Web Information Extraction Using Markov Logic Networks

Sandeepkumar Satpal [†1] Sahely Bhadra [#2] S Sundararajan [*3] Rajeev Rastogi [*4] Prithviraj Sen [*5]

[†]Microsoft
Hyderabad India

[*]Yahoo! Labs
Bangalore India

[#]CSA, Indian Institute of Science
Bangalore India

[1]ssatpal@microsoft.com,{[3]ssrajan, [4]rrastogi, [5]sen}@yahoo-inc.com, [2]sahely@csa.iisc.ernet.in

ABSTRACT

In this paper, we consider the problem of extracting structured data from web pages taking into account both the content of individual attributes as well as the structure of pages and sites. We use *Markov Logic Networks* (MLNs) to capture both content and structural features in a single unified framework, and this enables us to perform more accurate inference. MLNs allow us to model a wide range of rich structural features like *proximity*, *precedence*, *alignment*, and *contiguity*, using first-order clauses. We show that inference in our information extraction scenario reduces to solving an instance of the *maximum weight subgraph* problem. We develop specialized procedures for solving the maximum subgraph variants that are far more efficient than previously proposed inference methods for MLNs that solve variants of MAX-SAT. Experiments with real-life datasets demonstrate the effectiveness of our MLN-based approach compared to existing state-of-the-art extraction methods.

1. INTRODUCTION

The web is a vast repository of human knowledge. Popular web sites like amazon.com, yelp.com, etc. contain a wealth of information about products (e.g., description, price, reviews), businesses (e.g., address, phone, category, hours of operation), restaurants, books, and so on. Extracting this information from popular web site pages can allow us to create extensive databases of entities. These databases can then be queried by search engines to improve ranking and rendering of search results, and by users to access product features and reviews, order products by price, list restaurants in a specific location, etc.

Unfortunately, much of the useful information on the web is embedded in *noisy semi-structured* web pages containing extraneous data like navigation links, sidebars, and advertizements. Consequently, extracting structured data from web pages is a key challenge, and we focus on this problem in the paper.

1.1 Statistical Extraction Approaches

In recent years, there has been a flurry of research activity on statistical models for information extraction [2, 1, 6, 19, 16]. These models are learnt from small amounts of labeled training data from a few initial web sites, and so incur little manual effort. Furthermore, they leverage both content and structural cues to detect attribute values, and are tolerant to errors and noise in the input pages. For extracting data from the web with little human supervision, *Hierarchical Conditional Random Fields* (HCRFs) [19] represent the state-of-the-art today. HCRFs are graphical models that extend CRFs to include parent-child and sibling dependencies between nodes in the tree corresponding to a web page.

Despite their expressive power, a limitation of HCRFs is that they only capture short-range structural dependencies between neighboring elements in the DOM tree. However, in practice, web sites contain long-range structural dependencies involving non-adjacent elements. For example, a pair of attribute values within a page may be separated by noise, but one may always appear before the other (e.g., name appears before other attributes in business or product pages), or the two may always appear close together. Similarly, in a template-based site, attribute values across pages occur in similar locations. HCRFs cannot express such long-range structural relationships between non-adjacent elements, and this can adversely impact extraction accuracy.

In recent work, [16] employed *Markov Logic Networks* (MLNs) to extract structured data from various web forum sites. MLN models provide a highly expressive framework based on first-order logic that allows long-range structural dependencies between arbitrary web page elements (and not just neighbors) to be specified. However, the expressiveness comes at a higher computational cost, for e.g., inference in MLNs is intractable and equivalent to the weighted MAX-SAT problem, which is known to be NP-hard [12]. Generally, satisfiability solvers (e.g., [7]) are used for MLN inference; however, solvers operate on ground formulas and so their performance degrades severely as domain sizes grow bigger and formulas become more complex. Thus, for MLNs to be practically applicable, faster inference engines are needed.

1.2 Our Contributions

In this paper, we develop an MLN-based framework for general-purpose extraction from web sites (and not just web forum sites). Our MLN models capture both attribute content properties as well as long-range structural relationships in a single unified framework, and this enables us to perform extractions with higher accuracy.

A key contribution of our work is a novel set of structural features (expressed as first-order formulas) for the general web extraction problem. Structural features capture intra-page and inter-page relationships among attribute value occurrences within a page and across pages, respectively. Examples of structural features include: (1) *Precedence:* In restaurant or business web pages, the name attribute typically precedes address which in turn visually occurs before phone number, (2) *Proximity:* Name and address typically occur close together, and so do address and phone number, (3) *Contiguity:* Attributes like address can span multiple web page nodes, but these nodes appear contiguously within the page, and finally, (4) *Alignment:* Across template-based pages of a web site, attribute values occur in similar positions.

