Using the Structure of Web Sites for Automatic Segmentation of Tables

Kristina Lerman[1], Lise Getoor[2], Steven Minton[3] and Craig Knoblock[1]
1. USC Information Sciences Institute
Marina del Rey, CA 90292
2. University of Maryland
College Park, MD 20742
3. Fetch Technologies
Manhattan Beach, CA 90266
{lerman,knoblock}@isi.edu, getoor@cs.umd.edu, minton@fetch.com

ABSTRACT

Many Web sites, especially those that dynamically generate HTML pages to display the results of a user's query, present information in the form of list or tables. Current tools that allow applications to programmatically extract this information rely heavily on user input, often in the form of labeled extracted records. The sheer size and rate of growth of the Web make any solution that relies primarily on user input is infeasible in the long term. Fortunately, many Web sites contain much explicit and implicit structure, both in layout and content, that we can exploit for the purpose of information extraction. This paper describes an approach to *automatic* extraction and segmentation of records from Web tables. Automatic methods do not require any user input, but rely solely on the layout and content of the Web source. Our approach relies on the common structure of many Web sites, which present information as a list or a table, with a link in each entry leading to a detail page containing additional information about that item. We describe two algorithms that use redundancies in the content of table and detail pages to aid in information extraction. The first algorithm encodes additional information provided by detail pages as constraints and finds the segmentation by solving a constraint satisfaction problem. The second algorithm uses probabilistic inference to find the record segmentation. We show how each approach can exploit the web site structure in a general, domain-independent manner, and we demonstrate the effectiveness of each algorithm on a set of twelve Web sites.

1. INTRODUCTION

The World Wide Web is a vast repository of information. The amount of data stored in electronic databases accessible to users through search forms and dynamically generated Web pages, the so-called *hidden Web* [26], dwarfs the amount of information available on static Web pages. Unfortunately, most of this information is presented in a form accessible only to a human user, *e.g.*, list or tables that visually lay out relational data. Although newer technologies, such as XML and the Semantic Web, address this problem directly, only a small fraction of the information on the Web is semantically labeled. The overwhelming majority of the available data has to be accessed in other ways.

Web wrappers are popular tools for efficiently extracting information from Web pages. Much of the research in this area over the last decade has been concerned with quick and robust construction of Web wrappers [14], usually with the help of machine learning techniques. Because even the most advanced of such systems learn correct wrappers from examples provided by the user, the focus recently has been on minimizing the number of examples the user has to label, *e.g.*, through active learning [21]. Still, even when user effort is significantly reduced, the amount and the rate of growth of information on the Web will quickly overwhelm user resources. Maintaining wrappers so that they continue to extract information correctly as Web sites change, requires significant effort, although some progress has been made on automating this task [18]. Heuristic techniques that may work in one information domain are unlikely to work in another. A domain-independent, fully automatic solution that requires no user intervention is the Holy Grail of information extraction from the Web. Despite the inherent difficulty of the problem, there *are* general principles and algorithms that can be used to automatically extract data from structured web sites. In this paper, we present new novel techniques that are applicable to a broad range of hidden Web sources.

Extraction of records or tuples of data from lists or tables in HTML documents is of particular interest, as the majority of Web sites that belong to the hidden Web are presented in this manner. Record extraction is required for a multitude of applications, including web data mining and question-answering. The main challenge to automatic extraction of data from tables is the great variability in HTML table styles and layout. A naive approach based on using HTML `<table>` tags will not work. Only a fraction of HTML tables

SIGMOD 2004 June 13-18, 2004, Paris, France.

are actually created with <table> tags, and these tags are also used to format multi-column text, images, and other non-table applications. The vast majority of HTML documents use non-standard tags to format tables, including text separators, such as ~, to separate fields and
 to separate different items as well as fields. More sophisticated automatic approaches to table recognition and information extraction have been suggested which rely on the Document Object Model [13] or regularities in HTML tags [8, 7]. These approaches are not robust and will fail for some sites. It is our experience that the variability in HTML tags is often too great to rely on them for table recognition and information extraction.

Fortunately, dynamically generated Web pages contain much explicit and implicit structure, *both in layout and content*, that we can exploit for purposes of automatic information extraction. Previous approaches have focused on exploiting structure within a page [17, 8, 1]; here we use the Web site's structure to improve information extraction.

Web sites belonging to the hidden Web have a surprisingly uniform structure. The entry point is an HTML form for the user to input her query. The result of the query is a list of items or a collection of records from a database. The results are usually displayed as a list or a table on an automatically generated page. We call such a results page the *list page*. Each item or record often has a link to a *detail page* that contains additional information about that entry. Detail pages are also generated automatically and populated with results of database queries. Some of the item attributes are likely to appear on the list page as well as on the detail page.

As others have noted, there is important structure in the layout of the list page: some HTML tags are used to separate records, and other HTML tags or text symbols are used to separate columns. In addition to similarities in layout, we expect similarities in the content of data: data in the same column should be of the same type, name or phone number for instance. Of course, the underlying structure of real world data may not be immediately clear: an item may have missing columns; column separators may be different, depending on whether the attribute value is present or missing, or attribute values may be formatted in different ways; a data field may have a lot of variability that we are not able to capture if we assume a uniform layout for each row.

In this paper we propose methods that allow us to efficiently segment data on the list page into records using information contained in detail pages. We describe two approaches: one formulates the task as a constraint satisfaction problem (CSP) and the other uses a probabilistic inference approach. Both techniques exploit the additional information provided by the overlap in the content between list and detail pages. The idea we are trying to leverage is the fact that each detail page represents a separate record. The constraint satisfaction-based technique encodes the relations between data found on the list and detail pages as logical constraints. Solving a constraint satisfaction problem yields a record segmentation. In the probabilistic approach, the information provided by detail pages is used to learn parameters of a probabilistic model. Record segmentation is the assignment of attributes to records that maximizes the likelihood of the data given the model. In addition to computing the record segmentation, the probabilistic approach produces a mapping of data to columns.

The approaches we describe are novel, in that unlike many other techniques, they rely on the content of Web pages rather than their layout (*i.e.*, tags). The number of text strings on a typical Web page is very small compared to the number of HTML tags; therefore, inference algorithms that rely on content will be much faster than the algorithms that use layout features. Both of our approaches are fully automatic, domain independent and unsupervised, *i.e.*, they do not require any training or user-labeled data. We have validated both methods on 12 sites from diverse information domains, including white pages, property tax and corrections domains. Despite the great variability in the appearance, presentation, and data across these sites, both approaches performed quite well.

2. RELATED WORK

Several researchers have addressed the problem of detecting tables in Web and plain text documents and segmenting them into records.

