKA

### 3.1 Motivation and Design of IEKA

Web pages in a Web site can be characterized by two types of feature. The first type of feature is *site dependent feature*. Generally, site dependent features remain largely unchanged for Web pages under the same site, but they are different for pages originated from different sites. For example, the layout format of Web pages can be represented by site dependent features. The values of these features are likely to be dependent on the Web site only. The second type of feature is *site invariant feature*. This type of feature is used to characterize the item content (e.g., *book title*) in a particular domain (e.g., *book domain*). These features likely remain unchanged in different Web sites and are only dependent on the item content itself. We observe that the text fragments regarding the layout format of the Web pages from the same Web site are similar, while the text fragments regarding the layout format of the Web pages from different Web sites are different. Hence, we model the text fragments regarding the layout format of the Web pages as site dependent feature. We make use of the site invariant features to characterize the item content

since the item content is likely to be similar in different Web sites of the same domain.

Our proposed adaptation framework is called IEKA (Information Extraction Knowledge Adaptation). IEKA is able to make use of the previously learned information extraction knowledge from the source Web site to learn new extraction knowledge in the new unseen Web site. The rationale of IEKA is to exploit the site invariant features derived from two pieces of information from the source site. The first piece of information is the extraction knowledge contained in the previously learned wrapper. The other piece of information is the items previously extracted or collected in the source site.

The previously learned wrapper contains extraction knowledge for the source Web site. This extraction knowledge is learned from the training examples which contain the site invariant features and the site dependent features of the source Web site. To perform information extraction for a new unseen Web site, the existing extraction knowledge is useful since the site invariant features are likely applicable. However, the site dependent features cannot be used since they are different in the new site. We call such knowledge as *weak* extraction knowledge.

The items previously extracted or collected in the source Web site embody rich information about the item content. For example, these extracted items contain some characteristics and orthographical information about the item content. In fact, these items can be viewed as training examples for the new unseen site. These training examples are different from the ordinary training examples because the former only contain information about the site invariant features, while the latter contain information about both the site invariant features and site dependent features. We call this property *partially specified*.

IEKA is designed based on the properties of site invariant features and site dependent features. Figure 5 depicts the overview of the design of IEKA. Based on analyzing the site invariant features in the weak extraction knowledge and the partially specified training examples, IEKA employs multiple text mining methods to automatically discover some training examples for the new unseen Web site. These newly discovered training examples are called *machine labeled training examples*. The next step is to analyze both the site invariant features and site dependent features of those machine labeled training examples of the new site. IEKA then learns the new information extraction knowledge tailored to the new site using a wrapper learning component.

IEKA consists of three stages employing multiple text mining methods to tackle the adaptation problem. The first stage of IEKA is the potential training text

fragment identification. At this stage, the weak extraction knowledge contained in the wrapper of the source site is utilized to identify appropriate text fragments as the potential training text fragments for the new unseen site. This stage considers the site dependent features of the Web pages as discussed above. Some *auxiliary example pages* are automatically generated for the analysis of the site dependent features. A modified nearest neighbour classification model is developed for effectively locating the potential training text fragments.

The second stage is the machine labeled training example discovery. It aims at scoring the potential training text fragments. The “good” potential training text fragments will become the machine labeled training examples for learning the new wrapper for the new site. This stage considers the site invariant features of the partially specified training examples. An automatic text fragment classification model is developed to score the potential training text fragments. The classification model consists of two components. The first component is the content classification component. It considers several features to characterize the item content. The second component is the approximate matching component which analyzes the orthographical information of the potential training text fragments.

Based on the automatically generated machine labeled training examples, a new wrapper for the new Web site is learned using the wrapper learning component. The machine labeled training examples may contain inaccurate training examples (or noise). The wrapper learning component in IEKA, derived from our previous work, can cope with this problem which is different from typical wrapper induction.

**3.2 Wrapper Learning Component** In IEKA, there is a wrapper learning component which generates information extraction knowledge from sample text fragments. We employ HISER [18] which is derived from our previous work. In this paper, we will only briefly present the overview of HISER.

HISER is a two stage learning algorithm. At the first stage, it induces a hierarchical representation for the structure of the records. This hierarchical record structure is a tree-like structure which can model the relationship between the items of the records. It can model records with missing items, multi-valued items, and items arranged in unrestricted order. For example, Figure 6 depicts a sample of hierarchical record structure representing the records in the Web site as shown in Figure 1. The record structure in this example contains a *book title*, a list of *authors*, and a *price*. The price consists of a *list price* and a *final price*. There is no restriction on the order of the nodes

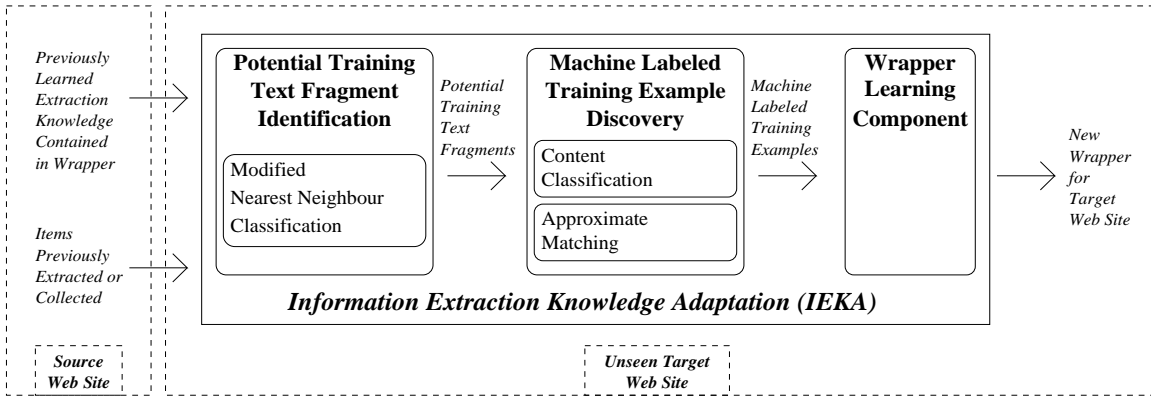


Figure 5: The overview of IEKA.

