Simultaneous Record Detection and Attribute Labeling in Web Data Extraction

Jun Zhu[*][†] Zaiqing Nie[‡] Ji-Rong Wen[‡] Bo Zhang[†] Wei-Ying Ma[‡]

[†]Department of Computer Science & Technology
Tsinghua University
Beijing, China

[‡]Web Search & Mining Group
Microsoft Research Asia
Beijing, China

[†]{jun-zhu, dcszb}@mails.tsinghua.edu.cn [‡]{znie, jrwen, wyma}@microsoft.com

ABSTRACT

Recent work has shown the feasibility and promise of template-independent Web data extraction. However, existing approaches use decoupled strategies – attempting to do data record detection and attribute labeling in two separate phases. In this paper, we show that separately extracting data records and attributes is highly ineffective and propose a probabilistic model to perform these two tasks simultaneously. In our approach, record detection can benefit from the availability of semantics required in attribute labeling and, at the same time, the accuracy of attribute labeling can be improved when data records are labeled in a collective manner. The proposed model is called Hierarchical Conditional Random Fields. It can efficiently integrate all useful features by learning their importance, and it can also incorporate hierarchical interactions which are very important for Web data extraction. We empirically compare the proposed model with existing decoupled approaches for product information extraction, and the results show significant improvements in both record detection and attribute labeling.

Categories and Subject Descriptors: I.5.1 [Pattern Recognition]: Models - Statistical

General Terms: Algorithms, Experimentation

Keywords: Data record detection, Attribute labeling, Web page segmentation, Conditional Random Fields, Hierarchical Conditional Random Fields.

1. INTRODUCTION

The World Wide Web is a vast and rapidly growing repository of information. There are various kinds of objects, such as products, people, conferences, etc., embedded in both statically and dynamically generated Web pages. Recent work has shown that using template-independent approaches to extracting meta-data for the same type of real-world objects is feasible and promising. However, existing approaches use highly ineffective decoupled strategies - attempting to do data record detection and attribute labeling in two separate phases. This paper studies how to extend existing Web data extraction methods to achieve the mutual enhancement of record detection and attribute labeling.

*The work is done when the author is visiting Microsoft Research Asia.

1.1 Motivating Example

We begin by illustrating the problem with an example, drawn from an actual application of product information extraction. The goal of the application is to extract meta-data about real-world products from every product page on the Web. Specifically, for each crawled Web page, we first use a classifier to decide whether it is a product page and then extract the *name*, *image*, *price* and *description* of each product from detected product pages.

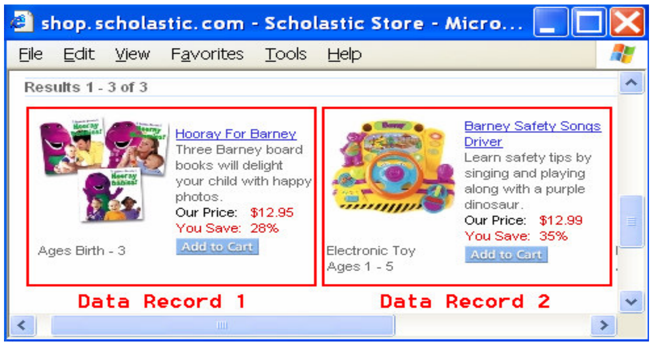
Our statistical study on 51K randomly crawled Web pages shows that about 12.6 percent are product pages. That is, there are about 1 billion product pages within a search index containing 9 billion crawled Web pages. If all of these pages or just half of them are correctly extracted, we will have a huge collection of meta-data about real-world products that could be used for further knowledge discovery and data management tasks such as comparison shopping and user intention detection.

However, how to extract product information from Web pages generated by many (maybe tens of thousands of) different templates is non-trivial. One possible solution is that we first distinguish Web pages generated by different templates, and then build an extractor for each template. We say that this type of solution is *template-dependent*. However, accurately identifying Web pages for each template is not a trivial task because even Web pages from the same website may be generated by dozens of templates. Even if we can distinguish Web pages, template-dependent methods are still impractical because the learning and maintenance of so many different extractors for different templates will require substantial efforts.

Fortunately, recent work in [18][31][27][33] has shown the feasibility and promise of *template-independent* meta-data extraction for the same type of objects, and it is possible to simply combine existing techniques to build a template-independent meta-data extractor for product pages. Two specific types of Web pages are treated differently by the existing extraction techniques: *list pages* and *detail pages*[1].

- **List pages** are web pages containing several structured data records (See Figure 1(a) for an example). We can build an

[1] Our empirical study shows that about 35% of product pages are list pages and the rest are detail pages.


shop.scholastic.com - Scholastic Store - Micro...

File Edit View Favorites Tools Help

Results 1 - 3 of 3

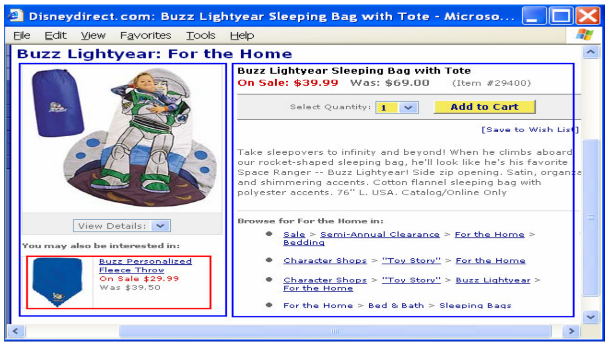
Hooray For Barney
Three Barney board books will delight your child with happy photos.
Our Price: $12.95
You Save: 28%
Add to Cart
Ages Birth - 3

Barney Safety Songs Driver
Learn safety tips by singing and playing along with a purple dinosaur.
Our Price: $12.99
You Save: 35%
Add to Cart
Electronic Toy
Ages 1 - 5

Data Record 1

Data Record 2

Figure 1(a). A sample Web page with two similar data records which are contained in red boxes.


Disneydirect.com: Buzz Lightyear Sleeping Bag with Tote - Microso...

File Edit View Favorites Tools Help

Buzz Lightyear: For the Home

Buzz Lightyear Sleeping Bag with Tote
On Sale: $39.99 Was: $69.00 (Item #29400)

Select Quantity: 1 Add to Cart

[Save to Wish List]

Take sleepovers to infinity and beyond! When he climbs aboard our rocket-shaped sleeping bag, he'll look like he's his favorite Space Ranger -- Buzz Lightyear! Side zip opening. Satin, organza and shimmering accents. Cotton flannel sleeping bag with polyester accents. 76" L. USA. Catalog/Online Only

View Details:

You may also be interested in:

Buzz Personalized Fleece Throw
On Sale $29.99
Was $39.50

Browse for For the Home in:

- Sale > Semi-Annual Clearance > For the Home > Bedding
- Character Shops > "Toy Story" > For the Home
- Character Shops > "Toy Story" > Buzz Lightyear > For the Home
- For the Home > Bed & Bath > Sleeping Bags

Figure 1(b). A detail page containing detailed formation about a product.

object meta-data extractor for list pages by first using the techniques in [31] or [18] to detect the data records and then use the techniques in [33] to label the data elements within the detected records. Specifically, the techniques in [31][18] segment the data records in a list page (this task is called *record detection*) and also further extract their data elements. However, there is no semantic label about these extracted data elements. That is, we do not know whether an extracted item is the name, the description or the price of a product. The work in [33], which is complementary to [31][18], provides a method to assign semantic labels to the data elements of each extracted data record (this task is called *attribute labeling*).