Observe that the above precedence, proximity, contiguity, and alignment features are long-range and can involve non-adjacent elements. Thus, they are ideally suited for data extraction from web pages containing attribute values interspersed with noise. Existing models like HCRFs cannot express such features. Also, many of the features like precedence, proximity, and contiguity have not been used before by statistical extraction methods.

A second major contribution of our work is a specialized procedure for speeding up inference in MLNs. Specifically, in our extraction scenario, formulas are constrained – this allows us to pose the inference problem as finding a maximum weight subgraph within a graph. We develop a greedy heuristic for finding the maximum weight subgraph that is several orders of magnitude faster compared to the traditional inference methods for MLNs based on solving MAX-SAT variants. Our graph-based algorithm operates at a higher level of abstraction and is thus able to efficiently handle simple constraints like "attribute values are contiguous" that can be very expensive to enforce within a satisfiability solver.

Finally, in our experiments with real-life datasets, we found that exploiting rich structural features enables our MLN-based approach to extract attributes with significantly higher accuracy than the state-of-the-art HCRF approach. Furthermore, our graph-based inference algorithm ran in the order of minutes for inference tasks which satisfiability solvers took days to complete.

2. PROBLEM FORMULATION

In this section, we formally define the extraction problem that we address in this paper. We start by describing our model for data on the web. Subsequently, we present the machine learning framework we use for information extraction, and the associated problems related to inference and learning.

2.1 Web Data Model

Consider a set of web sites $\mathbf{W}$ belonging to a specific domain like Restaurants, Books, etc. For each of these domains, there exists a well-defined schema that specifies the attributes to be extracted. For example, attributes like Name, Address, Price and Phone are part of the Restaurants schema. We denote the set of attributes that we want to extract from the web sites in $\mathbf{W}$ by $\mathbf{A}$. In addition to the traditional attributes, $\mathbf{A}$ includes the special attribute Noise that denotes the noisy information contained in web sites.

Each web site $W \in \mathbf{W}$ consists of a set of templatized pages with similar structure. For each web page $p \in W$, consider the DOM tree representation of the page. Then the extraction problem is to assign *attribute labels* to all the leaf nodes in the DOM tree for p.

2.2 Markov Logic Networks

Markov Logic Networks (MLNs) [12] provide a powerful probabilistic modeling framework based on *first-order logic*. Formally, an MLN is a set of pairs (F_i, w_i), where F_i is a first-order formula and w_i is its corresponding weight. The weight of a formula is essentially a measure of its importance. Formulas are defined over a set of application-specific predicates. The set of predicates are categorized as *query (or hidden)* and *evidence (or observed)* predicates, and the formulas capture the various relationships between these predicates. For example, in our extraction application, the query predicates are the attribute labels assigned to page nodes n like $\mathsf{IsName}(n)$, $\mathsf{IsAddress}(n)$, etc., and the evidence predicates are the observed content and structural features over nodes like $\mathsf{Has5Digits}(n)$, $\mathsf{FirstLetterCapital}(n)$, $\mathsf{Close}(n_1,n_2)$, etc. Then using such predicates, formulas like $\forall n\ \mathsf{Has5digits}(n) \Rightarrow \mathsf{IsZipCode}(n)$ and $\forall n_1, n_2\ \mathsf{IsName}(n_1) \wedge \mathsf{IsAddress}(n_2) \Rightarrow \mathsf{Close}(n_1, n_2)$ are formed. We describe the evidence predicates and formulas that we employ for information extraction in more detail in Section 3.

Now, for a web site W, let x be the set of evidence predicates that are true for pages in W. Then, the probability that the set of query predicates q is true is given by:

$$P(q|x) = \frac{1}{Z}\exp\Big(\sum_{F_i}\sum_{g\in G_i} w_i \cdot g(q \cup x)\Big) \qquad (1)$$

where G_i is the set of groundings[1] of F_i, $g(q \cup x)$ equals to 1 if the grounded formula g is true for predicate set $q \cup x$ and 0 otherwise, and Z is a normalization constant which ensures that probabilities add up to 1.

Our web data extraction problem is to find the most likely attribute label assignment for page nodes, and this reduces to performing maximum a posteriori (MAP) inference in our MLN model.