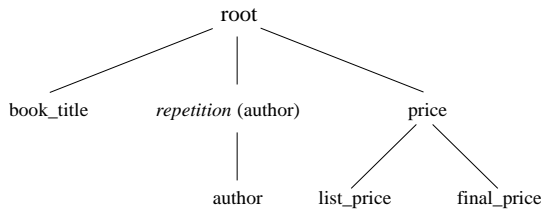


Figure 6: The hierarchical record structure for the book information shown in Figure 1.

**Left pattern component**  
`Scan_Until("Our", TOKEN),`  
`Scan_Until("Price", TOKEN),`  
`Scan_Until(":", TOKEN),`  
`Scan_Until("<HTML_IMG_TAG>", SEMANTIC).`  
**Target pattern component**  
`Contain(<FLOAT>)`  
**Right pattern component**  
`Scan_Until("&nbsp;", TOKEN),`  
`Scan_Until("&nbsp;", TOKEN),`  
`Scan_Until("</b>", TOKEN),`  
`Scan_Until("<HTML_FONT_TAG>", SEMANTIC).`

Table 2: A sample of extraction rule for the final price for the Web document shown in Figure 1.

under the same parent. A record can also have any item missing. The multiple occurrence property of *author* is modeled by a special internal node called *repetition*.

Each node in the hierarchical record structure is associated with a set of extraction rules. These extraction rules are automatically learned at the second stage in HISER. An extraction rule consists of three parts: the left pattern component, the right pattern component, and the target pattern component. Table 2 depicts one of the extraction rules for the *final price* for the Web document in Figure 1. Both of the left and right pattern components make use of a token scanning instruction, *Scan\_Until()*, to identify the left and right delimiters of the item. The token scanning instruction instructs the wrapper to scan and consume any token until a particular token matching is found. The argument of the

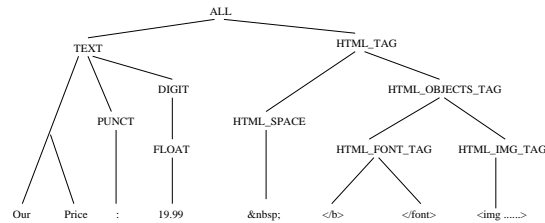


Figure 7: Examples of semantic classes organized in a hierarchy. instruction can be a token or a semantic class. For the target pattern component, it makes use of an instruction, *Contain()*, to represent the semantic class of the item content.

An extraction rule learning algorithm is developed based on a covering-based learning algorithm. HISER first tokenizes the Web document into sequence of tokens. A token can be a word, number, punctuation, date, HTML tag, some specific ASCII characters such as “&nbsp;,” which represents a space in HTML documents, or some domain specific contents such as manufacture names. Each token will be associated with a set of semantic classes, which is organized in a hierarchy. For example, Figure 7 depicts the semantic class hierarchy for the following text fragments from Figure 2 after tokenization.

Our Price: 19.99  
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</b></font>

HISER learns the extraction rules by performing lexical and semantic generalization, until the best extraction rules are discovered. The details of HISER can be found in our previous work [18].

#### 4 Potential Training Text Fragment Identification

From the new unseen site, a Web page containing target data items is collected. This page can be easily obtained

automatically (e.g., by using the search engine provided in the new site). In the first stage of IEKA, the previously learned extraction knowledge of the source site is utilized to identify potential training examples for the new site.

**4.1 Useful Text Fragments** A Web page can be regarded as a sequence of text tokens. A token can be a word, number, punctuation, date, HTML tag, specific ASCII character, or some domain specific contents such as manufacture names.

DEFINITION 4.1. We define a *segment* to be a sequence of continuous tokens in a Web page not containing any HTML tag, and is immediately before and after tokens belonging to the semantic class *delimiter*. The delimiter semantic class contains user defined tokens such as HTML tag, punctuation, specific ASCII characters, or some domain specific contents.

DEFINITION 4.2. Given an extraction rule for a particular item, we define  $\Sigma$  to be the set of the semantic classes contained in the target pattern component of the extraction rule.

DEFINITION 4.3. Given a particular item, we define a *seed* to be a segment of a Web page which contains at least one token belonging to one of the semantic classes in  $\Sigma$ .

IEKA first extracts some seeds as defined above. After obtaining the seeds, some text fragments by extending the seeds forward and backward are generated. A parameter  $T$  is used to control the window size in terms of the number of tokens.

DEFINITION 4.4. For a seed  $s$ ,  $Pre-Seed(s)$  is defined as the set of the positions of the  $i$ -th token immediately before  $s$ , where  $i = 1, \dots, T$ .

DEFINITION 4.5. For a seed  $s$ ,  $After-Seed(s)$  is defined as the set of the positions of the  $i$ -th token immediately after  $s$ , where  $i = 1, \dots, T$ .

DEFINITION 4.6. For a seed  $s$ ,  $Extended-Seed(s)$  is defined as the set of text fragments whose starting position and ending position are indicated by the cross product of  $Pre-Seed(s)$  and  $After-Seed(s)$ .