2.1 Table Extraction from HTML Documents

Existing approaches to extracting table data from Web documents can be classified as heuristic or machine learning. Heuristic approaches to detecting tables and record boundaries in Web documents include using the Document Object Model (DOM) and other features [13] to identify tables. Domain-specific heuristic rules that rely on features such as percent signs and date/time formats have also been tried successfully [5].

Machine-learning approaches learn a model of data from a set of labeled training examples using hand-selected features. Borkar et al. [3] use multiple heuristic features, including domain-specific controlled vocabularies, to learn a Hidden Markov-based probabilistic model from a set of training examples. Hurst [11] trained a Naive Bayes classifier, while Wang et al. [29] describe a domain-independent classifier that uses non-text layout features (average number of columns/rows, average cell length and consistency) and content features (image, form, hyperlink, alphabetic, digit, others). The classifier achieves good performance after being trained on thousands of labeled examples. In a related work, Wang et al. [31, 30] optimizes whole page segmentation probability over spatial layout features, much as it is done in document image analysis, to find correct tables in Web documents. Cohen et al. [6] present another example of an approach that combines alternate representations, text, DOM and non-text layout features, with a learning algorithm to improve the wrapper learning process. These methods require many training examples in order to learn a useful model of data.

Clearly, the methods listed above suffer from being domain-specific or requiring user-labeled training examples. Recently, several unsupervised learning approaches, which require no training examples, have been investigated. Yoshida [32] automatically recognized table structures on the basis of probabilistic models where parameters are estimated using the Expectation Maximization algorithm. This approach

was validated on two domains on the information extraction and integration task with 78% accuracy.

The RoadRunner system [8, 7] automatically extracts data from Web sites by exploiting similarities in page layout. The premise behind the system is that many Web pages are generated by a grammar, which can be inferred from example pages. Thus, RoadRunner can learn the table template and use it to automatically extract data from the Web site. RoadRunner's authors focus on a subclass of Web pages that can be generated by union-free grammar and describe the algorithm to learn the grammar. The learning algorithm is exponential, and further simplifications are necessary to keep it computationally tractable. Although the RoadRunner system uses an elegant approach and method for automatic extraction of data, its applicability is limited, because union-free grammars do not allow for disjunctions, and disjunctions appear frequently in the grammar of Web pages. Disjunctions are necessary to represent alternative layout instructions often used by Web sites for a same field. Our approach, in contrast, is able to handle disjunctions.

Chang & Lui [4] present an algorithm based on PAT trees for detecting repeated HTML tag sequences that represented rows of Web tables. They then apply the sequences to automatically extract data from Web search engine pages. Although they show good performance in this domain, search engine pages are much simpler than HTML pages containing tables that are typically found on the Web. We have tried a similar approach and found that it had limited utility when applied to most Web pages. Other recent layout-based automatic extraction algorithms, such as [1], have been shown to work well on detail pages, but cannot handle lists.

2.2 Table Extraction from Plain Text

Automatic table extraction from plain text documents is a line of research parallel to the work on HTML table extraction. There are differences between plain text and HTML tables that make the two fundamentally different problems. Plain text documents use white space and new line for the purpose of formatting tables: new lines are used to separate records and white spaces are used to separate columns, among other purposes. Record segmentation from plain text documents is, therefore, a much easier task. Closely linking format and content in plain text documents also gives rise to new challenges. In plain text tables, a long attribute value that may not fit in a table cell will be broken between two lines, creating a non-locality in a text stream. An automatic algorithm will have to associate part of a string with another string that will appear arbitrarily later in the text stream. This problem does not usually arise in HTML documents. On the other hand, HTML tables vary widely in their layout and formatting conventions, making it difficult to rely on any set of features to be good row or column separators. Although one may employ the same set of tools for extracting from HTML and plain text tables, specialized algorithms that address the conventions of each domain may outperform a general algorithm.

Despite these differences, a similar variety of techniques has been used in table extraction from plain text and for HTML documents, from heuristic rules [12] to machine learning algorithms that learn a feature or a combination of features that predict a table and its elements [22, 23]. Pyrredy et al. [24] propose a unique approach that uses structural features, such as alignment of characters on a page, to identify the tables and its main elements. As we do, Pinto et al. [23] capitalize on recent developments in probabilistic models, and examine their use in extracting tables from plain text documents and identifying their structure. The authors are interested in identifying header and data rows, not in segmenting data into individual records and attributes. Their probabilistic models are trained on user-labeled examples using a feature set that includes white space, text and separator features. White spaces are used heavily to format tables in plain text documents. In HTML documents, on the other hand, white space is almost never used for this purpose. Although their methods may, in principle, work for HTML pages, the authors have not applied them for this purpose. It is likely that they will need a wide variety of training examples to capture the range of variability in Web documents. Our approach, in contrast, requires no training data.

3. OVERVIEW OF THE PROBLEM

In this section we give an overview of the problem of record extraction and segmentation using the structure of a Web site to aid in extraction. As we discussed above, many Web sites that present information contained in databases follow a *de facto* convention in displaying information to the users and allowing them to navigate it. This convention affects how the Web site is organized, and gives us additional information we can leverage for information extraction. Such Web sites generate list and detail pages dynamically from templates and fill them with results of database queries. Figure 1 shows example list and detail pages from the Verizon Superpages site. The Superpages site allows customers to search over 16 million yellow page listings and a national white pages database by name, phone number or business type. As shown in the figure, the results returned for a search include the fields, name, address, city, state, zip and phone. Here the text "More Info" serves as a link to the detail page. Note that list and detail pages present two views of the record. Using automatic techniques, we can potentially combine the two views to get a more complete view of the record. For example, maps of the addresses are shown on the detail pages in Figure 1, but absent from the list pages.

We envision an application where the user provides a pointer to the top-level page — index page or a form — and the system automatically navigates the site, retrieving all pages, classifying them as list and detail pages, and extracting structured data from these pages. We are already close to this vision [19]. The current paper addresses the technical issues involved in the problem of structured information extraction from list pages.