- **Detail Pages** are Web pages containing only detailed information about a single object (See Figure 1(b) for an example). We can build an object meta-data extractor for detail pages by first using the techniques in [27] to identify the main data block of a detail page, and then use the techniques in [33] to do attribute labeling for the main block.

However, it is *highly ineffective to use decoupled strategies* – attempting to do data record detection and attribute labeling in two separate phases. This is because:

Error Propagation: Since record detection and attribute labeling are two separate phases, the errors in record detection will be propagated to attribute labeling. Thus, the overall performance is limited and upper-bounded by that of record detection. Suppose record detection and attribute labeling have precisions 0.8 and 0.9 respectively, then the overall precision will be no more than 0.8. If they also perform independently, the precision will be 0.72.

Lack of Semantics in Record Detection: Human readers always take into account the semantics of the text to understand Web pages. For instance, in Figure 1(a), when claiming a block is a data record, we use the evidence that it contains a product's name, image, price and description. Thus, a more effective record detection algorithm should take into account the semantics of the text, but existing methods [18][31][27] do not consider that.

Lack of Mutual Interactions in Attribute Labeling: The data records in the same page are related. They always share a common template and the elements at the same position of different records always have similar features and semantics. For example, in Figure 1(a) the button "Add to Cart" appears in the same position in both records and the element on the left-top of each record is an image. So, if we label the elements of the records within the same page together in a collective manner, it is easier for us to detect that the repeated elements "Add to Cart" are less informative and more likely to be noise. However, existing method [33] fails to consider that because the data records are independently labeled.

First-Order Markov Assumption: For Web pages and especially for detail pages, long distance dependencies always exist between different attribute elements. This is because there are always many irrelevant elements (i.e. noise) appearing between the attributes of an object. For example, in Figure 1(b) there is substantial noise, such as "Add to Cart" and "Select Quantity" between the price and description. However, traditional CRFs [17][33] cannot incorporate these long distance dependencies because of its first-order Markov assumption (i.e. only the dependencies of neighboring nodes are considered and represented).

1.2 Our Solution

In this paper, we propose a novel graphical model called Hierarchical Conditional Random Field (HCRF) model to jointly optimize record detection and attribute labeling.

By using the vision-based page segmentation (VIPS) technology [4], which makes use of page layout features such as font, color, and size to construct a *vision-tree* for a Web page, we can get a better representation of a page compared with the commonly used tag-tree. Given a vision-tree, record detection can be considered as the task of locating the minimum set of elements that contain the content of a record. In this way, both record detection and attribute labeling become the task of assigning labels to the nodes on a vision-tree, so they can be integrated in one probabilistic model as in this paper. In contrast to existing decoupled strategies that perform record detection and attribute labeling as two separate phases, our approach leverages the labeling results of attribute labeling for record detection, and at the same time benefits from the incorporation of some global features based on tree-alignment for attribute labeling. As a conditional model [17], HCRF can efficiently incorporate any useful feature for Web data extraction. By incorporating hierarchical interactions, HCRF could incorporate long distance dependencies and achieve promising results on detail Web pages.

To the best of our knowledge, our approach is the first to simultaneously conduct record detection and attribute labeling for both list and detail pages. Specifically, we make the following contributions.

- A mutually beneficial integration of data record detection and attribute labeling. The integrated approach can extract data from both list and detail pages.
- A novel graphical model called Hierarchical Conditional Random Fields. The model relaxes the first-order Markov assumption made in traditional CRFs [17][33] by incorporating hierarchical dependencies, and provides a way to efficiently incorporate any useful feature for Web data extraction.
- An empirical study of our approach on the task of product information extraction.

The rest of the paper is organized as follows. Section 2 formally defines our problem. Section 3 discusses the proposed Hierarchical Conditional Random Fields. Section 4 defines label spaces for the model. Section 5 presents the features we currently used. Section 6 presents our evaluation results and discussions. Section 7 discusses related work and section 8 brings this paper to a conclusion.

2. PROBLEM DEFINITION

2.1 Data Representation

For Web data extraction, the first thing is to find a good representation format for Web pages. Good representation can make the extraction task easier and improve extraction accuracy. In most previous work, tag-tree, which is a natural representation of the tag structure, is commonly used to represent a Web page. However, as [4] pointed out, tag-trees tend to reveal presentation structure rather than content structure, and are often not accurate enough to discriminate different semantic portions in a Web page. Moreover, since authors have different styles to compose Web pages, tag-trees are often complex and diverse.

To overcome these difficulties, [4] proposed a vision-based page segmentation (VIPS) approach. VIPS makes use of page layout features such as font, color, and size to construct a *vision-tree* for a page. It first extracts all suitable nodes from the tag-tree, and then finds the separators between these nodes. Here, separators denote the horizontal or vertical lines in a Web page that visually do not cross any node. Based on these separators, the vision-tree of the Web page is constructed. Each node on this tree represents a data region in the Web page, which is called a *block*. The root block represents the whole page. Each inner block is the aggregation of all its child blocks. All leaf blocks are atomic units (*i.e.* elements) and form a flat segmentation of the Web page. Since vision-tree can effectively keep related content together while separating semantically different blocks from one another, we use it as our data representation format. Figure 2 is a vision-tree for the page in Figure 1(a), where we use rectangles to denote inner blocks and use ellipses to denote leaf blocks (or elements). Due to space limitations, the blocks denoted by dotted rectangles are not fully expanded.

2.2 Record Detection and Attribute Labeling

Based on the definition of vision-tree, we can now formally define the concepts of record detection and attribute labeling.