For instance, consider a seed  $c_0$  in a Web page and suppose  $T$  is set to 3. We get  $\{Pre-Seed(c_0)\} = \{s_1, s_2, s_3\}$  where  $s_i$  is the position of the  $i$ -th token immediately before  $c_0$ , and  $\{After-Seed(c_0)\} = \{e_1, e_2, e_3\}$  where  $e_j$  is the position of the  $j$ -th token immediately after  $c_0$ . Then  $\{Extended-Seed(c_0)\}$  becomes the set of text fragments whose starting position and ending position is indicated by the set as follows:

$$\{(s_1, e_1), (s_1, e_2), (s_1, e_3), (s_2, e_1), (s_2, e_2), (s_2, e_3), (s_3, e_1), (s_3, e_2), (s_3, e_3)\}.$$

DEFINITION 4.7. For a Web page  $P$ , we define the useful text fragment,  $UTF(P)$ , as:

$$UTF(P) = \cup_s \{Extended-Seed(s)\}$$

where  $s$  are all the seeds found in  $P$ .



Figure 8: A sample of Web page about networking books.

```
...
<B> The Terror Network </B> </A> &nbsp;&nbsp;&nbsp; Claire
Sterling <BR> <FONT FACE="Verdana, Geneva, Arial"
SIZE="2"> &raquo;&nbsp;&nbsp;&nbsp; <a
href="http://half.ebay.com/cat/buy/prod.cgi?
cpid=3051974&domainid=1856&meta_id=1"> Paperback,
1985 </a> &nbsp;&nbsp;&nbsp; - Buy it for <a
href="http://half.ebay.com/cat/buy/prod.cgi?
cpid=3051974&domainid=1856&meta_id=1"> <font
color="#CC0000"> $0.75 </font> </a>
...
```

Figure 9: An excerpt of the HTML texts for the Web page shown in Figure 8

**4.2 Auxiliary Example Pages** IEKA will automatically generate some machine labeled training examples in one of the Web pages in the new unseen Web site. We call the Web page where the machine labeled training examples are to be automatically produced as *main example page*  $M$ . Relative to a main example page, *auxiliary example pages*  $A(M)$  are Web pages from the same Web site, but containing different item contents. As the main example page and the auxiliary example pages contain different item contents, the text fragments regarding the field items are different in different Web pages, while the text fragment regarding the layout format are similar. This observation gives a good indication for locating the potential training text fragments.

Auxiliary example pages can be automatically obtained from different pages easily in a Web site. One typical method is to supply different keywords or queries automatically to the search engine provided by the Web site. For instance, consider the book catalog associated with the Web page shown in Figure 3. This Web page is generated by supplying automatically the keyword "PROGRAM" to the search engine provided by the Web site. Suppose a different keyword such as "NETWORK" is automatically supplied to the search engine, a new Web page as shown in Figure 8 is returned. Only a few keywords are needed for a domain and they can be easily chosen in advance. The Web page in Figure 8 can be regarded as an auxiliary example page relative to the Web page in Figure 3. Figures 4 and 9 show the excerpt of the HTML text document associated with the Web page shown in Figures 3 and 8 respectively. The bolded text fragments are related to the items of interest, while the remaining text fragments are related to the format layout. The text fragments related to items are very different in different Web pages, whereas the text fragments related to the format layout are very sim-

ilar. Such text content of the auxiliary example pages provides very useful clues for seeking appropriate text fragments related to the item.

**4.3 Modified Nearest Neighbour Classification Model** Recall that the target pattern component of the previously discovered wrapper from the source Web site contains the semantic classes of the items. From the main example page  $M$  of the new unseen Web site, we can obtain the set  $UTF(M)$ . From an auxiliary example page  $A(M)$ , we can also obtain the set  $UTF(A(M))$ . As mentioned in the previous subsection, the text fragments regarding the items in the main example page are less likely to appear in the auxiliary example page, while the text fragments regarding the layout format will probably appear in both of the main example page and the auxiliary example page. Hence, all the elements in  $UTF(A(M))$  are treated as negative instances relative to the text fragment regarding the items in the main example page  $M$ . Each instance in the modified neighbour classification model is represented by a set,  $t_i$ , containing the unique words in the text fragment.

DEFINITION 4.8. Suppose we have two text fragments  $t_1$  and  $t_2$ . We define the similarity between these two text fragments  $sim(t_1, t_2)$ , as follows:

$$sim(t_1, t_2) = \frac{|t_1 \cap t_2|}{\max(|t_1|, |t_2|)} \quad (4.1)$$

The goal of our modified classification model is to classify the potential training text fragments from  $UTF(M)$ . To achieve this task, for each element in  $UTF(M)$ , we first find its nearest neighbour in  $UTF(A(M))$  based on our defined similarity measure. If the similarity between the element in  $UTF(M)$  and its nearest neighbour in  $UTF(A(M))$  exceeds a threshold,  $\theta$ , it will be classified as negative instance. On the other hand, if the similarity is below  $\theta$ , it will be classified as a potential training text fragment.

Once the potential training text fragments for an item are identified, they will be processed by the text fragment classification model in the machine labeled training example discovery stage. Those “good” text fragments become the machine labeled training examples for the unseen site.

## 5 Machine Labeled Training Example Discovery

As mentioned in Section 3, the partially specified training examples refer to the items previously extracted or collected in the source Web site. The rationale of using the partially specified training examples is that the item content can be represented by the site invariant features which remain unchanged in different Web sites of the

same domain. The partially specified training examples are used to train a text fragment classification model which can classify the “good” text fragments from the potential training text fragments. This text fragment classification model consists of two components which consider two different aspects of the item content. One aspect is the characteristics of the item content. For example, in the consumer electronics domain, a model number of DVD player usually contains tokens mixed with alphabets and digits and starts with capital letter. The first component, called content classification component, considers several features which can effectively characterize the item content. The second aspect is the orthographical information of the item content. For example, the model numbers of the products in the same brand may be only different in few characters. The second component, called approximate matching component, is developed to make use of the orthographical information of the item content to help classify the machine labeled training examples.