3.1 Page Templates

Consider a typical list page from a Web site. As the server constructs the list page in response to a query, it generates a header containing the company logo, followed in most cases by an advertisement, then possibly a summary of the results, such as ``Displaying 1-10 of 214 records.'', table header and footer, followed by some concluding remarks, such as a copyright information or navigation aids. In the

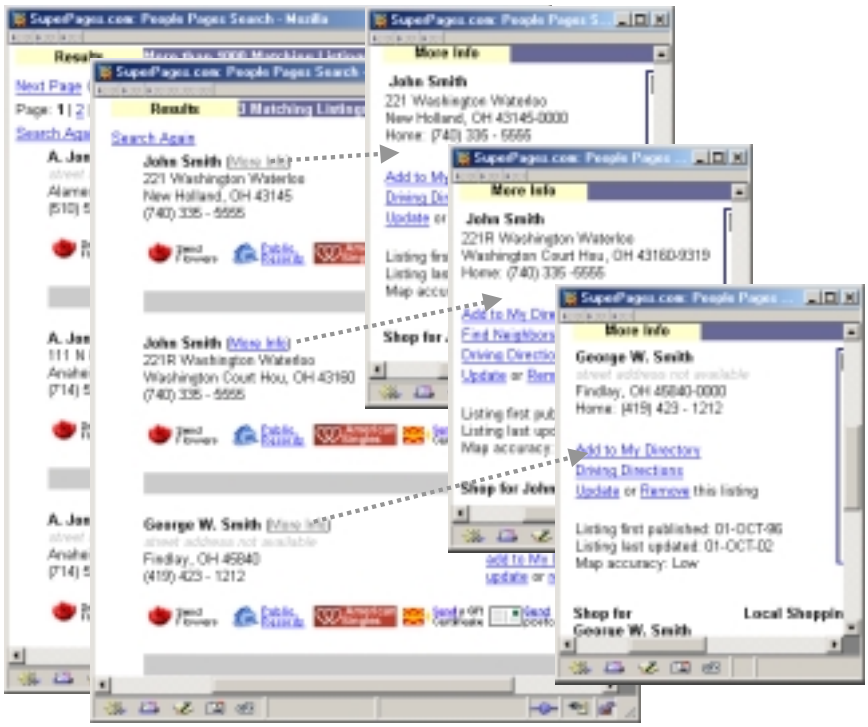

Figure 1: Example list and detail pages from the Superpages site (identifying information has been removed to preserve confidentiality).

above example, the header includes `Results`, `3 Matching Listings`, `Search Again` and the associated HTML. The footer includes advertisement and navigational links. We call this part of the page the *page template*. The page template of a list page contains data that is shared by all list pages and is invariant from page to page. A different page template is used to generate detail pages. As the server writes out records to the table, it uses a *table template*. This template contains layout information for the table, that is the HTML tags or ASCII characters used to separate columns and rows, or format attribute values.

Given two, or preferably more, example list pages from a site, we can derive the template used to generate these pages and use it to identify the table and extract data from it. The process starts with a set of list pages from a site and a set of detail pages obtained by following links from one of the list pages. The pages are tokenized — the text is split into individual words, or more accurately tokens, and HTML escape sequences are converted to ASCII text. Each token is assigned one or more syntactic types [16, 18], based on the characters appearing in it. The three basic syntactic types we consider are: HTML, punctuation, and alphanumeric. In addition, the alphanumeric type can be either numeric or alphabetic, and the alphabetic can be capitalized, lowercased or allcaps. This gives us a total of eight (non-mutually exclusive) possible token types. A template finding algorithm (*e.g.*, one described in [1, 18]) is used to extract the main page template.

Slots are sections of the page that are not part of the page template. If any of the tables on the pages contain more than two rows, the tags specifying the structure of the table will not be part of the page template, because they will appear more than once on that page. Likewise, table data will also not be part of the page template, since it varies from page to page. Therefore, the entire table, data plus separators, will be contained in a single slot. Considering that tables usually contain a significant amount of data, we use a heuristic that the table will be found in the slot that contains the largest number of text tokens.

3.2 Data Extraction

Next, we extract data from the table. We do this simply by extracting, from the slot we believe to contain the table, the contiguous sequences of tokens that do not contain separators. Separators are HTML tags and special punctuation characters (any character that is not in the set ".,()-"). Practically speaking, we end up with all visible strings in the table. We call these sequences *extracts*, $E = \{E_1, E_2, \ldots, E_N\}$. These are the attribute values that the records in the table contain, and possibly some extraneous data, such as "More Info", "Send Flowers" as shown in Figure 1. Our goal is to segment these extracts into individual records.

The CSP and probabilistic approaches share the same basic premise: detail and list pages present two views of the same record and each detail page corresponds to a distinct record. The labels for detail page are: $\{r_1, r_2, \ldots, r_K\}$.

For each extract E_i, we record all detail pages on which it

was observed, D_i. [1] The extracts are checked in the order they appear in the text stream of the list page. If an extract appears in all the list pages or in all the detail pages, it is ignored: such extracts will not contribute useful information to the record segmentation task.

The methods presented below are appropriate for tables that are laid out horizontally, meaning that the records are on separate rows. A table can also be laid out vertically, with records appearing in different columns; fortunately, few Web sites lay out their data in this way. In horizontally laid out tables, the order in which records appear in the text stream of the page is the same as the order in which they appear in the table. In other words, any attribute of the second record will appear in the stream after all the attributes of the first record and before any of the attributes of the third record have been observed.

Table 1 is an example of a table of observations of extracts on detail pages from the Superpages site (see Figure 1).[2] The columns correspond to extracts and they are displayed in the order they appear on the list page.

As can be seen from the table, the same extract can appear on multiple detail pages. In the Superpages site, for example, several people may have the same name or the phone number. Note that only the extracts or record attributes that appear in both the list and detail pages are used. There may be other potential attributes, but if they do not appear on any of the details pages, they will not be considered in the analysis. This works to our advantage by reducing the need to precisely identify the table slot.

3.3 Record Segmentation

Extracts common to list and detail pages give us an additional source of information that we can use to segment data into individual records. An extract (attribute value) belongs to a record only if it appears on the detail page corresponding to that record. The same extract cannot be assigned to more than one record or more than once to the same record. Thus, in the table above, we can use these rules to assign E_1, E_2, E_3 and E_4 to the first record, and E_5, E_6, E_7 and E_8 to the second record, as shown in Table 2, even though E_1 and E_4 appear in both records. These rules can be easily encoded in both the CSP and the probabilistic framework. Solving this problem yields an assignment of data to records.

3.4 Column Extraction

The probabilistic model is more expressive than the CSP. In addition to record segmentation, we can learn a model for predicting the column of an extract, based not only on its token type, but also on the neighboring columns. We can learn a probabilistic model of the data given the column label: *i.e.*, first attribute of the record is of alphabetic type, the second a numeric type, etc. The inference algorithm estimates parameters of the model from the observations of extracts on details pages. The parameters are then used to find a column assignment that maximizes the total probability of the observations given the model. The column labels will be $L_1, \ldots, L_k$ (we can find k by the longest potential sequence of r_i in the data). To provide them with more semantically meaningful labels, we can use other automatic extraction techniques, such as those described in the Roadrunner system [2].