Definition 2.1 (Record detection)*: Given a vision-tree, record detection is the task of locating the minimum set of blocks that contain the content of a record. For a list page containing multiple records, all the records need to be identified. In other words, record detection is equal to assigning "Data Record" labels to the blocks in the vision-tree.*

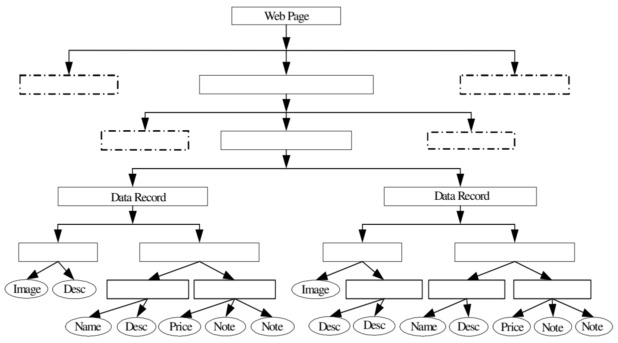


Figure 2. The vision-tree of the page in Figure 1(a).

For instance, for the vision-tree in Figure 2, two blocks are detected as data records.

Definition 2.2 (Attribute labeling)*: For each identified record, attribute labeling is the process of assigning attribute labels to the leaf blocks (or elements) within the record.*

In Figure 2, for the two records, attribute labels Name, Image, Price, Description (Desc) and Note are assigned to different parts within the records. We use Note to denote uninteresting elements or Web noise.

It is obvious that, by sequentially combining record detection and attribute labeling algorithms, a complete algorithm can be constructed to extract records and the attributes. However, as explained in the introduction, this decoupled strategy will be highly ineffective. Therefore, in this paper, we explore an integrated approach that can simultaneously conduct record extraction and attribute labeling and achieve a joint optimization of both tasks.

2.3 Problem Definition

Based on the above definitions, both record detection and attribute labeling are the processes of assigning labels to blocks of the vision-tree for a page, although the labels they use are different. Therefore, we can use a uniform probabilistic model to deal with both tasks. Formally, we define the Web data extraction problem as:

Definition 2.3 (Joint optimization of record extraction and attribute labeling)*: Give a vision-tree of a page, let $\mathrm{x} = \{\mathrm{x}_0, \mathrm{x}_1, \cdots, \mathrm{x}_N\}$ be the features of all the blocks and each component x_i is a feature vector of one block, and let $\mathrm{y} = \{\mathrm{y}_0, \mathrm{y}_1, \cdots, \mathrm{y}_N\}$ be one possible label assignment of the corresponding blocks. The goal of Web data extraction is to compute maximum a posteriori (MAP) probability of y and extract data from this assignment y^*:*

$$\mathrm{y}^* = \arg\max p(\mathrm{y} \mid \mathrm{x})$$

Based on the above definition, the main difficulty is the calculation of the conditional probability $p(\mathrm{y} \mid \mathrm{x})$. In this paper, we introduce a Hierarchical Conditional Random Fields algorithm to compute it, which will be described in detail in the next section.

3. HIERARCHICAL CONDITIONAL RANDOM FIELDS

In this section, we first introduce some basic concepts of Conditional Random Fields and then describe our proposed model for integrated Web data extraction, including how to learn the parameters and how to perform extraction.

3.1 Preliminaries

Conditional Random Fields (CRFs) [17] are Markov random fields that are globally conditioned on observations. Let $G=(V,E)$ be an undirected model over a set of random variables Y and X. X are variables over the observations to be labeled and Y are variables over the corresponding labels. The random variables Y could have non-trivial structure, such as linear-chain in [17] and 2D grid in [33]. Although all the components of Y are usually assumed to range over a single label space $\mathcal{Y}$, this assumption is not an innate character of CRFs and we will see that in our proposed model there are two label spaces for two different types of variables. The conditional distribution of the labels y (an instance of Y) given the observations x (an instance of X) has the form,

$$p(\mathrm{y} \mid \mathrm{x}) = \frac{1}{Z(\mathrm{x})} \prod_{c \in C} \varphi_c(\mathrm{y}|_c, \mathrm{x}) \quad \cdots\cdots \ (1)$$

where C is the set of cliques in G; $\mathrm{y}|_c$ are the components of y associated with the clique c; φ_c is a potential function defined on $\mathrm{y}|_c$ and takes non-negative real values; $Z(\mathrm{x})$ is the normalization factor or partition function, and has the form,

$$Z(\mathrm{x}) = \sum_{\mathrm{y}} \prod_{c \in C} \varphi_c(\mathrm{y}|_c, \mathrm{x})$$

The potential functions are expressed in terms of feature functions $f_k(\mathrm{y}|_c, \mathrm{x})$ and their weights λ_k:

$$\varphi_c(\mathrm{y}|_c, \mathrm{x}) = \exp\left(\sum_k \lambda_k f_k(\mathrm{y}|_c, \mathrm{x}) \right)$$

3.2 Hierarchical CRFs

Based on the hierarchical representation of the data, a Hierarchical Conditional Random Field (HCRF) model can be easily constructed. For the page in Figure 1(a), the model is shown in Figure 3, where we also use rectangles to denote inner nodes and use ellipses to denote leaf nodes. The dotted rectangles are for the blocks that are not fully expanded. Each node on the graph is associated with a random variable Y_i. We currently model the interactions of sibling variables via a linear-chain, although more complex structure such as two-dimensional grid can be used. The observations which are globally conditioned on are omitted from this graph for simplicity. Here, we assume that every inner node contains at least two children. Otherwise, we replace the parent with its single child. Notice that this assumption has no affect on the performance because the parent is identical to its child in this case. We made this assumption just for easy explanation and implementation.

The cliques of the graph in Figure 3 are its vertices, edges and triangles. So, the conditional probability in formula (1) can be concretely expressed as,

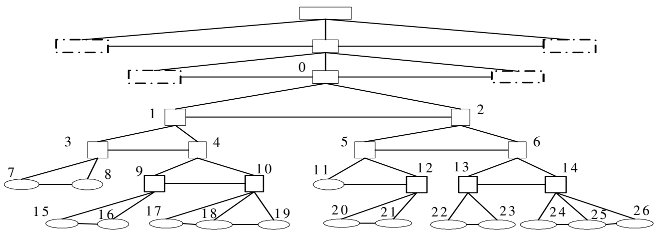


Figure 3. The model's graph of the page in Figure 1(a).

$$p(\mathrm{y} \mid \mathrm{x}) = \frac{1}{Z(\mathrm{x})} \exp\left(\begin{array}{l} \sum_{v,k} \mu_k g_k(\mathrm{y}|_v, \mathrm{x}) + \sum_{e,k} \lambda_k f_k(\mathrm{y}|_e, \mathrm{x}) \\ + \sum_{t,k} \gamma_k h_k(\mathrm{y}|_t, \mathrm{x}) \end{array} \right) \quad \ldots \ (2)$$

where g_k, f_k and h_k are feature functions defined on three types of cliques (i.e. vertex, edge and triangle) respectively; μ_k, λ_k and γ_k are the corresponding weights; $v \in V$, $e \in E$ and t is a triangle. Although the feature functions can take real values, here we assume they are Boolean, that is, *true* if the feature matches and otherwise *false*. Some examples could be found in section 5.