**5.1 Content Classification Component** We identify some features for characterizing the content of the items. A classification model can then be learned to classify the “good” potential training text fragments. The features used are as follows:

- $F_1$ : the number of characters in the content
- $F_2$ : the number of tokens in the content
- $F_3$ : the average number of characters per token
- $F_4$ : the proportion of the number of digit number to the number of tokens
- $F_5$ : the proportion of the number of floating point number to the number of tokens
- $F_6$ : the proportion of the number of alphabet to the number of characters
- $F_7$ : the proportion of the number of upper case characters to the number of characters
- $F_8$ : the proportion of the number of lower case characters to the number of characters
- $F_9$ : the proportion of the number of punctuation to the number of characters
- $F_{10}$ : the proportion of the number of HTML tags to the number of tokens
- $F_{11}$ : the proportion of the number of tokens starting with capital letter to the number of tokens
- $F_{12}$ : whether the content starts with a capital letter

These features attempt to characterize the format of the items. Some of the features are also used in [13]. With the above feature design, a classification model can be learned from a set of training examples. The content classification model will return a score,  $f_1$ , which indicates the degree of confidence being “good” potential training text fragment.  $f_1$  will be normalized to a value between 0 and 1.

The content classification model is trained from a

set of training examples composed of a set of *positive item content examples* and *negative item content examples*. The set of positive item content examples are the partially specified training examples in the source site. In the main page of the source Web site,  $M_s$ , we obtain the  $UTF(M_s)$  by replacing  $\Sigma$  (defined in Definition 4.3) with the semantic classes in the target pattern component of the extraction rules for the items. Those elements in  $UTF(M_s)$  which are not in the set of positive item content examples are collected to become the negative item content examples. Next, the values of the features  $F_i$  ( $1 \leq i \leq 12$ ) of each positive and negative item content examples are computed. To learn the content classification model, we employ Support Vector Machines.

**5.2 Approximate Matching Component** French et al. [8] discussed the effectiveness of approximate word matching in information retrieval. To enhance the robustness, we make use of edit distance [11] and design an two-level approximate matching algorithm to compare the similarity between two strings. At the lower level, we compute the character-level edit distance of a given pair of tokens. At the upper level, we compute the token-level edit distance of a given pair of text fragments. We will illustrate our algorithm by an example.

Suppose we obtain a potential training text fragment of model number “*PANASONIC DVDCV52*” and a particular previously collected item content “*PAN DVDRV32K*”. (Actually these two model numbers are obtained from two different Web sites in our consumer electronics domain experiment. They refer to the same brand of products, but having different model numbers.) At the lower level, we compute the character-level edit distance between two tokens with the cost of insertion, deletion, and modification of a character all equal to one. Then the character-level edit distances computed are normalized by the longest length of the tokens. For example, the normalized character-level edit distance between “*PAN*” and “*PANASONIC*” is 0.667.

At the upper level, we compute the token-level edit distance between a potential training text fragment and a partially specified training example, with the cost of insertion and deletion of a token equal to one, and the cost of modification of a token equal to the character-level edit distance between the tokens. The token-level edit distance obtained is then normalized by the largest number of tokens among the potential training text fragment and the partially specified training example. For instance, the normalized token-level edit distance between “*PANASONIC DVDCV52*” and “*PAN DVDRV32K*” is 0.521.

Both of the character-level and token-level edit distance can be computed efficiently by dynamic programming. The score,  $f_2$ , of a potential training text fragment is then computed as follows:

$$f_2 = \max_i \{D'(c, l^i)\} \quad (5.2)$$

where  $D'(c, l^i) = 1 - D(c, l^i) / \max\{m, n\}$  and  $D(c, l^i)$  is the normalized token level edit distance between the the potential training text fragment,  $c$ , and the  $i$ -th partially specified training example.

## 6 New Wrapper Learning for the Unseen Web Site

In the machine labeled training example discovery stage, the scores from content classification component and approximate matching component are computed. The final score  $Score(c)$  of each potential training text fragment  $c$  is given by:

$$Score(c) = wf_1 + (1 - w)f_2 \quad (6.3)$$

where  $f_1$  and  $f_2$  are the scores obtained in content classification component and approximate matching component respectively;  $w$  is a parameter controlling the relative weight of the content classification and approximate matching components and  $0 < w < 1$ .

After the scores of the potential training text fragment are computed, IEKA will select “good” potential training text fragments as machine labeled training examples for the unseen site. The  $N$ -best potential training text fragments will be selected as the machine labeled training examples.

The machine labeled training examples obtained are not grouped in records. We adopt the discovery of repeated pattern approach [2] to discover the record boundary and group the machine labeled training examples into records. This method can automatically identify the repeated pattern in a Web page, by making use of PAT trees. The repeated pattern will be considered to determine if it contains useful information. The machine labeled training examples within two repeated patterns are then grouped to the same record. The records will become the training examples for learning the new wrapper for the new unseen Web site. Users could optionally scrutinize the discovered training examples to improve the quality of the training examples. However, in our experiments, we did not conduct manual intervention and the adaptation was conducted in a fully automatic way.