[1]The string matching algorithm ignores intervening separators on detail pages. For example, a string "FirstName LastName" on list page will be matched to "FirstName
 LastName" on the detail page.

[2]All identifying information has been changed to preserve anonymity.

4. A CSP APPROACH TO RECORD SEGMENTATION

CSPs are stated as logical expressions, or constraints, over a set of variables, each of which can take a value from a finite domain (Boolean, integer, *etc.*). The case where the variables and logical formulas are boolean is known as Boolean satisfiability, the most widely studied area of CSP. In a pseudo-boolean representation, variables are 0-1, and the constraints can be inequalities. The CSP problem consists of finding the value assignment for each variable such that all constraints are satisfied at the same time. When constraints are inequalities, the resulting problem is an optimization problem.

4.1 Structure Constraints

We encode the record segmentation problem into pseudo-boolean representation and solve it using integer variable constraint optimization techniques. Let x_{ij} be the assignment variable, such that $x_{ij} = 1$ when extract E_i is assigned to record r_j. and $x_{ij} = 0$ when the extract E_i is not part of the record r_j. The assignment of extracts to records will look something like the table in Table 2. Blank cells in table correspond to $x_{ij} = 0$. By examining this table, we see that there are two logical constraints it makes sense to impose on the assignment variables. First, there has to be exactly a single "1" in each column of the assignment table. This is the *uniqueness constraint*.

Uniqueness constraint: Every extract E_i belongs to exactly one record r_j.

Mathematically, the uniqueness constraint can be stated as $\sum_j x_{ij} = 1$. If necessary, we can make the constraint less rigid by requesting that every extract appear in at most one record. The relaxed constraint can be written as $\sum_j x_{ij} <= 1$.

The assignment table, Table 2, suggests a second constraint, what we call the *consecutiveness constraint*.

Consecutiveness constraint: only contiguous blocks of extracts can be assigned to the same record.

These constraints can be expressed mathematically in the following way: $x_{ij} + x_{kj} \leq 1$ when there is n, $k < n < i$, such that $x_{nj} = 0$. In other words, we cannot assign extract E_1 in Table 1 to r_2 along with extracts E_4, E_5, and E_6 because neither E_2 nor E_3 can be assigned to r_2. A better choice is to assign E_1 E_2 E_3 E_4 to r_1 and E_5 E_6 E_7 E_8 to r_2.

	E_1 John Smith	E_2 221 Wa ington...	E_3 New Holland...	E_4 (740) 335-5555	E_5 John Smith	E_6 221R Wa shington...	E_7 Wash ington...	E_8 (740) 335-5555	E_9 George W. Smith	E_{10} Findlay, OH...	E_{11} (419) 423-1212
D_i	r1,r2	r1	r1	r1,r2	r1,r2	r2	r2	r1,r2	r3	r3	r3

Table 1: Observations of extracts on detail pages D_i for the Superpages site

	E_1 John Smith	E_2 221 Wa ington...	E_3 New Holland...	E_4 (740) 335-5555	E_5 John Smith	E_6 221R Wa shington...	E_7 Wash ington...	E_8 (740) 335-5555	E_9 George W. Smith	E_{10} Findlay, OH...	E_{11} (419) 423-1212
r1	1	1	1	1							
r2					1	1	1	1			
r3									1	1	1

Table 2: Assignment of extracts to records

The observations and assignment tables are closely linked. If extract E_i was not observed on detail page r_j ($r_j \notin D_i$), then $x_{ij} = 0$, in other words, E_i cannot be assigned to r_j. If E_i was observed on detail page r_j, then x_{ij} is either 1 or 0. The constraints on the assignment variables can be easily written down from the observations data. For the Superpages site data shown in Table 1 the uniqueness and consecutiveness constraints are written below:

$$\begin{aligned} x_{11} + x_{12} &= 1 & x_{11} + x_{81} &<= 1 \\ x_{21} &= 1 & x_{21} + x_{81} &<= 1 \\ x_{31} &= 1 & x_{31} + x_{81} &<= 1 \\ x_{41} + x_{42} &= 1 & x_{31} + x_{81} &<= 1 \\ x_{51} + x_{52} &= 1 & x_{41} + x_{81} &<= 1 \\ x_{62} &= 1 & x_{12} + x_{42} &<= 1 \\ \vdots & & \vdots & \end{aligned}$$

4.2 Position Constraints

Detail pages present another source of constraints we can exploit, because in addition to the occurrence of an extract on a page, they provide information about the position of that extract. This information is summarized in Table 3 for the Superpages site. The horizontal rows are the positions on page j (*e.g.*, token number of the starting token) where the extracts were observed. Note that in Table 3 the first four positions are from detail page r_1, while the next four are from r_2. If extract E_i was observed in position $pos_k^j(E_i)$ on detail page r_k, the (i, j) cell in table has an entry "1"; otherwise, it is empty. It is clear that no two extracts assigned to the same record can appear in the same position on that page. The collorary is: if two extracts appear in the same position on the detail page, they must be assigned to different records. We express it more formally as

Position Constraint: if $pos_j(E_i) \neq pos_j(E_k)$, then E_i and E_k cannot both be assigned to r_j

In the example in Table 3, the position constraints are

$$\begin{aligned} x_{11} + x_{51} &= 1 \\ x_{12} + x_{52} &= 1 \\ x_{41} + x_{81} &= 1 \\ \vdots & \end{aligned}$$

These constraints can also be relaxed to produce inequalities.

After we have constructed the uniqueness, consecutiveness and position constraints on the assignment variables for a particular Web page, we solve them using WSAT(OIP) [27], an integer optimization algorithm [28]. The solution is the assignment of extracts to records. Results are presented in Section 6.

5. A PROBABILISTIC APPROACH TO RECORD SEGMENTATION

An alternate approach is to frame the record segmentation and extraction task as a probabilistic inference problem. Common probabilistic models for information extraction include hidden Markov models (HMMs) [25], and conditional random fields (CRFs) [15, 23]. In these approaches, a labeled training set is provided and the model is learned using standard probabilistic inference techniques; because the state is *hidden*, the common approach to learning the models is to use the expectation maximization (EM) algorithm. While these approaches have their computational limitations, they have been applied successfully to a wide range of problems beyond information extraction including speech recognition and robot navigation.

Unfortunately, here we are faced with a more challenging problem. We do **not** have a labeled training set to start from. The key to our success will be to:

Factor: We will factor the state space and observation set of the HMM to allow for more efficient learning (because fewer parameters will be required).

Bootstrap: We will use the information from the detail pages to help bootstrap the learning algorithm. The constraints from the detail extracts will provide useful information that can keep our learning algorithm on track.