Web information extraction problem: Given an MLN model with (formula, weight) pairs (F_i, w_i) and a web site $W \in \mathbf{W}$ with true evidence predicates x, compute query predicates q^* such that $P(q^*|x)$ is maximum. □

Note that finding the assignment q^* that maximizes the likelihood $P(q^*|x)$ is equivalent to maximizing the sum of weighted formulas given by: $\sum_{F_i}\sum_{g\in G_i} w_i \cdot g(q \cup x)$. Thus, the inference problem is an instance of the weighted MAX-SAT problem, which is known to be NP-hard [12]. One of the most efficient approaches to this problem is stochastic local search, exemplified by the MaxWalkSAT solver [7]. Beginning with a random truth assignment, MaxWalkSAT repeatedly flips the truth value of a predicate in a random unsatisfied formula. The flipped predicate is probabilistically selected to be either (1) the predicate that maximizes the weight of satisfied formulas, or (2) a random predicate.

In our experiments (see Section 5), we found that satisfiability solvers like MaxWalkSAT perform poorly for our extraction application. Instead, in this paper, we devise an

[1]Groundings of a formula are obtained by instantiating variables with web page nodes in W.

efficient graph-based inference algorithm that exploits the structure and semantics of the formulas.

2.3 Learning the Model Parameters

Given the formulas F_i in our MLN model (Equation (1)), the learning problem is to learn the weights w_i. For this, we use human labeled pages from a small subset of web sites $\mathbf{W}_t \subseteq \mathbf{W}$ which serve as training examples. For a site $W \in \mathbf{W}_t$, let q_W and x_W denote the true query and evidence predicates, respectively. Then, the parameter learning problem is to find the weights w_i that maximize the log likelihood function $\sum_{W \in \mathbf{W}_t} \log(P(q_W|x_W))$. We use the *Margin Infused Relaxed Algorithm* (MIRA) [4] which updates weights iteratively using one example at a time and whenever the MAP prediction on the current example is different from the actual label. Thus, our learning algorithm involves solving the inference problem internally in each iteration. Therefore, fast inference algorithms are critical for efficient learning.

3. PREDICATES AND FORMULAS

As mentioned earlier, we have two categories of predicates: *query* and *evidence*. For each attribute A_j, there is a query predicate $\mathsf{IsA_j}$ which is true for a node if the node is assigned the attribute label A_j. Examples include $\mathsf{IsName}(n)$, $\mathsf{IsAddress}(n)$, etc.

Evidence predicates can be further classified into *content* and *structural* depending on the nature of the features captured by the predicates. Furthermore, content (structural) formulas specify the relationships between content (structural) and query predicates.

3.1 Content Predicates and Formulas

Content predicates capture the local content features for each individual node like word features, orthographic features, and dictionary-based features [14]. Some examples of content predicates are: $\mathsf{Has5Digits}(n)$, $\mathsf{FirstLetterCapital}(n)$, $\mathsf{ContainsWordRoad}(n)$, $\mathsf{ContainsReviewDictionaryWord}(n)$, $\mathsf{FontSizeLarge}(n)$, etc.

Content formulas link content predicates with query predicates for a node. For each content predicate, attribute pair, C_i, A_j (including Noise), we create a separate content formula $\forall n\ C_i(n) \Rightarrow \mathsf{IsA_j}(n)$. An example of a content formula is $\forall n\ \mathsf{ContainsWordRoad}(n) \Rightarrow \mathsf{IsAddress}(n)$. Observe that content formulas involve a single node n.

3.2 Structural Predicates and Formulas

In our application, apart from content predicates, we also employ structural predicates and formulas to capture long-range relationships (e.g., proximity, precedence, alignment) among *non-noise* attribute values within and across pages.

(1) *Proximity:* The proximity predicate $\mathsf{Close}(n_1, n_2)$ reflects the *closeness* between a pair of nodes n_1, n_2 in a page p. Here, closeness is defined based on the visual coordinates of nodes within a page. We consider nodes n_1, n_2 to be close if the distance between them is less than 10% of the maximum distance between node pairs in the page. Proximity formulas have the general form $\forall n_1, n_2 \in p\ \mathsf{IsA_i}(n_1) \wedge \mathsf{IsA_j}(n_2) \Rightarrow \mathsf{Close}(n_1, n_2)$. We create a separate formula for each pair A_i, A_j of non-noise attributes and each page p. An example is $\forall n_1, n_2 \in p\ \mathsf{IsName}(n_1) \wedge \mathsf{IsAddress}(n_2) \Rightarrow \mathsf{Close}(n_1, n_2)$. Proximity formulas help to boost extraction accuracy by exploiting the fact that non-noise attributes typically appear close together in a page. For example, if there are multiple addresses present in a restaurant page (e.g., for related restaurants), and we are interested only in the address of the restaurant that the page is about, then the other addresses (for related restaurants) can be eliminated with the help of such formulas.