After obtaining the set of machine labeled training examples, IEKA makes use of the wrapper learning component HISER, derived from our previous work [18], to learn the wrapper tailored to the new unseen Web site. A small refinement on HISER is performed to suit the new requirement. The set of machine labeled training examples is different from the set of user labeled training examples because the former may

	Item	Item value	Score
Example 1	<b>Book Title:</b>	Programming with C++: Program Design Including Data Structures	0.63
	<b>Final Price:</b>	45.00	1.00
Example 2	<b>Author:</b>	Steve Heller	0.60
	<b>Final Price:</b>	4.99	1.00

Table 3: Samples of a machine labeled training example obtained by adapting the wrapper from the Web site shown in Figure 1 to the Web site shown in Figure 3 with our IEKA framework.

contain inaccurate training examples. The noise in the training example set tends to exhibit over-generalization of the extraction rules due to the scoring criteria of the extraction rule learning algorithm in [18]. To reduce this effect, we introduce a meta rule for restricting the number of token generalization of the extraction rule induction algorithm. Each extraction rule may then cover less training examples due to the restriction on the generalization power. However, this will not degrade the extraction performance of the learned wrapper as each extraction rule set in the wrapper may contain more extraction rules to broaden its coverage. This meta rule can avoid the over-generalization effect. The newly learned wrapper is tailored to the new unseen Web site and it can be applied to the remaining pages in the unseen site for information extraction.

## 7 Case Study

Consider the Web site shown in Figure 1. In order to learn the wrapper for this Web site, the user needs to collect some training examples, similar to the one shown in Table 1, via a graphical user interface. HISER learns a wrapper from these training examples. A hierarchical record structure and a set of extraction rules are discovered. Figure 6 shows the hierarchical record structure discovered by HISER and Table 2 shows the extraction rule for the final price node. The learned wrapper is then applied to extract items from other Web pages within the same Web site. The extraction performance is almost perfect.

The learned wrapper for the Web site shown in Figure 1 can extract records from the Web pages in the same site. However, if the learned wrapper is directly used to extract records from the Web page shown in Figure 3, it cannot extract any record. To demonstrate the adaptation capability, we applied our IEKA framework to tackle this problem. After the machine labeled training example discovery stage in IEKA, some machine labeled training examples, such as the two samples shown in Table 3, were automatically obtained. The last column of Table 3 shows the score of the item obtained in our IEKA framework. Example 1 has a book title with score 0.63 and a final price with score 1.0. Example 2 has an author with score 0.60 and a final price with

	Web site (URL)	pp. #	rec. #
T1	1Bookstreet.com (http://www.1bookstreet.com)	5	125
T2	DigitalGuru Technical Bookshops (http://www.digitalguru.com)	17	102
T3	Jim's Computer Books (http://www.vstore.com/cgi-bin/ pagegen/vstorecomputers/jimsbooks/)	7	139
T4	1 Stop Camera & Electronics (http://www.1stopcamera.com)	16	152
T5	AAA Price Electronics (http://www.aaaprice.com)	11	174
T6	Best Buy Digital (http://www.bestbuydigital.com)	11	174
S1	Amazon.com (http://www.amazon.com)	5	125
S2	Barnes & Noble.com (http://www.barnesandnoble.com)	5	120
S3	BookCloseouts.com (http://www.bookcloseouts.com)	3	112
S4	Powell's Books (http://www.powells.com)	5	100
S5	WordsWorth Books (http://www.wordsworth.com)	10	125
S6	bookpool.com (http://www.bookpool.com)	5	124
S7	half.com (http://half.ebay.com)	6	120
S8	Half Price Computer Books (http://www.halfpricecomputerbooks.com)	5	100
S9	Discount-PCBooks.com (http://www.discount-pcbooks.com)	14	110
S10	mmistore.com (http://www.mmistore.com)	11	110
S11	American eSuperstore.com (http://store.yahoo.com/americanesuperstore)	10	100
S12	220Appliances.com (http://www.220appliances.com)	8	113
S13	Circuit City (http://www.circuitcity.com)	6	120
S14	Etronics.com (http://www.etrionics.com)	12	107
S15	DVD Overseas Electronics (http://www.dvdoverseas.com)	13	110
S16	Cambridge SoundWorks (http://www.hifi.com)	12	157
S17	BestBuy.com (http://www.bestbuy.com)	4	123

Table 4: Information sources for experiments (“pp. #” and “rec. #” refer to the number of pages and the number of records collected in the Web site respectively.).

score 1.0. Users could optionally scrutinize the machine labeled training examples to improve the quality of the training examples. In this case study, we did not conduct any manual intervention and the adaptation was conducted in a fully automatic manner. In the wrapper learning component stage, IEKA learns a wrapper for Figure 3 from the machine labeled training examples. Although some of the machine labeled training examples are incomplete and contain missing items, IEKA can still learn the hierarchical record structure and extraction rules from incomplete examples.

The newly learned wrapper was then applied to Web pages within the same site as shown in Figure 3. We obtained very promising results: the precision and recall for book title are 100.0% and 90.0% respectively; the precision and recall for author are 94.4% and 90.3% respectively; the precision and recall for final price are

100.0% and 87.5% respectively<sup>3</sup>. Hence, the extraction performance with our IEKA framework is much better than the extraction performance without IEKA.

## 8 Experimental Results

We conducted extensive experiments on several real-world Web sites in two different domains, namely, book domain and consumer electronics domain, to demonstrate the performance of IEKA. Table 4 depicts the Web sites used in our experiment. The first column shows the Web site labels. The second column shows the names of the Web site and the corresponding Web site addresses. The third and fourth columns depict the number of pages and the number of records collected in the Web site respectively. T1 - T3 and S1 - S10 are the Web sites from the book domain. T1 - T3 are used for parameter tuning while S1 - S10 are used for testing. T4 - T6 and S11 - S17 are the Web sites from the consumer electronics domain. T4 - T6 are used for parameter tuning while S11 - S17 are used for testing.