Structure: We will use a hierarchical model to capture global parameters such as the length of the record, or the period, to make our inference more tractable. Note that while there is a global record length, the record lengths of the individual records may vary; for some records not all columns will be presented.

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
pos_1^{730}	1				1			
pos_1^{772}		1						
pos_1^{812}			1					
pos_1^{846}				1				1
pos_2^{536}	1				1			
pos_2^{578}				1				1
pos_2^{608}						1		
pos_2^{642}							1	

Table 3: Positions of extracts on detail pages. Entry of 1 means extract E_i was observed at position k on page j (pos_j^k).

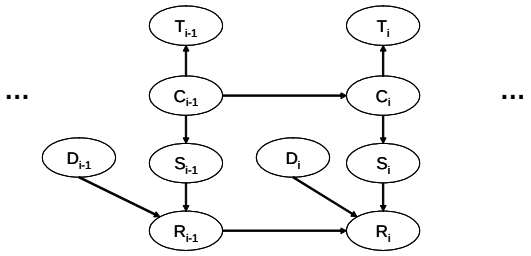

Figure 2: A probabilistic model for record extraction from list and detail pages.

We begin by describing the probablistic model that we will use (Section 5.1), then describe how the model is used to do record segmentation. (Section 5.2).

5.1 Probabilistic Model for Record Extraction

Figure 2 shows a graphical model representation for our problem. The basic variables in the model are the following:

$T = \{T_1, \ldots, T_n\}$: token types of extract E_i. Examples of token types are alphanumeric, capitalized, punctuation, as described earlier. We have 8 token types, so each T_i is represented as a vector $T_{i_1}, T_{i_2}, \ldots, T_{i_8}$.

$D = \{D_1, \ldots, D_n\}$: record numbers of the detail pages $D_i \subseteq 1, \ldots, K$ on which E_i occurred.

The above variables are *observed* in our data — given a list and set of detail pages, we can compute the values for the above variables.

In addition, we have the following set of *unobserved* variables:

$R = \{R_1, \ldots, R_n\}$: the record number of the extract. We assume these range from $1 \ldots K$ (these correspond to detail pages).

$C = \{C_1, \ldots, C_n\}$: the column label of the extract. We assume these range from $L_1, \ldots, L_k$ (a bound on this is the largest number of extracts found on a detail page).

$S = \{S_1, \ldots, S_n\}$: S_i is true if E_i is the start of a new record, false otherwise.

Of course, in addition to the variables, our model describes the dependencies between them. Rather than using the standard HMM representation, we use a factored representation [10, 20], which allows us to more economically model (and learn) the state transition probabilities. We have defined the factored structure of the model, as shown in Figure 2. Arrows indicate probabilistic dependencies. In some cases these dependencies are deterministic — for example if an extract appears on only one detail page, then we know the record to which it belongs. In other cases the dependencies are probabilistic — for example, the starting token type of a column might usually be a *capitalized* string, but occasionally will be an *HTML* tag. Here, we assume the structure of the dependencies (and where reasonable, the functional form for the dependencies) and we will learn the probabilities for the model using our observed data.

Our model assumes the following dependencies:

$P(T_i|C_i)$: The token type for extract i depends on the column label. For example, for the name field, the token type is likely to be *capitalized* token, but this is a probabilistic relationship.

$P(C_i|C_{i-1})$: The column label for extract i depends on the column label of the previous column. For example, the *address* field is likely to follow the *name* field. Note that because the token type in turn depends on the column label, it will also provide information to help us determine the column label. For example, if the previous column label is *name*, and the current token is *numeric*, we may think the column label *address* is most likely. However if the token is *all-caps*, we might think the column label is more likely to be *state*. Note that while we used the values *name* and *address* above, we really only have the column labels $L_1, \ldots, L_k$. As mentioned earlier, we may be able to automatically create semantically meaningful names for them using other automatic extraction techniques.

$P(S_i|C_i)$: S_i, whether extract i starts a new record, depends on the column label. It turns out that while later columns may be missing from a record, in all of the domains that we have examined the first column, which usually contains the most salient identifier, such as the Name, is never missing. This allows us to make a very

useful simplification: rather than needing to learn the transition probabilities for the data, it makes sense to assume a deterministic relationship: $P(S_i = \text{true}|C_i = L_1) = 1.0$ and $P(S_i = \text{true}|C_i = L_j, j \neq 1) = 0$. Note that since C_i is not observed, in order to do segmentation we will still need to do probabilistic inference to compute C_i.

$P(R_i|R_{i-1}, D_i, S_i)$: The record number for extract i will depend on the record number of the previous extract, whether or not S_i is the start of a new record, and D_i, the detail pages on which E_i has been observed. In general, this is a deterministic relationship: if S_i is false, then $P(R_i = R_{i-1}) = 1.0$ and if S_i is true, then $P(R_i = R_{i-1} + 1) = 1.0$ However D_i also constrains R_i. R_i must take one of the values of D_i.

The task of record segmentation boils down to finding values for the unobserved R and C variables. As is commonly done in probabilistic models for sequence data, we compute maximum a posteriori (MAP) probability for R and C and use this as our segmentation:

$$\arg\max P(R, C|T, D)$$

Because we are assuming a Markov model, we can write:

$$P(R, C|T, D) = \prod_{i=1}^{n} P(R_i, C_i|D_i, T_i, R_{i-1}, C_{i-1})$$

and using the structure of the graphical model above, this simplifies to:

$$P(R, C|T, D) = \prod_{i=1}^{n} \sum_{S_i} P(R_i|D_i, S_i) P(S_i|C_i) P(C_i|T_i)$$

5.2 Learning the Model

At this point, we could apply a standard off-the-shelf probabilistic inference algorithm to learn the model parameters and to make the appropriate inferences. Unfortunately, our model is so unconstrained, we would have little luck inferring anything useful. We will use two ideas: **bootstrapping** and **structure** to make successful inferences.

5.2.1 Bootstrapping

The key way in which information from detail pages helps us is it gives us a guide to some of the initial R_i assignments. It turns out this little bit of information can be surprisingly informative.

Recall that D_i is the set of detail pages on which extract E_i occurs. This provides invaluable evidence for the record assignment for R_i. We make use of this information by setting:

$$P(R_i = r_i) = \frac{1}{|D_i|}$$

and $P(R_i = r_i) = 0)$ for all $r_i \notin D_i$.

In addition, we make the following initial assumptions for the parameters of the model: $P(T_{i_j} = \text{true}|C_i) = 1/8$, in other words, without observing any of the data, we assume that the probability of a token type occurring, given the column, is 1/8. Note that this does not preclude an extract being of more than one token type. While we begin with this uniform assumption, we will be updating the model parameters as we go along, so the parameters will quickly be updated in accordance with the frequency that we observe the different tokens and column labels.