(2) *Precedence:* Precedence formulas specify *ordering* relationships among the attributes. For example, suppose that no non-noise attribute can occur before the attribute Name. Such an order can be captured using precedence formulas like $\forall n_1, n_2 \in p\ \mathsf{IsName}(n_1) \wedge \mathsf{IsNumberofPages}(n_2) \Rightarrow \mathsf{Precedes}(n_1, n_2)$. The structural predicate $\mathsf{Precedes}(n_1, n_2)$ is true if the position of node n_1 lies either above or to the left of n_2. For all non-noise attribute pairs A_i, A_j and pages p, we introduce precedence formulas of the form $\forall n_1, n_2 \in p\ \mathsf{IsA_i}(n_1) \wedge \mathsf{IsA_j}(n_2) \Rightarrow \mathsf{Precedes}(n_1, n_2)$. Such formulas are especially useful in our web extraction scenario where non-noise values are frequently interspersed with noise. Consequently, neighboring values of a non-noise attribute may contain few cues, and precedence formulas are needed to enforce the longer-range dependencies.

(3) *Alignment:* The structural predicate $\mathsf{Aligned}(n_1, n_2)$ captures the similarity in non-noise attribute value positions across the template-based pages of a web site. We consider a pair of nodes n_1, n_2 in two different pages p_1, p_2 to be aligned (that is, $\mathsf{Aligned}(n_1, n_2)$ is true) if n_1 and n_2 have identical DOM paths. Note that it is possible that multiple nodes within a page have the same DOM path, and so there may be multiple nodes that are aligned with a given node. This can complicate extraction using the Aligned predicate because aligned nodes across pages may not necessarily belong to the same attribute. However, nodes belonging to the same attribute do tend to be aligned, and so we use alignment formulas of the form $\forall n_1 \in p_1, n_2 \in p_2\ \mathsf{IsA_j}(n_1) \wedge \mathsf{IsA_j}(n_2) \Rightarrow \mathsf{Aligned}(n_1, n_2)$ for distinct page pairs p_1, p_2 and non-noise attribute A_j. An example is $\forall n_1 \in p_1, n_2 \in p_2\ \mathsf{IsName}(n_1) \wedge \mathsf{IsName}(n_2) \Rightarrow \mathsf{Aligned}(n_1, n_2)$.

3.3 Hard Constraints

Apart from the *soft* formulas F_i (described in the previous sections) for which the weights w_i are learnt, another class of formulas known as *hard constraints* are very useful. These formulas are in principle assigned *infinite* weights because any world that violates them is considered highly *improbable.* We list below the hard constraints that are useful in our extraction setting.

(1) *SingleLabel*: Each node is assigned exactly one attribute label. These constraints can be expressed using the formulas $\forall n \in p\ \mathsf{IsA_i}(n) \Rightarrow \neg\mathsf{IsA_j}(n)$ for all attribute pairs A_i, A_j. An example formula is $\forall n \in p\ \mathsf{IsName}(n) \Rightarrow \neg\mathsf{IsAddress}(n)$.

(2) *Mandatory*: This constraint ensures that each page contains at least one node belonging to a mandatory attribute. For example, every restaurant or book page will necessarily contain the attribute Name. But it may not contain attributes like Description or Review. Mandatory constraints can be expressed as $\exists n \in p\ \mathsf{IsA_j}(n)$; for e.g., $\exists n \in p\ \mathsf{IsName}(n)$. Note that the mandatory constraint may be applicable only for a subset of attributes.

(3) *Contiguity*: Certain non-noise attributes like Address may span more than one node. Generally, these nodes with

the *same* attribute label are contiguous, and this is enforced by the contiguity constraint. Contiguity contraints are the most complex and have the general form involving 3 nodes: $\forall n_1 \prec n_2 \prec n_3 \in p\ \mathsf{IsA_j}(n_1) \wedge \mathsf{IsA_j}(n_3) \Rightarrow \mathsf{IsA_j}(n_2)$. Here, A_j is a non-noise attribute, and $n_k \prec n_l$ is an evidence predicate that is true if node n_k appears before n_l in the DOM tree of a page. Note that the number of groundings for contiguity constraints is cubic in the number of nodes in a page. Thus, since the number of groundings can be large, these constraints pose a serious challenge for satisfiability solvers.