In each domain, We first provide five training examples in each Web site to learn the wrapper. After obtaining the wrapper for each of the Web sites, we conducted two sets of experiments to demonstrate the performance of IEKA. The first set of experiments is to simply apply the learned wrapper from one particular Web site without IEKA to all the remaining sites for information extraction. This experiment can be treated as a baseline for our adaptation approach. The other set of experiments is to adapt the learned wrapper from one particular Web site with IEKA to all the remaining sites. The extraction performance is evaluated by two metrics called *precision* and *recall*. Precision is defined as the number of items for which the system correctly identified divided by the total number of items it extracts. Recall is defined as the number of items for which the system correctly identified divided by the total number of actual items.

In IEKA, three parameters are required to be determined in advance. The first parameter is the threshold  $\theta$  in the modified nearest neighbour classification model as described in Section 4.3. The second and third parameters are the weight  $w$  in the text fragment classification model and the  $N$  in the  $N$ -best potential training text fragments as described in Section 6 respectively. We randomly chose the Web sites, T1, T2 and T3 from the book domain, and T4, T5 and T6 from the consumer electronics domain for tuning these parameters. We exhaustively conducted trials on these Web sites with different parameter settings. The average of the F-measure

	Book title				Author				Price			
	Without IEKA		With IEKA		Without IEKA		With IEKA		Without IEKA		With IEKA	
	P	R	P	R	P	R	P	R	P	R	P	R
S1	0.0	0.0	69.1	94.7	0.0	0.0	62.7	58.0	0.0	0.0	66.5	80.4
S2	0.0	0.0	57.8	81.4	0.0	0.0	62.8	77.4	0.0	0.0	62.8	77.1
S3	0.0	0.0	59.6	80.8	0.0	0.0	46.0	54.4	0.0	0.0	58.2	77.3
S4	0.0	0.0	49.3	72.1	0.0	0.0	46.6	60.6	0.0	0.0	58.3	77.7
S5	0.0	0.0	64.3	83.4	0.0	0.0	34.1	47.9	0.0	0.0	60.7	79.6
S6	0.0	0.0	64.0	74.4	0.0	0.0	45.7	54.2	0.0	0.0	58.3	77.2
S7	0.0	0.0	74.0	78.9	0.0	0.0	67.5	67.5	0.0	0.0	58.2	78.5
S8	0.0	0.0	58.6	63.3	0.0	0.0	69.3	68.6	0.0	0.0	65.7	78.2
S9	0.0	0.0	69.1	86.1	0.0	0.0	70.5	84.1	0.0	0.0	66.2	80.0
S10	0.0	0.0	77.9	92.0	0.0	0.0	77.6	79.6	0.0	0.0	0.0	0.0

Table 6: Average extraction performance on *title*, *author*, and *price* for the book domain for the cases of without adaptation and with adaptation when training examples of one particular information source are provided. (P and R refer to precision and recall respectively.)

is used for the evaluation of the parameter settings<sup>4</sup>. For each item in each domain, the parameter setting which achieves the best extraction performance were used in the subsequent testing tasks in our experiments.

**8.1 Book Domain** In the book domain, the items of interest are *book title*, *author*, and *price*. In the first set of experiments, we simply applied the wrapper learned from one particular Web site without IEKA to all the remaining Web sites for information extraction. For example, the wrapper learned from S1 is directly applied to S2 - S10 to extract items. We find that no wrapper is able to extract records from other Web sites. In addition, we also used WIEN [14] to perform the same adaptation task<sup>5</sup>. The wrapper learned by WIEN for a particular Web site cannot extract items in other Web sites.

Table 5 shows the results of the second set of experiments for the book domain. The first column shows the Web sites (source sites) from which the wrappers are learned with manually given training examples. The first row shows the Web sites (new unseen sites) to which the learned wrapper of a particular Web site is adapted. Each cell in Table 5 is divided into two sub-columns and three sub-rows. The three sub-rows represent the extraction performance on the items book title, author and price respectively. The two sub-columns represent the precision (P) and recall (R) for extracting the items respectively. These results are obtained by adapting a learned wrapper from one Web site to the remaining sites using our IEKA framework. The results indicate that the extraction performance is very satisfactory. Table 6 summarizes the average extraction performance on title, author, and price respectively for the cases of without IEKA and with IEKA when training examples of one particular Web site are provided. The first column shows the Web sites where training examples are

<sup>3</sup>The definitions of precision and recall can be found in Section 8.

<sup>4</sup>F-measure =  $(2 \times \text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$

<sup>5</sup>WIEN is available in the Web site: <http://www.cs.ucd.ie/staff/nick/research/research/wrappers/wien/>.