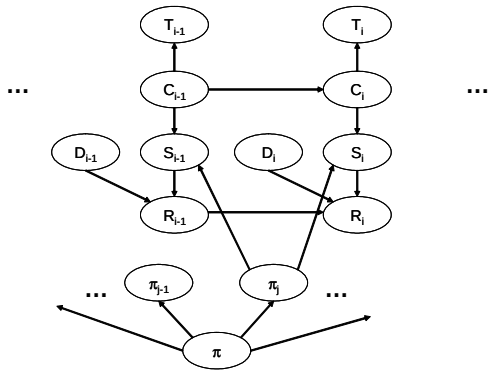

Figure 3: A probabilistic model for record extraction from list and detail pages which includes a record period model π.

We also make use of the D_i to infer values for S_i. If $D_{i-1} \cap D_i = \emptyset$, then $P(S_i = \text{true}) = 1$. As a simple example, if extract i only appears on detail page j and extract $i-1$ only appears on detail page $j-1$, then $S_i = \text{true}$.

5.2.2 Structure

Besides the information from the detail pages, another important piece of information we can make use of is the record length, or the period π of the table; π is the number of columns in the table. Recall, however, that not every record will have all π columns. Instead, for each record r_j, there will be an associated π_j, which is the number of fields in record j. This approach allows for missing fields in a record — a common occurrence in Web data.

One way of allowing this is simply to make use of the S_i. If $S_i = \text{true}$ and $R_i = j$ and $S_{i'}$ is the next record start indicator that is true, then we can simply compute $\pi_j = i - i'$. But this fails to capture the correlations among record lengths. In our example, there are 4 fields possible in a full record, perhaps most often we will have all 4 fields, Name, Address, City, Phone, but another common occurrence is to just have the 3 fields Name, City and Phone. We want to be able to learn this. Now, the columns C_i will be conditioned on the corresponding π_j. We can learn for example that $C_i = City$ is much more likely if $\pi_j = 3$ (and $C_{i-1} = Name$).

The difficulty here is that until we know the correspondence of extract i to r_j; we do not even know which π_j to depend on. This could make inference even more complex. It turns out however, that there are only a small number of possibilities, so this is in fact feasible. Furthermore this more complex model does in fact give us improvements in accuracy. Figure 3 shows a graphical model representation of the updated model.

5.2.3 Implementation

We use EM to implement the probabilistic inference algorithm. We have developed a variant of the forward-backward algorithm that exploits the hierarchical nature of the record segmentation problem. By explicitly modeling the period probabilities, we can constrain the structure of the model and in turn use this structure to guide the inference process.

The basic components of the algorithm are:

1. Compute initial distribution for the global period π using the current values for the S_i. Because of the constraints the detail pages offer, our initial S_i will provide us with some useful information for π.
2. Now, having updated π, we compute the π_k for each record j.
3. For each potential starting point and record length, we update the column start probabilities, $P(C_i|T_i, C_{i-1})$.
4. Next we update $P(S_i|C_i)$.
5. And finally we update $P(R_i|R_{i-1}, D_i, S_i)$

In the end we output the most likely assignment to R and C. This gives us the segmentation for the list page. While in theory, the running time of this algorithm is potentially exponential in the length of the extract sequence, by using the period probabilities to structure the inference we get enormous savings. In practice the algorithm is very efficient and took just a few seconds to run on each of our test cases.

6. RESULTS

We now present results of automatic segmentation of records using the CSP and probabilistic approaches described in the sections above.

6.1 Experimental Setup

The data set consisted of list and detail pages from 12 Web sites in four different information domains, including book sellers (*Amazon*, *BNBooks*), property tax sites (*Buttler*, *Allegheny*, *Lee* counties), white pages (*Superpages*, *Yahoo*, *Canada*411, *SprintCanada*) and corrections (*Ohio*, *Minnesotta*, *Michigan*) domains. From each site, we randomly selected two list pages and manually downloaded the detail pages. In this work, we were not concerned with the task of automatically determining detail pages. In fact, there are often other links from the list page that point to advertisements and other extraneous data. Such data may or may not influence the segmentation results. In future work, we will attempt to automate detail page identification. One approach is to use heuristic rules, such as "follow links in the table" or "follow a link whose text field is More Info." Alternatively, one can download all the pages that are linked on the list pages, and then use a classification algorithm [9, 19] to find a subset that contains the detail pages only. The detail pages, generated from the same template, will look similar to one another and different from advertisement pages, which probably don't share any common structure.

In addition to displaying different data, the pages varied greatly in their presentation and layout. Some used grid-like tables, with or without borders, with easily identifiable columns and rows. Others were more free-form, with a block of the page containing information about an item, followed by another block containing information about another item. Within the block, attributes could appear in separate columns, rows, or in a formatted table. The entries could be numbered or unnumbered. Commercial sites had the greatest complexity and more likely to be free-form than government sites.

The CSP and probabilistic algorithms were exceedingly fast, taking only a few seconds to run in all cases.

6.2 Evaluation

Table 4 shows results of automatic record segmentation for these 12 sites using two different approaches, probabilistic and constraint satisfaction. Both approaches share a common step, the page template finding algorithm. In cases where the template finding algorithm could not find a good page template, we have taken the entire text of the list page for analysis. Only the strings that appeared on both list and detail pages were used in record segmentation. The rest of the table data are assumed to belong to the same record as the last assigned extract. The reason for this is that each row of the table usually starts with the most important attribute, such as the name or ID number. This attribute will almost always appear on the detail page as well.

We manually checked the results of automatic segmentation and classified them as correctly segmented (Cor) and incorrectly segmented (InCor) records, unsegmented records (FN) and non-records (FP). Precision and recall are defined below. We used the F measure to gauge the accuracy of the task.

$$\begin{aligned} P &= Cor/(Cor + InCor + FP) \\ R &= Cor/(Cor + FN) \\ F &= 2PR/(P + R) \end{aligned}$$

We calculated $P = 0.74$, $R = 0.99$ and $F = 0.85$ for the probabilistic approach and $P = 0.85$, $R = 0.84$ and $F = 0.84$ for the CSP approach. This is an exceedingly good performance for automatic algorithms. Using heuristics, as described below, we can further improve on the results.