3.3 Learning the Parameters

Given the training data $D = \{(\mathrm{y}^i, \mathrm{x}^i)\}_{i=1}^N$ with an empirical distribution $\tilde{p}(\mathrm{x},\mathrm{y})$, the parameter learning problem is to find a set of parameters $\Theta = \{\mu_1, \mu_2, \cdots; \lambda_1, \lambda_2, \cdots; \gamma_1, \gamma_2, \cdots\}$ that optimize the log-likelihood function:

$$L(\Theta) = \sum_i \tilde{p}(\mathrm{x}^i, \mathrm{y}^i) \log p(\mathrm{y}^i \mid \mathrm{x}^i, \Theta)$$

To avoid over-fitting, we also use the spherical Gaussian prior with mean $\mu = 0$ and variance matrix $\Sigma = \sigma^2 I$ to penalize the log-likelihood. The concave function can be optimized by the techniques used in other models [17]. We use the gradient-based L-BFGS [21] for its outstanding performance over other algorithms [22]. Each element of the gradient vector is given by,

$$\frac{\partial L(\Theta)}{\partial \lambda_k} = E_{\tilde{p}(\mathrm{x},\mathrm{y})}[f_k] - E_{p(\mathrm{y}|\mathrm{x},\Theta)}[f_k] - \frac{\lambda_k}{\sigma^2}$$

where $E_{\tilde{p}(\mathrm{x},\mathrm{y})}[f_k]$ is the expectation with respect to the empirical distribution, and it can be easily calculated and once for all; $E_{p(\mathrm{y}|\mathrm{x},\Theta)}[f_k]$ is the expectation with respect to the model distribution. For example, the expectation of f_k is:

$$E_{p(\mathrm{y}|\mathrm{x},\Theta)}[f_k] = \sum_i \tilde{p}(\mathrm{x}^i) \sum_{e \in E} \sum_{\mathrm{y}|_e} p(\mathrm{y}|_e \mid \mathrm{x}^i) f_k(\mathrm{y}|_e, \mathrm{x}^i)$$

where $\mathrm{y}|_e$ is any possible label assignment of the variables on the edge e.

Thus, the main computation is to compute the marginal probabilities $p(\mathrm{y}|_e \mid \mathrm{x}^i)$ needed when computing the gradients at every iteration. As the graph in Figure 3 is a chordal graph, we can efficiently do inference by using the junction tree algorithm [8]. For an undirected graph, this algorithm consists of three steps: junction tree construction, initialization, and belief propagation.

It is straightforward to construct a junction tree for a chordal graph. Figure 4 is the junction tree of the sub-graph starting from the node 0 in Figure 3, and we will denote it by T in the rest of

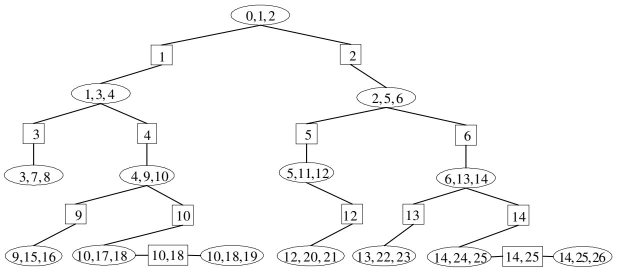


Figure 4. The junction tree of the sub-graph starting from 0 in Figure 3.

the paper. In Figure 4, we use ellipses to denote clique nodes and use rectangles to denote separators. All the variables are denoted by their subscripts. Note that all the clique nodes on the junction tree have size 3 because the maximum cliques in the original graph are triangles.

Once the junction tree has been constructed, the initialization can be finished with first initializing all the potentials on T to have value unity, and then multiplying the potential of a vertex, an edge or a triangle into the potential of one of the clique nodes of T which cover the variables of the vertex, the edge or the triangle. The potential of a vertex v, an edge e and a triangle t is computed as follows respectively,

$$\varphi_v(\mathrm{y}|_v, \mathrm{x}) = \exp\left(\sum_k \mu_k g_k(\mathrm{y}|_v, \mathrm{x})\right)$$

$$\varphi_e(\mathrm{y}|_e, \mathrm{x}) = \exp\left(\sum_k \lambda_k f_k(\mathrm{y}|_e, \mathrm{x})\right), \quad \varphi_t(\mathrm{y}|_t, \mathrm{x}) = \exp\left(\sum_k \gamma_k h_k(\mathrm{y}|_t, \mathrm{x})\right)$$

Finally, we use the two-phase schedule algorithm [14] to schedule belief propagation on T. This algorithm requires the selection of an arbitrary clique node to be the root, and then is followed by a *collection* phase and a *distribution* phase. Once these two phases are finished, the potentials on all the cliques and separators are marginal potentials. The marginal probabilities can be directly computed with the marginal potentials by normalization.

3.4 Finding the Most Likely Assignment

As we have described in section 2.3, we are interested in finding the most probable configuration of Y. This can be done using the same junction tree algorithm [8] just with a simple modification of the two-phase schedule algorithm [14] by merely replacing the summation, which is an operator needed in the two-phase schedule algorithm, by maximization everywhere. After the modified algorithm has finished, the most likely assignment for each variable can be found from the potentials of any clique that contains the variable. The junction tree's running intersection property guarantees that the most likely states of a variable found on different cliques are the same. We refer the readers to [8] for more details about the junction tree algorithm.

4. LABEL SPACE DEFINITION

When applying HCRF models for Web data extraction, we must determine the label spaces for variables Y. We now explain how to define these labels and also give an example of the label spaces which are used in our product information extraction.

When defining the label spaces, we need to distinguish between the variables at leaf nodes and the variables at inner nodes. For variables at leaf nodes, we are interested in deciding whether a leaf block is an attribute value of the object we want to extract. However, for variables at inner nodes, our interest shifts to the understanding of whether an inner block is a data record. So, we have two types of label spaces – *leaf label space* for variables at leaf nodes and *inner label space* for variables at inner nodes.

Table 1. Label spaces for product information extraction

Type	Label	Semantic Meaning
Leaf Label Space	Name	The name of a product
	Image	The image of a product
	Price	The price of a product
	Desc	The description of a product
	Note	The non-interesting elements or noise
Inner Label Space	Con_Image	Contain product's image
	Con_Name	Contain product's name
	Con_Price	Contain product's price
	Con_Desc	Contain product's description
	Con_ImgNam	Contain product's image and name
	Con_NamPrc	Contain product's name and price
	Con_ImgPrc	Contain product's image and price
	Page Head	The head part of a Web page
	Page Tail	The tail part of a Web page
	Nav Bar	The navigation bar of a Web page
	Info Block	Contain one or more data records and some additional information
	Data Region	Contain only similar data records
	Data Record	Contain all the attributes (*i.e.* name, price, image and description) if exist
	Note Block	Contain no interesting information and are also not meaningful parts of a Web page

The leaf label space consists of all the attribute names of the object we want to extract. For example, in product information extraction, we want to extract a product's name, image, price and description, so the leaf label space consists of *Name*, *Image*, *Price*, *Desc*, and *Note*. The label *Note* is for the data we are not interested in (i.e. noise). All the labels for product information extraction are listed in Table 1 with their semantic meanings.