4. INFERENCE ALGORITHM

We are now ready to present the algorithm for inference in our MLN model. We first show that the inference problem can be formulated as finding the maximum weight subgraph within a graph. Then, since our inference problem is NP-hard, we present a simple greedy heuristic for efficiently computing the maximum weight subgraph.

4.1 Maximum Weight Subgraph Problem

Recall from Section 2.2 that our inference problem is: For a given a site W with true evidence predicates x, find query predicates q such that $\sum_{F_i} \sum_{g \in G_i} w_i \cdot g(q \cup x)$ is maximum. Here, $g(q \cup x)$ is 1 if the ground formula g is true for predicates $q \cup x$, and 0 otherwise. Since predicates in x are always true, to simplify notation, we will drop x from our equations when it is clear from the context.

A satisfiability solver like MaxWalkSAT can be used to find the label assignment q. But MaxWalkSAT operates on ground formulas which are at a low level of abstraction. As we saw earlier, a single contiguity constraint can produce a cubic number of groundings. Consequently, low-level representations can be bulky, and many more operations are required to manipulate the large number of groundings. In this subsection, we present a higher-level graphical abstraction that captures the various formulas and constraints in a highly compact graph representation. Each graph operation maps to multiple query predicate truth value updates, and this leads to substantially higher efficiency.

Our graph framework leverages the fact that soft ground formulas g contain one or two nodes. Content formulas contain only one node while structural formulas like proximity or alignment are defined over two nodes. Hard constraints, on the other hand, can involve more than two nodes but the weights associated with them are infinity. For a node n_i, let G_{n_i} denote the set of content ground formulas containing node n_i, and let G_{n_i,n_k} denote the structural ground formulas containing nodes n_i and n_k. Then, our optimization problem is to find a q that satisfies the hard constraints and maximizes the following objective function:

$$X(q) = \sum_{n_i} \sum_{g \in G_{n_i}} w(g) \cdot g(q) + \sum_{n_i,n_k} \sum_{g \in G_{n_i,n_k}} w(g) \cdot g(q) \quad (2)$$

Above, for grounding g derived from formula F_i, the weight $w(g)$ is equal to w_i. Note that for node pairs n_i, n_k in the same page, G_{n_i,n_k} contains groundings of precedence and proximity formulas while if n_i and n_k belong to different pages, then G_{n_i,n_k} contains groundings of alignment formulas.

In the next two subsections (4.1.1 and 4.1.2), we first describe the constructed graph for maximizing Equation (2), and then in Section 4.1.3, we describe the additions needed

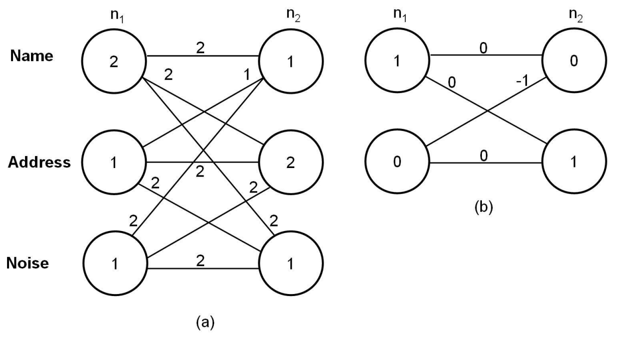

Figure 1: Graph construction example.

to enforce the hard constraints.

4.1.1 Basic Graph Construction

We show that finding a q that maximizes Equation (2) is equivalent to computing the maximum weight subgraph in a graph $\mathcal{G}$ constructed as described below.

In $\mathcal{G}$, there is a separate vertex v_{ij} for each node n_i within a page and each attribute A_j (including Noise). Thus, there are $|\mathbf{A}|$ layers of vertices in $\mathcal{G}$ with layer j corresponding to attribute A_j. Intuitively, vertex v_{ij} in $\mathcal{G}$ corresponds to the predicate $\mathsf{IsA_j}(n_i)$ in q. With each vertex v_{ij}, we associate a weight $t(v_{ij}) = \sum_{g \in G_{n_i}} w(g) \cdot g(\{\mathsf{IsA_j}(n_i)\})$ which is the sum of the weights of all the true ground formulas containing n_i when n_i is assigned the label A_j. This is essentially the contribution of n_i to the first term of $X(q)$ when $\mathsf{IsA_j}(n_i) \in q$. To incorporate the contribution of distinct node pairs n_i, n_k to $X(q)$, we set the edge weight $t(v_{ij}, v_{kl})$ to $\sum_{g \in G_{n_i,n_k}} w(g) \cdot g(\{\mathsf{IsA_j}(n_i), \mathsf{IsA_l}(n_k)\})$. The edge (v_{ij}, v_{kl}) thus has a weight equal to the sum of the weights of all the true ground formulas containing n_i and n_k when n_i is assigned the label A_j and n_k is assigned the label A_l.