6.3 Discussion

Each approach has its benefits and drawbacks, which make them suitable in different situations. The CSP approach is very reliable on clean data, but it is sensitive to errors and inconsistencies in the data source. One such source of data inconsistency was observed on the Michigan corrections site, where an attribute had one value on the list pages and another value on the detail pages. This by itself is not a problem; however, the list page string appeared on one detail page in an unrelated context. The CSP algorithm could not find an assignment of the variables that satisfied all the constraints. The probabilistic approach, on the other hand, tolerates such inconsistencies and is more expressive than the CSP representation. Its expressiveness gives us the power to potentially assign extracts to individual attributes, and, when combined with a system that automatically extracts column labels [2] from tables, reconstruct the relational database behind the Web site. Both techniques (or a

	Probabilistic				CSP				
Wrapper	**Cor**	**InC**	**FN**	**FP**	**Cor**	**InC**	**FN**	**FP**	**notes**
Amazon	4	6	0	1	0	0	10	0	a, b
Books	2	5	3	4	0	0	10	0	
BN	5	5	0	0	2	0	8	0	a, b, c, d
Books	5	5	0	0	0	0	10	0	
Allegheny	20	0	0	0	20	0	0	0	
County	16	4	0	0	20	0	0	0	
Butler	15	0	0	0	15	0	0	0	
County	12	0	0	0	12	0	0	0	
Lee	16	0	0	0	16	0	0	0	
County	5	0	0	0	5	0	0	0	
Michigan	7	0	0	0	4	3	0	0	
Corrections	12	4	0	0	2	8	6	0	c, d
Minnesota	11	0	0	0	4	7	0	0	a, b, c, d
Corrections	17	2	0	0	8	9	0	2	
Ohio	8	2	0	0	10	0	0	0	
Corrections	10	0	0	0	10	0	0	0	
Canada	18	7	0	0	25	0	0	0	
411	1	4	0	0	1	4	0	0	c, d
Sprint	17	3	0	0	20	0	0	0	
Canada	8	12	0	0	20	0	0	0	
Yahoo	0	10	0	0	5	5	0	0	a, b, c, d
People	10	0	0	0	10	0	0	0	b
Super	3	0	0	0	3	0	0	0	a, b
Pages	9	6	0	0	15	0	0	0	
Precision	0.74				0.85				
Recall	0.99				0.84				
F	0.85				0.84				

Notes
a. Page template problem; b. Entire page used; c. No solution found;
d. Relax constraints

Table 4: Results of automatic record segmentation of tables in Web pages using the probabilistic and CSP approaches

combination of the two) are likely to be required for robust and reliable large-scale information extraction. We stress that the approaches are novel in that they are domain independent, unsupervised, and rely on the content of Web pages rather than their layout.

The page template finding algorithm performed poorly on five of the 12 sites: *Amazon, BnBooks*, *Minnesota Corrections*, *Yahoo People* and *Superpages*. In the first three sites, the entries were numbered. Thus, sequences such as "1.", will be found on every page. If the tables are of different lengths, the shortest table will limit what is to be considered a page template, and the remainder of data on the longer tables will be extracted. When we encountered a problem with the page template algorithm, we use the entire page as the table slot — in other words, we used the entire content of the list page for matching with the detail page. At times, using the entire list page led to problems (*Amazon*, first *Yahoo People* list page), as many strings that were not part of the table found matches on detail pages, although in some cases this approach performed quite well (second list in *Yahoo People*, *Superpages*). There are a number of ways to circumvent this problem which were not explored in this paper. One method is to simply follow the "Next" link, and download the next page of results. The entry numbers of the next page will be different from others in the sample. Another approach is to build a heuristic into the page template algorithm that finds enumerated entries. We will try this approach in our future work.

The CSP approach performed extremely well on clean data; however, it was prone to fail when there were errors and inconsistencies in the data underlying the Web sites. For example, on one *Canada411* page, one of the records had the town attribute missing on the detail page but not on the list page. Since the town name was the same as in other records, it was found on every detail page but the one corresponding to the record in question. As a result, WSAT(OIP) could not find a solution satisfying all constraints. Relaxing constraints by replacing equalities with inequalities produced a solution, but it was a partial solution, because not every extract was assigned to a record. Another common data inconsistency that caused WSAT(OIP) to fail was when the attribute had different values on list and detail pages. For example, on the *Amazon* site, a long list of authors was abbreviated as "FirstName LastName, et al" on list pages, while the names appeared in full on the detail page. On the *Minnesota Corrections* site, there was a case mismatch between attribute values on list and detail pages. On the *Michigan Corrections* site, status of an paroled inmate was listed as "Parole" on list pages and "Parolee" on detail pages. Unfortunately, the string "Parole" appeared on another page in a completely different context. As a result, all constraints could not be satisfied. In such cases we relaxed the constraints, for example, by requiring that an extract appear on at most one detail page. WSAT(OIP) was able to find solutions for the relaxed constraint problem, but the solution corresponded to a partial assignment. If we excluded from consideration those Web pages for which the CSP algorithm could not find a solu-

tion, performance metrics on the remaining 17 pages were $P = 0.99$, $R = 0.92$ and $F = 0.95$. This performance is comparable to that obtained with hand-crafted heuristic or domain-specific rules. [3, 5] The probabilistic approach was less sensitive to data inconsistencies, but was slightly less accurate overall. On the same 17 pages as above, its performance was $P = 0.78$, $R = 1.0$ and $F = 0.88$.

Except for the two book sites, we were able to correctly segment at least one table using either method. The tables on the book site pages presented challenges. The entries in these lists were numbered. As a result, the page template algorithm did not work, and we had to use the text of the entire list page. Unfortunately, many of the strings in the list page, that were not part of the list, appeared in detail pages, confounding our algorithms. For the *Amazon* site the problem was further compounded by the fact that we downloaded the pages manually. The site offers the user a useful feature of displaying her browsing history on the pages. This led to title of books from previously downloaded detail pages to appear on unrelated pages, completely derailing the CSP algorithm. Even relaxing constraints to find a partial assignment did not produce good results for these two sites. These observations do not apply to the performance of the algorithm, only the data collection and preparation steps. Adding domain-specific data collection techniques should improve the final segmentation results.

The probabilistic approach allows us to assign extracts to attributes, not only records. This remains an open problem for future research. It may also be possible to obtain the attribute assignment in the CSP approach, by using the observation that different values of the same attribute should be similar in content, *e.g.*, start with the same token type. We may be able to express this observation as a set of constraints.