The inner label space can be partitioned into an *object-type independent* part and an *object-type dependent* part. We explain how to define these two parts in turn:

Object-type Independent Labels: Since we want to extract data from Web pages, the labels *Page Head*, *Page Tail*, *Nav Bar*, and *Info Block* are naturally required to denote different parts of a Web page. The labels *Data Record* and *Data Region* are also required for detecting data records. To denote blocks that do not contain any meaningful information, such as the attributes to be extracted and the head, tail or navigation bar of a Web page, the label *Note Block* is also required. All these labels are general to any Web data extraction problem, and they are independent of any specific object type.

Object-type Dependent Labels: Between data record blocks and leaf blocks, there are intermediate blocks on a vision-tree. So, we must define some intermediate labels between *Data Record* and the labels in the leaf label space. These labels are object-type dependent because these intermediate blocks contain some object-type specific attribute values, but they are still inferior to a complete data record. A natural method is to use the combinations of the attributes to define intermediate labels. Of course, if we use all the possible combinations, the state space could be too large. If this is the case, we can discard unimportant combinations by considering the co-occurrence frequencies of their corresponding attribute values in the training data records. The object-type dependent labels in our product information extraction are listed in Table 1 with the format *Con_**.

5. FEATURES FOR HCRFS

As a conditional model [17], Hierarchical Conditional Random Fields have the power to incorporate any useful feature for Web data extraction. In this section, we present the types of features that are essential for record detection and attribute labeling, and discuss how to incorporate them into our HCRF model. As we explain below, some of the features were first introduced as heuristic rules in some existing methods [31][32] to detect data records. The contribution of our work is that we provide a principled integration of them by defining feature functions and learning their importance weights.

5.1 Features of Elements

For each element, we extract both content and visual features as listed in table 2. All the features can be obtained from the vision-tree since VIPS assigns position information for each node through rendering a page. Previous work [4][32][33][27] has shown the effectiveness of visual features for Web page analysis and information extraction.

5.2 Features of Blocks

After extracting the features for all the elements, a bottom-up procedure is taken to extract the visual and content features for inner blocks. We also compute the following distances for each block to exploit the regularity of similar data records in a page.

Tree Distance Features: On a vision tree**,** if the blocks are visually similar, usually their sub-trees are also similar. We define the tree distance of two blocks as a measure of their structure similarity. The tree distance of two blocks is defined as the edit distance of their corresponding sub-trees. Although the time-complexity of computing this distance could be quite high, we can substantially reduce the computation with some rules. For example, if the depth difference of two sub-trees is too large, they are not likely to be similar and this computation is not necessary.

Once we have computed the tree distances, we can use some thresholds to define Boolean-valued feature functions. For example, an f function is defined as:

$$f_k(\mathrm{y}_i, \mathrm{y}_{i+1}, \mathrm{x}) = \begin{cases} true, & if\ \ \mathrm{y}_i = Data\ Record,\ \mathrm{y}_{i+1} = Data\ Record \\ & \quad and\ TreeDist(\mathrm{x}_i, \mathrm{x}_{i+1}) \le 0.2 \\ false, & otherwise \end{cases}$$

That is, if the tree distance of two adjacent blocks x_i and x_{i+1} is not more than 0.2, they are both likely to be data records.

Table 2. The content and visual features of each element

Name	Description
Content	The Content of a text element
Link URL	The action URL of an element if it exists
Font Size	The font size of an element
Font Weight	The font weight of an element
Position	The coordinates of an element
Height	The height of an element's rectangle
Width	The width of an element's rectangle
Area	The area of an element's rectangle
Image Source URL	The source URL of an image element
Image Alt-Text	The alternative text of an image element
Tag	The tag name of an element

Shape Distance & Type Distance Features: Based on the ideas in [32], we also compute the shape distance and type distance of two blocks to exploit their similarity. For shape distance, we use the same definition of shape codes and also use the same shape distance calculation method as in [32]. Due to space limitations, we would like to refer the readers to [32] for more details.

To compute the type distance of two blocks, we define the following types for each element:

- IMAGE: the element is an image.
- JPEG IMAGE: the image element that is also a jpeg picture.
- CODED IMAGE: the image element whose source URL contains at least three succeeding numbers, such as "/products/s_thumb/eb04iu_0190893_200t1.jpg".
- TEXT: the element has text content.
- LINK TEXT: the text element that contains an action URL.
- DOLLAR TEXT: the text element that contains dollar sign.
- NOTE TEXT: the text element whose tag is "input", "select" or "option".
- NULL: the default type of each element.

After defining each element's type code, a block's type code is defined as a sequence of the type codes of its children. As in [32], we also compress the multiple consecutive occurrences of each type to one occurrence and take the edit distance of type codes as the type distance of two blocks.

Similar to the use of tree distance, we can easily incorporate shape distance and type distance into our model by defining other Boolean-valued feature functions with some pre-determined thresholds. Notice that our method will not be sensitive to these thresholds because the defined feature functions are softened by learning a weight for each of them. As in formula (2), each feature function contributes its weight to the probability only when it is active, that is, the returned value is *true*. If a feature function is always true or false, it has no effect on the probability; and if some feature functions appear sparsely in the training set, smoothing techniques [6] can be used to avoid over-fitting. Here, we use the spherical Gaussian prior to penalize the log-likelihood function as in section 3.3.

5.3 Global Features

As described in the introduction, data records in the same Web page are always related and mutually constrained. Based on the work in [31], we try to align the elements of the two adjacent blocks in the same Web page and extract some global features to help the labeling of related records.

For two neighboring blocks, we try to align their elements using the partial tree-alignment algorithm [31]. An alignment is discarded if most of the elements are not aligned. For successful alignments, the following feature is extracted.

Repeated elements are less informative: This feature is based on the observation that repeated elements in different records are more likely to be less useful while important information such as the *name* of a product is not likely to repeat in multiple records in the same Web page. For example, the "Add to cart" button appears in both data records as in Figure 1(a), but each record has a unique name. Similar ideas have been exploited in [30] to remove Web noise. Currently, we just denote repeated elements in different blocks and do not take a complex information measure as in [30]. More complex measures can be easily adopted if needed in the future. An example feature function is defined as:

$$g_k(\mathrm{y}_i, \mathrm{x}) = \begin{cases} \textit{true}, & \textit{if } \mathrm{y}_i = \textit{Note and } \mathrm{x}_i \textit{ is repeated} \\ \textit{false}, & \textit{otherwise} \end{cases}$$

That is, if the element x_i repeatedly appears in the aligned records, it will be more likely to be labeled as Note.