Now, for query predicates q, consider the subgraph of $\mathcal{G}$ containing vertices v_{ij} for each predicate $\mathsf{IsA_j}(n_i) \in q$. It is easy to see that the sum of the weights of vertices and edges in this subgraph induced by q is equal to $X(q)$. Thus, the problem of computing a label assignment q that maximizes $X(q)$ is equivalent to finding the maximum weight subgraph in $\mathcal{G}$.

Example 4.1. *Consider a web site with a single page p containing two nodes n_1 and n_2. Nodes n_1 and n_2 contain the text strings "Java City" and "51 Lavelle road", respectively. The attributes we want to extract are A_1 =* Name *and A_2 =* $\mathsf{Address}$. *We will use attribute A_3 to represent* Noise.

The following two content formulas have non-zero weights of 1: (1) $\forall n\ \mathsf{ContainsWordRoad}(n) \Rightarrow \mathsf{IsAddress}(n)$ *and (2)* $\forall n\ \mathsf{ContainsWordsBeginCaps}(n) \Rightarrow \mathsf{IsName}(n)$. *In addition, the following precedence structural formula also has a weight of 1:* $\forall n_1, n_2 \in p\ \mathsf{IsName}(n_1) \wedge \mathsf{IsAddress}(n_2) \Rightarrow \mathsf{Precedes}(n_1, n_2)$. *The set of true evidence predicates x is* $\{\mathsf{ContainsWordRoad}(n_2), \mathsf{ContainsWordsBeginCaps}(n_1), \mathsf{Precedes}(n_1, n_2)\}$.

The graph $\mathcal{G}$ for the web site page contains the 6 vertices shown in Figure 1(a). There are a pair of vertices v_{1j} and v_{2j} corresponding to nodes n_1 and n_2 for each attribute A_j (including Noise*). The vertex weights are shown inside the circles and the edges weights are shown adjacent to the edges. For example, the ground formulas* $\mathsf{ContainsWordRoad}(n_1) \Rightarrow$

$\mathsf{IsAddress}(n_1)$ *and* $\mathsf{ContainsWordsBeginCaps}(n_1) \Rightarrow \mathsf{IsName}(n_1)$ *are both true if node n_1 is assigned the label* Name*, but the second ground formula is false if n_1 is assigned the labels* Address *or* Noise*. So the weight for vertices v_{11}, v_{12}, and v_{13} are set to 2, 1 and 1, respectively.*

Now, the node pair n_1, n_2 is contained in two ground formulas: $\mathsf{IsName}(n_1) \wedge \mathsf{IsAddress}(n_2) \Rightarrow \mathsf{Precedes}(n_1, n_2)$ *and* $\mathsf{IsName}(n_2) \wedge \mathsf{IsAddress}(n_1) \Rightarrow \mathsf{Precedes}(n_2, n_1)$*. For vertices v_{11} and v_{22}, both ground formulas are true when n_1 is assigned the label* Name *and n_2 is assigned the label* Address*. Thus, edge (v_{11}, v_{22}) is assigned a weight of 2. However, for vertices v_{12} and v_{21}, the second ground formula is false when n_1 is assigned the label* Address *and n_2 is assigned the label* Name *since n_1 precedes n_2. Thus, edge (v_{12}, v_{21}) is assigned a weight of 1.* □

4.1.2 Eliminating the Noise Layer

In any label assignment, a majority of the nodes will be labeled as Noise. So we can improve the efficiency of our inference algorithm by considering Noise as the "default" label, and only computing the non-noise label assignments. In this subsection, we describe an optimization for reducing the size of graph $\mathcal{G}$ by eliminating Noise vertices from it. This helps to speed up our inference algorithm since it needs to examine fewer vertices and compute a much smaller subgraph.