As discussed earlier in this paper, the RoadRunner system uses an elegant approach to automatically extract data from data-rich Web sites. RoadRunner assumes that the results pages from a site were generated by a union-free grammar and induces this grammar from example list pages. This approach, however, fails for Web sites that use alternate formatting instructions for the same field. Such alternate instructions are syntactically equivalent to disjunctions, which are disallowed by union-free grammars. Consider Superpages list page in Fig. 1. If an address field is missing, the text "street address not available" is displayed in gray font; otherwise, an address is displayed in black. In each alternative, different HTML tags are used to format the address field. There is still an underlying grammar that gave rise to this Superpages page, but this grammar admits unions. Inferring such grammars from examples is an even more computationally complex task than inferring union-free grammars. Our approach, on the other hand, is more computationally efficient, because rather than using a page's layout, it relies on the content of the page, which is often much less than the layout. Both of our methods handle the Superpages site very effectively, as shown in the results.

7. CONCLUSION

There are multiple ways to represent and leverage the additional information contained in the structure of Web sites. In this work we investigated two of them: 1) a logic-based approach in which we encode the information provided by detail pages as constraints and solve them to obtain the record segmentation, and 2) a probabilistic inference approach in which we represent the observations and structure of the table as a probabilistic model and use an inference algorithm to find appropriate segmentation. Both approaches have widely used, efficient algorithms for solving problems. Each has its benefits and drawbacks, that make them preferable in different situations. The constraint-satisfaction approach is very reliable on clean data, but it is sensitive to errors and inconsistencies in the data source. The probabilistic approach on the other hand, tolerates inconsistencies and is more expressive than the constraint-based approach, and, beyond record segmentation, it can perform record extraction. Both techniques (or a combination of the two) are likely to be required for large-scale robust and reliable information extraction.

8. ACKNOWLEDGEMENTS

We are grateful to J. P. Walser for generously licensing the WSAT(OIP) code and to Alejandro Bugacov for useful discussions about CSP problem encoding and solvers. We wish to thank reviewers for suggesting relevant references.

This material is based upon work supported in part by the National Science Foundation under Award No. IIS-0324955, in part by the Air Force Office of Scientific Research under grant number F49620-01-1-0053 and in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under contract/agreement numbers F30602-01-C-0197. Lise Getoor was partially supported by the National Science Foundation under Award No. 0308030 and by the Advanced Research and Development Activity (ARDA) under Award Number NMA401-02-1-2018.

9. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of data*, 2003.

[2] L. Arlotta, V. Crescenzi, G. Mecca, and P. Marialdo. Automatic annotation of data extracted from large web sites. In *Proceedings of the Sixth International Workshop on Web and Databases (WebDB03)*, 2003.

[3] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records full text. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001.

[4] C. H. Chang, and S. C. Lui. IEPAD: Information Extraction based on Pattern Discovery. In *10th International World Wide Web Conference (WWW10), Hong Kong*, 2001.

[5] H. Chen, S. Tsai, and J. Tsai. Mining tables from large scale html texts. In *18th International Conference on Computational Linguistics (COLING)*, 2000.

[6] W. W. Cohen, M. Hurst, and L. S. Jensen. A Flexible Learning System for Wrapping Tables and Lists in HTML Documents. In *11th International World Wide Web Conference (WWW10), Honolulu, Hawaii*, 2002.

[7] V. Crescenzi, G. Mecca, and P. Merialdo. Automatic web information extraction in the ROADRUNNER system. In *Proceedings of the International Workshop on Data Semantics in Web Information Systems (DASWIS-2001)*, 2001.

[8] V. Crescenzi, G. Mecca, and P. Merialdo. ROADRUNNER: Towards automatic data extraction from large web sites. In *Proceedings of the 27th Conference on Very Large Databases (VLDB)*, Rome, Italy, 2001.

[9] C. Gazen. Thesis proposal, Carnegie Mellon University.

[10] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 8, pages 472–478. MIT Press, 1995.

[11] M. Hurst. Layout and language: Challenges for table understanding on the web. In *In Web Document Analysis, Proceedings of the 1st International Workshop on Web Document Analysis*, 2001.

[12] M. Hurst and S. Douglas. Layout and language: Preliminary investigations in recognizing the structure of tables. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 1997.

[13] Y. Jiang. *Record-Boundary Discovery In Web Documents*. PhD thesis, BYU, Utah, 1998.

[14] N. Kushmerick and B. Thoma. *Intelligent Information Agents R&D in Europe: An AgentLink perspective*, chapter Adaptive information extraction: Core technologies for information agents. Springer, 2002.

[15] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[16] K. Lerman and S. Minton. Learning the Common Structure of Data. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-2000)*, Menlo Park, 2000. AAAI Press.

[17] K. Lerman, C. A. Knoblock, and S. Minton. Automatic data extraction from lists and tables in web sources. In *Proceedings of the workshop on Advances in Text Extraction and Mining (IJCAI-2001)*, Menlo Park, 2001. AAAI Press.

[18] K. Lerman, S. Minton, and C. Knoblock. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, 18:149–181, 2003.

[19] K. Lerman, C. Gazen, S. Minton, and C. A. Knoblock,. Populating the Semantic Web. *submitted to the workshop on Advances in Text Extraction and Mining (ATEM-2004)*, 2004.

[20] K. Murphy. Dynamic bayesian networks: Representation, inference and learning. PhD Thesis, UC Berkeley, 2002.

[21] I. Muslea, S. Minton, and C. Knoblock. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, pages 435–442. Morgan Kaufmann, San Francisco, CA, 2002.

[22] H. T. Ng, H. Y. Lim, and J. L. T. Koo. Learning to recognize tables in free text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL99)*, 1999.

[23] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table extraction using conditional random fields. In *Proceedings of the ACM SIGIR)*, 2003.

[24] P. Pyreddy and W. B. Croft. Tintin: A system for retrieval in text tables. In *Proceedings of 2nd International Conference on Digital Libraries*, 1997.

[25] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in Speech Recognition*.

[26] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the Twenty-seventh International Conference on Very Large Databases*, 2001.

[27] J. P. Walser. Wsat(oip) package.

[28] J. P. Walser. *Integer Optimization by Local Search: A Domain Independent Approach*, volume 1637 of *LNCS*. Springer, New York, 1999.

[29] Y. Wang and J. Hu. Detecting tables in html documents. In *Fifth IAPR International Workshop on Document Analysis Systems, Princeton, New Jersey*, August 2002.

[30] Y. Wang and J. Hu. A machine learning based approach for table detection on the web. In *In The Eleventh International World Web Conference, Honolulu, Hawaii, USA, May 2002.*, 2002.

[31] Y. Wang, I. T. Phillips, and R. Haralick. Table detection via probability optimization. In *Fifth IAPR International Workshop on Document Analysis Systems, Princeton, New Jersey*, August 2002.

[32] M. Yoshida, K. Torisawa, and J. Tsujii. A method to integrate tables of the world wide web. In *in Proceedings of the International Workshop on Web Document Analysis (WDA 2001), Seattle, U.S.*, September 2001.