	S1		S2		S3		S4		S5		S6		S7		S8		S9		S10	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
S1	-	-	44.6	100.0	91.8	100.0	98.3	94.4	21.2	100.0	74.8	96.0	100.0	90.0	48.7	95.0	42.3	87.2	100.0	90.0
	-	-	98.9	73.3	88.8	99.1	44.4	12.8	67.3	69.4	44.8	37.9	93.0	46.9	99.0	95.0	28.4	87.2	0.0	0.0
	-	-	58.7	100.0	100.0	99.1	100.0	95.2	78.0	78.0	100.0	99.2	100.0	87.5	33.3	90.0	28.5	74.5	0.0	0.0
S2	34.6	70.4	-	-	100.0	100.0	93.7	94.4	23.0	100.0	98.3	96.0	99.1	90.0	26.0	95.0	45.3	87.2	0.0	0.0
	100.0	54.0	-	-	74.0	99.1	44.1	12.0	67.3	69.4	86.1	100.0	47.2	90.3	49.7	95.0	28.0	87.2	68.8	90.0
	26.1	71.0	-	-	100.0	99.1	100.0	95.2	78.0	78.0	100.0	99.2	100.0	87.5	33.3	90.0	28.2	73.6	0.0	0.0
S3	53.3	70.4	44.4	100.0	-	-	76.1	94.4	42.9	100.0	0.0	0.0	95.6	90.0	96.0	95.0	28.0	87.2	100.0	90.0
	97.8	71.0	3.9	4.2	-	-	51.2	16.8	67.3	69.4	84.9	100.0	30.8	46.0	49.5	95.0	28.4	87.2	0.0	0.0
	26.1	71.0	58.1	100.0	-	-	100.0	95.2	78.0	78.0	100.0	99.2	100.0	87.5	33.3	90.0	28.5	74.5	0.0	0.0
S4	96.7	70.4	57.1	100.0	50.2	100.0	-	-	32.4	100.0	96.7	96.0	0.0	0.0	70.4	95.0	39.8	87.2	0.0	0.0
	0.0	0.0	94.6	73.3	0.0	0.0	-	-	0.0	0.0	49.6	100.0	94.4	90.3	86.4	95.0	26.1	87.2	68.3	100.0
	26.1	71.0	58.7	100.0	100.0	99.1	-	-	78.0	78.0	100.0	99.2	100.0	87.5	33.3	90.0	28.5	74.5	0.0	0.0
S5	55.7	70.4	28.2	73.3	96.6	100.0	86.1	94.4	-	-	71.7	96.0	100.0	90.0	33.0	95.0	41.4	81.9	66.3	50.0
	42.7	54.0	0.0	0.0	100.0	99.1	0.0	0.0	-	-	96.1	100.0	0.9	0.9	50.0	90.0	17.5	87.2	0.0	0.0
	26.1	71.0	58.1	100.0	100.0	99.1	100.0	95.2	-	-	100.0	99.2	100.0	87.5	33.3	90.0	28.5	74.5	0.0	0.0
S6	1.5	0.8	45.1	73.3	98.2	100.0	75.2	94.4	30.5	100.0	-	-	97.3	90.0	87.2	34.0	42.1	87.2	99.0	90.0
	50.0	54.0	86.7	65.0	0.0	0.0	0.0	0.0	1.0	6.1	-	-	94.4	90.3	60.1	95.0	20.0	87.2	99.0	90.0
	26.1	71.0	58.7	100.0	100.0	99.1	100.0	95.2	78.0	78.0	-	-	100.0	87.5	33.3	90.0	28.2	73.6	0.0	0.0
S7	96.7	70.4	59.1	73.3	84.8	100.0	95.9	94.4	91.7	100.0	0.0	0.0	-	-	96.9	95.0	42.1	87.2	99.0	90.0
	100.0	54.0	100.0	73.3	84.7	99.1	50.6	34.4	66.0	69.4	6.7	4.8	-	-	50.0	95.0	84.5	87.2	65.1	90.0
	26.1	71.0	58.4	100.0	100.0	99.1	100.0	95.2	78.0	78.0	100.0	99.2	-	-	33.3	90.0	28.2	73.6	0.0	0.0
S8	88.9	70.4	42.7	73.3	97.4	100.0	2.1	0.8	0.9	2.0	70.8	96.0	100.0	90.0	-	-	26.1	87.2	98.2	50.0
	48.9	71.0	97.8	73.3	83.5	99.1	89.5	27.2	67.3	69.4	100.0	100.0	94.4	90.3	-	-	42.7	87.2	0.0	0.0
	26.1	71.0	58.7	100.0	100.0	99.1	100.0	95.2	78.0	78.0	100.0	99.2	100.0	87.5	-	-	28.2	73.6	0.0	0.0
S9	47.8	70.4	47.2	100.0	98.2	100.0	98.3	94.4	32.6	100.0	73.5	96.0	98.2	90.0	26.2	34.0	-	-	100.0	90.0
	56.3	54.0	42.5	73.3	98.2	99.1	49.6	94.4	96.1	100.0	99.2	100.0	94.0	55.8	33.2	90.0	-	-	65.1	90.0
	26.1	71.0	58.1	100.0	100.0	99.1	100.0	95.2	78.0	78.0	100.0	99.2	100.0	87.5	33.3	90.0	-	-	0.0	0.0
S10	96.7	70.4	57.4	100.0	96.6	100.0	100.0	94.4	32.5	100.0	99.2	96.0	94.7	90.0	97.8	90.0	26.2	87.2	-	-
	97.8	71.0	91.8	65.0	100.0	97.3	95.0	90.4	67.3	69.4	97.6	100.0	78.8	46.0	42.3	90.0	28.3	87.2	-	-
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-	-

Table 5: Experimental results of adapting a learned wrapper from one information source to the remaining information sources in the book domain. P and R refer to precision and recall respectively.)

given. Each row summarizes the results obtained by using the learned wrapper of the Web site in the first column and applying to all other sites for extraction. The results indicate that the wrapper learned from a particular Web site cannot be directly applied to others without IEKA for information extraction. After applying IEKA, the wrapper learned from a particular Web site can be adapted to other sites. A very promising performance is achieved especially compared with the performance obtained without adaptation. S10 cannot extract the price item from other Web sites. The reason is that the price item of the records in S10 is displayed with a totally different format.