Let $\hat{q}$ be a non-noise query predicate set that assigns non-noise labels to a subset of nodes. Further, let $q \supset \hat{q}$ be the label assignment that assigns Noise to the remaining nodes. (Note that $\hat{q}$ will in general be a lot smaller than q.) Now, consider an assignment where all nodes are assigned the Noise label, and let X_{noise} be the objective function value for this assignment. When some of the labels are changed to non-noise labels in assignment $\hat{q}$, then the objective function value changes by $\Delta X(\hat{q}) = X(q) - X_{noise}$. Thus, if we find the non-noise label assignment $\hat{q}$ that maximizes $\Delta X(\hat{q})$, then the corresponding label assignment q that augments $\hat{q}$ with Noise labels will also be optimal with maximum $X(q)$.

We show that finding the non-noise label assignment $\hat{q}$ that maximizes $\Delta X(\hat{q})$ is equivalent to finding the maximum weight subgraph within a new graph $\hat{\mathcal{G}}$. Graph $\hat{\mathcal{G}}$ is identical to $\mathcal{G}$ except for the following:

- $\hat{\mathcal{G}}$ does not contain vertices from the Noise layer. Thus, graph $\hat{\mathcal{G}}$ only has $|\mathbf{A}| - 1$ layers.
- For each vertex $v_{ij} \in \hat{\mathcal{G}}$, the weight $\hat{t}(v_{ij})$ is set to $t(v_{ij}) - \sum_{g \in G_{n_i}} w(g) \cdot g(\{\mathsf{IsNoise}(n_i)\})$. Thus, we reduce the weight of vertex v_{ij} in $\mathcal{G}$ by the weight of the true groundings in G_{n_i} when n_i is labeled Noise.
- For each edge $(v_{ij}, v_{kl}) \in \hat{\mathcal{G}}$, the weight $\hat{t}(v_{ij}, v_{kl})$ is set to $t(v_{ij}, v_{kl}) - \sum_{g \in G_{n_i, n_k}} w(g) \cdot g(\{\mathsf{IsNoise}(n_i), \mathsf{IsNoise}(n_k)\})$. Thus, we reduce the weight of edge (v_{ij}, v_{kl}) in $\mathcal{G}$ by the weight of the true groundings in G_{n_i,n_k} when n_i and n_k are both labeled Noise.

Now for a non-noise label assignment $\hat{q}$, consider the vertices v_{ij} in $\hat{\mathcal{G}}$ corresponding to predicates $\mathsf{IsA_j}(n_i)$ in $\hat{q}$. It is easy to see that the subgraph induced by the vertices has weight $\Delta X(\hat{q})$. Thus, finding a label assignment $\hat{q}$ that maximizes $\Delta X(\hat{q})$ is equivalent to finding a maximum weight subgraph in $\hat{\mathcal{G}}$. This is a much smaller subgraph containing only the nodes that are assigned non-noise labels, and so is more efficient to compute. The optimal label assignment q corresponding to the subgraph then contains the predicate $\mathsf{IsA_j}(n_i)$ for every vertex v_{ij} in the subgraph, and the predicate $\mathsf{IsNoise}(n_i)$ for nodes n_i that don't have a corresponding vertex in the subgraph.

Example 4.2. *Figure 1(b) depicts the graph $\hat{\mathcal{G}}$ for the page with nodes n_1 and n_2, and the content and structural formulas described in Example 4.1. First, observe that the graph contains only 4 vertices; the two vertices in the* Noise *layer in $\mathcal{G}$ are not present in $\hat{\mathcal{G}}$. Furthermore, the weight of each vertex v_{ij} is set to $\hat{t}(v_{ij}) = t(v_{ij}) - t(v_{i3})$ in $\hat{\mathcal{G}}$. For example, the weight of vertex v_{11} is set to $t(v_{11}) - t(v_{13}) = 2 - 1 = 1$. Similarly, the weight of each edge (v_{ij}, v_{kl}) is set to $\hat{t}(v_{ij}, v_{kl}) = t(v_{ij}, v_{kl}) - t(v_{i3}, v_{k3})$. So for instance, the weight of edge (v_{11}, v_{22}) is set to $t(v_{11}, v_{22}) - t(v_{13}, v_{23}) = 2 - 2 = 0$.*

The maximum weight subgraph in $\hat{\mathcal{G}}$ contains the vertices v_{11} and v_{22} with a total weight of 2 (each vertex has a weight 1). This corresponds to assigning label Name *to node n_1 and* Address *to n_2.* □

4.1.3 Enforcing Hard Constraints

We model the single label hard constraint by assigning a weight of $-\infty$ to the edge between vertices v_{ij} and v_{il} in $\hat{\mathcal{G}}$ corresponding to the same node n_i, but in different layers j and l. Furthermore, we ensure that the mandatory attribute constraint is satisfied by requiring that the maximum weight subgraph contain at least one vertex from each mandatory attribute layer.