6. EXPERIMENTS

In this section, we report empirical results by applying our HCRF model to extract structured product information from both list and detail pages. We compare our proposed model with a sequential approach that combines the state-of-the-art record detection and attribute labeling algorithms. The results show that our model achieves significant improvements in both record detection and attribute labeling on list pages, and also performs promisingly on detail pages. We also analyze to what extent the gains come from: joint optimization of record detection and attribute labeling, the global features, and long distance dependencies.

6.1 Methods

We build the baseline methods by sequentially combining the record detection algorithm DEPTA [31] and 2D CRFs [33], and for detail pages, which the DEPTA algorithm cannot deal with, we first detect the main data block using [27] and then use a 2D CRF model to do attribute labeling on the detected main block. Our method is built by combining VIPS [4], and the HCRF model. A Web page is first segmented by VIPS to construct the vision-tree, and then an HCRF model is used to label both records and attributes in the vision-tree.

To see the effect of the global features as in section 5.3, we evaluate an HCRF model that does not incorporate the global features. We denote this model by *HCRF_NG* (without global features). Similarly, we evaluate DEPTA and two 2D CRF models in the baseline methods. As in [33], a basic 2D CRF model is setup with only the basic features (see Table 2) of the elements when labeling each detected data record. A variant model is setup with both the basic features and the global features as in section 5.3. We denote the basic model by 2D CRF and denote the variant model by *2D CRF_G*. For 2D CRF_G, we first cache all the detected records from one Web page and then extract the global features as in section 5.3. Both the basic features and the extracted global features are used during model learning and attribute labeling. As there is no tree-structure here, the alignments are based on the elements' relative positions in each record.

To evaluate the separate effect of our approach on record detection and attribute labeling, we first detect data records on the parsed vision-trees using the content features and the tree distance, shape distance, and type distance features. Then we use HCRF models to label the detected records. When doing attribute labeling, we also evaluate two HCRF models with and without the global features. These two models are denoted by *HCRF_S* and *HCRF_SNG* respectively.

6.2 Datasets

We setup two general datasets with randomly crawled product Web pages. The list dataset (*LDST*) contains 771 list pages and the detail dataset (*DDST*) contains 450 detail pages. All of the pages are parsed by VIPS. For HCRF models, we use 200 list pages and 150 detail pages together to learn their parameters. All these pages are hierarchically labeled, that is, every block in the parsed vision-trees is labeled. We use the same 200 list pages to train a 2D CRF model for extraction on list pages, and use the same 150 detail pages to train another 2D CRF model for extraction on detail pages. The reason for training two models for list and detail pages separately is that, for a 2D CRF model, the features and parameters for list and detail pages are quite different and a uniform model cannot work well. In the training stage, all of the algorithms converge quickly, within 20 iterations. We use the remaining pages (571 list pages and 300 detail pages) for testing. Totally 6387 data records are found in the testing set *LDST*.

6.3 Evaluation Metrics

For data record detection, we use the standard precision, recall and F1 measures to evaluate the methods. A block is considered as a correctly detected data record if it contains all the appeared attributes, otherwise it's not a data record. In our experiments, a data record could tolerate (miss or contain) some non-important information like "Add to Cart" button.

For attribute labeling, we use the same measures defined in [33]. The performance on each attribute is evaluated by precision (i.e. the percentage of returned elements that are correct), recall (i.e. the percentage of correct elements that are returned), and their harmonic mean F1. We also use two comprehensive evaluation criteria:

Block Instance Accuracy (BLK_IA): the percentage of data records of which the key attributes (name, image, and price) are all correctly labeled;

Average F1 (AVG_F1): the average of F1 values of different attributes.

6.4 Experimental Results and Discussions

We compare our approach with DEPTA [31] on *LDST* for data record detection. The running results of DEPTA on our dataset are kindly provided by its authors. DEPTA has a similarity

Table 3. Evaluation results of different methods on both *LDST* and *DDST*, where Desc stand for 'Description'.

Data Sets			*LDST*						*DDST*	
Methods			HCRF _SNG	HCRF _S	HCRF _NG	HCRF	DEPTA+ 2D CRF	DEPTA+ 2D CRF _G	HCRF	2D CRF
Record Detection		Precision	0.904	0.904	0.959	**0.959**	0.884	0.884	—	—
		Recall	0.921	0.921	0.930	**0.930**	0.849	0.849	—	—
		F1	0.912	0.912	0.944	**0.944**	0.866	0.866	—	—
Attribute Labeling	Precision	Name	0.836	0.860	0.880	**0.911**	0.763	0.851	**0.835**	0.398
		Image	0.901	0.905	0.952	**0.966**	0.842	0.838	**0.978**	0.546
		Price	0.906	0.903	0.959	**0.963**	0.913	0.915	**0.986**	0.809
		Desc	0.783	0.766	**0.792**	0.788	0.769	0.779	**0.663**	0.588
	Recall	Name	0.851	0.875	0.854	**0.882**	0.735	0.822	**0.761**	0.398
		Image	0.917	0.921	0.924	**0.936**	0.811	0.809	**0.892**	0.546
		Price	0.922	0.919	0.930	**0.933**	0.879	0.883	**0.899**	0.809
		Desc	**0.797**	0.780	0.768	0.764	0.741	0.752	**0.604**	0.395
	F1	Name	0.843	0.867	0.867	**0.896**	0.749	0.836	**0.796**	0.398
		Image	0.909	0.913	0.938	**0.951**	0.826	0.823	**0.933**	0.546
		Price	0.914	0.911	0.944	**0.948**	0.896	0.899	**0.940**	0.809
		Desc	**0.790**	0.773	0.780	0.776	0.755	0.765	**0.632**	0.473
	AVG_F1		0.864	0.866	0.882	**0.893**	0.807	0.831	**0.825**	0.556
	BLK_IA		0.789	0.816	0.856	**0.890**	0.669	0.751	**0.817**	0.231

threshold, and it is set at 60% in this experiment. Some simple heuristics are also used in DEPTA to remove some noise records. For example, a data region that is far from the centre or contains neither image nor dollar sign is removed. The evaluation results are shown in Table 3.

6.4.1 *Record Detection*

For record detection, we can see that both HCRF and HCRF_NG significantly outperform DEPTA in recall, improved by 8.1 points, and precision, improved by 7.5 points. The improvements come from two parts:

Advanced data representation and more features: in our model, we incorporate more features such as content features and the shape distance & type distance features as in section 5.2 than DEPTA. We also adopt an advanced representation of Web pages – vision-trees which have been shown to outperform tag-tree representation in other work [4]. As we can see from Table 3, HCRF_SNG and HCRF_S outperform DEPTA, and we gain about 2 points in precision, 7.3 points in recall, and 4.6 points in F1.