**8.2 Consumer Electronics Domain** In the consumer electronics domain, the items of interest are *model number*, *description*, and *price*. Table 7 summarizes the average extraction performance on different items for the cases of without IEKA and with IEKA when training examples of one particular Web site are provided. The results indicate that the extraction of the items fails without IEKA in all cases. After applying our wrapper adaptation approach, the wrapper learned from a particular Web site can adapt to other sites. The results show that our adaptation approach achieves a very satisfactory extraction performance for the model number and price. The extraction performances for the model number are not as good as others for S13 and S17. The reason is that the model number in S13 is particularly short while the model number in S17 is particularly long. The extraction performance on description is fair. This is due to the fact that most of

	Model number				Description				Price			
	Without IEKA		With IEKA		Without IEKA		With IEKA		Without IEKA		With IEKA	
	P	R	P	R	P	R	P	R	P	R	P	R
S11	0.0	0.0	50.5	54.9	0.0	0.0	31.3	51.0	0.0	0.0	79.1	97.6
S12	0.0	0.0	34.8	46.0	0.0	0.0	22.0	31.0	0.0	0.0	84.8	97.4
S13	0.0	0.0	27.6	31.3	0.0	0.0	28.8	31.0	0.0	0.0	84.8	97.4
S14	0.0	0.0	49.5	50.4	0.0	0.0	15.9	30.1	0.0	0.0	79.3	97.8
S15	0.0	0.0	41.3	48.1	0.0	0.0	24.6	48.5	0.0	0.0	79.1	99.6
S16	0.0	0.0	45.7	48.3	0.0	0.0	21.9	31.0	0.0	0.0	86.1	81.1
S17	0.0	0.0	22.0	33.0	0.0	0.0	47.5	54.4	0.0	0.0	87.0	81.4

Table 7: Average extraction performance on *model number*, *description*, and *price* for the electronic appliance domain for the cases of without adaptation and with adaptation when training examples of one particular Web site are provided. (P and R refer to precision and recall respectively.)

the content of the description item involves a large portion of free text. The content of the description item in different Web sites is very different and the site invariant features are not explicit.

## 9 Conclusions and Future Work

We describe our framework called IEKA for adapting information extraction knowledge. By modeling Web pages with site invariant features and site dependent features, IEKA is able to adapt the previously learned extraction knowledge from a source Web site to a new unseen site. Some site invariant features can be derived from the previously learned extraction knowledge and the partially specified training examples which refer to the items previously extracted or collected in the source Web site. IEKA generates the machine labeled training examples automatically. Then both site invariant features and site dependent features are considered for these machine labeled training examples. Finally IEKA can learn the new information extraction knowledge for the new unseen site. Several experiments on real-world

Web sites have been conducted to demonstrate the performance of IEKA.

One possible direction for future work is to incorporate domain specific knowledge from users. Very often, users may already have some background information or knowledge about the domain. For example, users may have a prior knowledge about the format of the items. We intend to develop a mechanism in which users can incorporate their domain specific knowledge easily. Another possible direction is to integrate some shallow natural language processing techniques for handling the free text portion. More site invariant features may be derived to enhance the performance.

## References

- [1] S. Brin. Extracting patterns and relations from the World Wide Web. In *Proceedings of the International Workshop on the Web and Databases*, pages 172–183, 1998.
- [2] C. Chang and S. C. Lui. IEPAD: information extraction based on pattern discovery. In *Proceedings of the Tenth International Conference on World Wide Web*, pages 681–688, 2001.
- [3] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the Information Processing Society of Japan*, pages 7–18, 1994.
- [4] W. W. Cohen, M. Hurst, and L. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 232–241, 2002.
- [5] V. Crescenzi, G. Mecca, and P. Merialdo. ROAD-RUNNER: Towards automatic data extraction from large web sites. In *Proceedings of the 27th Very Large Databases Conference*, pages 109–118, 2001.
- [6] Defense Advanced Research Projects Agency. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Morgan Kaufmann Publisher, Inc., 1998.
- [7] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agents*, pages 39–48, February 1997.
- [8] J. C. French, A. L. Powell, and E. Schulman. Applications of approximate word matching in information retrieval. In *Proceedings of the Sixth International Conference on Information and Knowledge Management*, pages 9–15, November 1997.
- [9] R. Ghani and R. Jones. A comparison of efficacy and assumptions of bootstrapping algorithms for training information extraction systems. In *Proceedings of the workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Data at the Linguistic Resources and Evaluation Conference*, 2002.
- [10] P. Golgher and A. da Silva. Bootstrapping for example-based data extraction. In *Proceedings of the Tenth ACM International Conference on Information and Knowledge Management*, pages 371–378, 2001.
- [11] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.
- [12] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semistructured information from the web. In *Proceedings of the Workshop on Management of Semistructured Data*, 1997.
- [13] N. Kushmerick. Regression testing for wrapper maintenance. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 74–79, 1999.
- [14] N. Kushmerick and B. Grace. The wrapper induction environment. In *Proceedings of the Workshop on Software Tools for Developing Agents (AAAI-98)*, pages 131–132, 1998.
- [15] N. Kushmerick and B. Thomas. Adaptive information extraction: Core technologies for information agents. In *Intelligent Information Agents R&D In Europe: An AgentLink Perspective*, pages 79–103, 2002.
- [16] K. Lerman and S. Minton. Learning the common structure of data. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 609–614, 2000.
- [17] K. Lerman, S. Minton, and C. Knoblock. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, pages 149–181, 2003.
- [18] W. Y. Lin and W. Lam. Learning to extract hierarchical information from semi-structured documents. In *Proceedings of the Ninth International Conference on Information and Knowledge Management CIKM*, pages 250–257, 2000.
- [19] L. Liu, C. Pu, and W. Tang. WebCQ - Detecting and delivering information changes on the web. In *Proceedings of the Ninth International Conference on Information and Knowledge Management CIKM*, pages 512–519, 2000.
- [20] I. Muslea, S. Minton, and C. Knoblock. Selective sampling with redundant views. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 621–626, 2000.
- [21] I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1-2):93–114, 2001.
- [22] R. Srihari and W. Li. Question answering supported by information extraction. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 185–196, 1999.
- [23] J. Wang and F. H. Lochovsky. Data extraction and label assignment for Web databases. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 187–196, 2003.
- [24] T. L. Wong and W. Lam. Adapting information extraction knowledge for unseen web sites. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 506–513, 2002.