Incorporation of semantics during record detection: DEPTA just detects the blocks with regular patterns (i.e. regular tree structures) and does not take semantics into account. Thus, although some heuristics are used to remove some noise blocks, the results could still contain blocks that are not data records or just parts of data records. In contrast, our approach can integrate attribute labeling into block detection, and can consider the semantics during detecting data records. So, the blocks detected are of better quality and are more likely to be data records. For instance, a block containing a product's name, image, price and some descriptions is almost certain to be a data record, but a block containing only irrelevant information is unlikely to be a data record. The lower precisions of HCRF_SNG and HCRF_S demonstrate this. When not considering the semantics of the elements, HCRF_SNG and HCRF_S extracts more noise blocks compared with HCRF_NG or HCRF, so the precisions of record detection decrease by 5.5 points and the overall F1 measures decrease by 3.2 points.

6.4.2 *Attribute Labeling*

As we can see from Table 3, our HCRF model significantly outperforms the baseline approach. On list pages, HCRF_NG gains 18.7 points over 2D CRF in block instance accuracy and the achievements of HCRF are 13.9 points compared with 2D CRF_G. On detail pages, our approach gains about 58 points over 2D CRF in block instance accuracy. The reasons for the better performance are:

Attribute labeling benefits from good quality records: One reason for this better performance is that attribute labeling can benefit from the good results of record detection. For example, if a detected record is not a data record or misses some important information such as name, the attribute labeling will fail to find the missed information back or just finds a wrong one. So, HCRF_SNG outperforms 2D CRF and HCRF_S outperforms 2D CRF_G. Of course the achievements of HCRF_SNG and HCRF_S may also come from the incorporation of long distance dependencies in hierarchical CRFs which will be discussed later.

Global Features help Attribute Labeling: Another reason for the improvements in attribute labeling is the incorporation of the global features as in section 5.3. From the results, we can see that when considering the global features, attribute labeling is more accurate. For example, 3.4 points are gained in block instance accuracy by HCRF compared with the basic HCRF_NG model. For two separate HCRF models, HCRF_S achieve 2.7 points in block instance accuracy compared with HCRF_SNG; for the two baseline methods, compared with 2D CRF which uses only the features of the elements in each detected record, more than 8 points are gained in block instance accuracy by 2D CRF_G which incorporates the global features.

HCRF Models incorporate Long Distance Dependencies: The third reason for the better performance in attribute labeling is the incorporation of long distance dependencies. From the results on detail pages, we can see that hierarchical models could get promising results while 2D CRFs perform poorly. For a detected record or a main block, 2D CRFs put its elements in a two-dimensional grid. Although they could incorporate more interactions compared with linear-chain CRFs [17], long distance interactions cannot be incorporated due to their first-order Markov assumption. In contrast, HCRF models can incorporate dependencies at various levels of granularity and thus incorporate long distance dependencies. So, HCRF models perform much better than 2D CRF models. For detail pages, as there is no record detection, HCRF_SNG and HCRF_S are not applicable here. Also there are no global features, so we just list the results of HCRF and 2D CRF on detail pages in Table 3.

The quite different performance of 2D CRFs on list and detail pages also shows the effectiveness of long distance dependencies. For list pages, the inputs are data records that always contain a small number of elements. In this case, 2D CRFs can effectively model the dependencies of the attributes and achieve reasonable accuracy. However, noisy elements always exist between attribute values in a detail page. So, 2D CRFs perform very poorly for their failure in incorporating long distance dependencies.

7. RELATED WORK

Wrapper learning approaches like [23][16] are template-dependent. They take in some manually labeled Web pages and learn some extraction rules (i.e. wrappers). Since the learned wrappers can only be used to extract data from similar pages, maintaining the wrappers as Web sites change will require great efforts. Furthermore, in wrapper learning a user must provide explicit information about each template. So it will be expensive to train a system that extracts data from many Web sites as in our application. [32][10][3][5][9][1] are also template-dependent, but they do not need labeled training samples. They automatically produce wrappers from a collection of similar Web pages.

[31][18] are two template-independent methods. [18] segments data on list pages using the information contained in their detail pages. The need of detail pages is a limitation because automatically identifying links that point to detail pages is non-trivial and there are also many pages that do not have detail pages behind them. [31] mines data records by string matching and also incorporates some visual features to achieve better performance. However, [31] detects data records only using tree regularities and not consider semantics. Furthermore, the data extracted by [31][18] have no semantic labels.

[12][33] are two methods that treat Web information extraction as a classification problem. [12] uses a support vector machine to identify the start and end tags for a single attribute. For the task of extracting multiple attributes, this method loses the dependencies between different attributes. [33] shows that these dependencies are common among several types of objects and also proposes 2D CRFs to better incorporate these dependencies. However, it has two assumptions: (1) data records are independently and identically distributed; (2) data records have been perfectly extracted. As described in the introduction, these assumptions will result in highly ineffective de-coupled solutions.

The idea of exploring mutual benefits by integrating two tasks has been attempted in previous work. [24] attempts a mutually beneficial integration of information extraction and data mining. Information extraction makes possible the text mining which needs to handle unstructured text documents, and data mining could provide additional clues to improve the recall of an IE system. [29] proposes an integrated model to do information extraction and coreference. Incorporating extraction uncertainty could help coreference, and leveraging the identified coreference could improve extraction accuracy. However, [29] is not a fully closed integration. Due to its model's complexity, separate learning and inference for different substructures is employed in [29]. [28] proposes a factorial CRF to jointly solve two NLP tasks (noun phrase chunking and part of speech tagging) on the same observation sequence. The difference is that our data are hierarchical trees and the data in [28] are sequences.

[26] uses Hierarchical Hidden Markov models (HHMMs) [11] to extract relation instances from biomedical articles. Our work differs from this one in two key aspects. First, as discriminative models HCRFs have the flexibility to incorporate arbitrary and non-independent features of the observations, but HHMMs, which are generative models, must make some strong independence assumption to achieve inference tractability. This is the key idea underlying the Conditional Random Fields [17]. Second, the data in [26] are two-level representations of sentences, but our data are arbitrary vision-trees. Recently, several hierarchical structured Conditional Random Field models [13][15][20] have been proposed in different domains. Unlike [13], our HCRF model is not a simple multi-scale model because it has inter-level interactions. In [15], a two-layer structured CRF model is proposed to incorporate both local and global label interactions for robust image classification. Unlike our model which is a Conditional Random Field as a whole, each layer of [15] is a separate Conditional Random Field and the two layers are coupled with directed links. In [20], a three layer Conditional Random Field model is dynamically constructed for activity recognition. In contrast, we use a layout data structure 'vision tree' and our hierarchical CRF model is built upon it. Due to its dynamic nature, [20] must take an iterative process and the whole inference is approximate. But our HCRF model can find the optimal labeling results.

Other work, such as collective information extraction [2] and Semi-Markov extraction models [7][25], could achieve higher performance in named entity extraction problems on flat text documents. However, for the integrated Web data extraction, where the data are hierarchically represented, the proposed hierarchical CRF model is the natural and efficient method.

8. CONCLUSIONS

In this paper, we propose a probabilistic model called Hierarchical Conditional Random Fields (HCRFs) to exploit the mutually beneficial integration of data record detection and attribute labeling. In this model, record detection can benefit from the availability of semantics which are required in attribute labeling, and attribute labeling could benefit from the collective labeling of similar records in the same Web page. Parameter estimation and labeling could be efficiently performed by applying the standard junction tree algorithm. As a conditional model, HCRFs can incorporate any useful feature for Web data extraction in a principled way; as a hierarchical model, HCRFs provide a way to incorporate long distance dependencies for Web data. Empirical studies show that mutual enhancement of record detection and attribute labeling can be achieved in our joint approach, and HCRFs can perform very well on both list and detail Web pages.

9. ACKNOWLEDGMENTS

The authors are grateful to Yanhong Zhai and Bing Liu for providing the running results of their system DEPTA on our dataset *LDST*. The authors Jun Zhu and Bo Zhang are supported by National Natural Science Foundation of China (60135010, 60321002), and Chinese National Key Foundation Research and Development Plan (2004CB318108).

10. REFERENCES

[1] Arasu, A., and Garcia-Molina, H. Extracting Structured Data from Web Pages. In *Proc. of ACM SIGMOD*, 2003.

[2] Bunescu, R. C., and Mooney, R. J. Collective information extraction with relational Markov networks. *In Proc. of ACL*, 2004.

[3] Buttler, D., Liu, L., and Pu, C. A Fully Automated Object Extraction System for the World Wide Web. In *Proc. of IEEE ICDCS*, 2001.

[4] Cai, D., Yu, S., Wen, J.-R. and Ma, W.-Y. Block-based Web Search. In *Proc. of SIGIR*, 2004.

[5] Chang, C.-H., and Liu, S.-L. IEPAD: Information Extraction Based on Pattern Discovery. In *Proc. of WWW*, 2001.

[6] Chen, S. F., and Rosenfeld, R. A Gaussian Prior for Smoothing Maximum Entropy Models. Technical Report CMU-CS-99-108, Carnegie Mellon University, 1999.

[7] Cohen, W. W., and Sarawagi, S. Exploiting Dictionaries in Named Entity Extraction: Combining Semi-Markov Extraction Processes and Data Integration Methods. In *Proc. of SIGKDD*, 2004.

[8] Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. Probabilistic Networks and Expert Systems. Springer, 1999.

[9] Crescenzi, V., Mecca, G., and Merialdo, P. ROADRUNNER: Towards Automatic Data Extraction from Large Web Sites. In *Proc. of VLDB*, 2001.

[10] Embley, D. W., Jiang, Y., and Ng, Y.-K. Record-Boundary Discovery in Web Documents. In *Proc. of SIGMOD*, 1999.

[11] Fine, S., Singer Y., and Tishby, N. The hierarchical hidden Markov model: Analysis and applications. Machine Learning, 32:41-62, 1998.

[12] Finn, A., and Kushmerick, N. Multi-level boundary classification for information extraction. In *Proc. of ECML*, 2004.

[13] He, X., Zemel, R. S., and Carreira-Perpiñán, M. Á. Multiscale Conditional Random Fields for Image Labeling. In *Proc. of CVPR*, 2004.

[14] Jensen, F. V., Lauritzen, S. L., and Olesen, K. G. Bayesian updating in causal probabilistic networks by local computation. Computational Statistics Quarterly, 4:269-82, 1990.

[15] Kumar, S., and Hebert, M. A Hierarchical Field Framework for Unified Context-Based Classification. In *Proc. of ICCV*, 2005.

[16] Kushmerick, N. Wrapper induction: efficiency and expressiveness. Artificial Intelligence, 118:15-68, 2000.

[17] Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *Proc. of ICML*, 2001.

[18] Lerman, K., Getoor, L., Minton, S., and Knoblock, C. Using the Structure of Web Sites for Automatic Segmentation of Tables. In *Proc. of ACM SIGMOD*, 2004.

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

[20] Liao, L., Fox, D., and Kautz, H. Location-based activity recognition. In *Proc. of NIPS*, 2005.

[21] Liu, D. C., and Nocedal, J. On The Limited Memory BFGS Method for Large Scale Optimization. Mathematical Programming 45, pp. 503-528, 1989.

[22] Malouf, R. A comparison of algorithms for maximum entropy parameter estimation. In *Sixth Conf. on Natural Language Learning*, pages 49-55, 2002.

[23] Muslea, I., Minton, S., and Knoblock C. A. Hierarchical Wrapper Induction for Semi-structured Information Sources. Autonomous Agents and Multi-Agent 4, 1/2 (2001), 2001.

[24] Nahm, U. Y., and Mooney, R. J. A Mutually Beneficial Integration of Data Mining and Information Extraction. In *Proc. of AAAI*, 2001.

[25] Sarawagi, S., and Cohen, W. W. Semi-Markov Conditional Random Fields for Information Extraction. In *Proc. of NIPS*, 2004.

[26] Skounakis, M., Craven, M., and Ray S. Hierarchical Hidden Markov Models for Information Extraction. In *Proc. of IJCAI*, 2003.

[27] Song, R., Liu, H., Wen, J.-R., and Ma, W-Y. Learning Block Importance Models for Web Pages. In *Proc. of WWW*, 2004.

[28] Sutton, C., Rohanimanesh, K., and McCallum, A. Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data. In *Proc. of ICML*, 2004.

[29] Wellner, B., McCallum, A., Peng, F., and Hay, M. An Integrated, Conditional Model of Information Extraction and Coreference with Application to Citation Matching. In *Proc. of UAI*, 2004.

[30] Yi, L., Liu, B., and Li, X. Eliminating Noisy Information in Web Pages for Data Mining. In *Proc. of SIGKDD*, 2003.

[31] Zhai, Y., and Liu, B. Web Data Extraction Based on Partial Tree Alignment. In *Proc. of WWW,* 2005.

[32] Zhao, H., Meng, W., Wu, Z., Raghavan, V., and Yu, C. Fully Automatic Wrapper Generation for Search Engines. In *Proc. of WWW*, 2005.

[33] Zhu, J., Nie, Z., Wen, J.-R., Zhang, B., and Ma, W.-Y. 2D Conditional Random Fields for Web Information Extraction. In *Proc. of ICML*, 2005.