Finally, we incorporate contiguity constraints into our graph framework by considering all possible contiguous vertex subsequences within each non-noise attribute layer of a page, and coalescing each subsequence into a single *super-vertex*. So in $\hat{\mathcal{G}}$, if $v_{1j}, v_{2j}, \ldots, v_{mj}$ is the sequence of vertices belonging to layer j of a page, then for each contiguous subsequence $v_{ij}, \ldots, v_{kj}$ we add a super-vertex to layer j in $\hat{\mathcal{G}}$. Each super-vertex has a weight equal to the sum of the weights $\hat{t}(v_{ij})$ of all the vertices v_{ij} that it contains, and the weight of an edge between two super-vertices (in different layers) is equal to the sum of the weights of all the edges between the vertices comprising the two super-vertices. Note that the edge weight is $-\infty$ between super-vertices in different layers but whose corresponding node sets overlap, and so the single label hard constraint is preserved.

Now, we can enforce the contiguity constraint by assigning a weight of $-\infty$ to edges between (super-)vertices v_{ij} and v_{kj} belonging to the same non-noise layer j within a page. This ensures that only the sequence of nodes corresponding to a single super-vertex in layer j are all assigned the same label A_j. It is easy to see that the maximum weight subgraph of $\hat{\mathcal{G}}$ with super-vertices corresponds to an optimal non-noise label assignment $\hat{q}$ that maximizes $\Delta X(\hat{q})$ while satisfying the single label and contiguity hard constraints.

We can reduce the number of super-vertices in layer j of graph $\hat{\mathcal{G}}$ by pruning the super-vertices with low weights. In our experiments, we found that our simple pruning procedure is very effective and reduces the number of super-vertices by almost 90%.

4.2 A Greedy Heuristic

Algorithm 1 describes a greedy algorithm to find the maximum weight subgraph S in $\hat{\mathcal{G}}$. In the first phase, the algorithm greedily adds vertices to S as long as the weight of the

Algorithm 1 Maximum weight subgraph computation.

```
Input: Graph $\hat{\mathcal{G}}$.
Output: Maximum weight subgraph of $\hat{\mathcal{G}}$.

$S = \emptyset$;
repeat
    $S_{old} = S$;
    Let $v_{ij}$ be the vertex for whom $\hat{t}(S \cup \{v_{ij}\})$ is maximum;
    if $\hat{t}(S \cup \{v_{ij}\}) > \hat{t}(S)$ then $S = S \cup \{v_{ij}\}$;
until $\hat{t}(S) = \hat{t}(S_{old})$
repeat
    $S_{old} = S$;
    Consider a random ordering of (page, attribute) pairs;
    for each $(p_r, A_j)$ pair in the ordering do
        if $S$ contains a vertex $v_{i,j}$ from layer $j$ of page $p_r$ then
            $S' = S - \{v_{ij}\}$;
            Let $v_{kj}$ be the vertex in layer $j$ of page $p_r$ for whom $\hat{t}(S' \cup \{v_{kj}\})$ is maximum;
            if $\hat{t}(S' \cup \{v_{kj}\}) > \hat{t}(S)$ then $S = S' \cup \{v_{kj}\}$;
            if $\hat{t}(S') > \hat{t}(S)$ and $A_j$ is non-mandatory then $S = S'$;
        else
            Let $v_{ij}$ be the node in layer $j$ of page $p_r$ for whom $\hat{t}(S \cup \{v_{ij}\})$ is maximum;
            if $\hat{t}(S \cup \{v_{ij}\}) > \hat{t}(S)$ or $A_j$ is mandatory then
                $S = S \cup \{v_{ij}\}$;
        end if
    end for
until $\hat{t}(S) - \hat{t}(S_{old}) < \epsilon$
return $S$;
```

subgraph S, $\hat{t}(S)$, continues to increase. In each iteration, the vertex v_{ij} that leads to the biggest increase in $\hat{t}(S)$ is added to S. Note that for efficiency, for all vertices v_{ij} in $\hat{\mathcal{G}}$, we can incrementally keep track of $\hat{t}(S \cup \{v_{ij}\})$ as S is continually updated. Thus, each iteration of the first phase has a time complexity that is linear in the number of vertices in $\hat{\mathcal{G}}$.

In the second phase, we iterate over multiple cycles to insert, delete, or replace vertices in S to further increase its weight. In each iteration, we select attribute layers j within pages at random, and look to increase $\hat{t}(S)$ by either swapping or deleting a vertex v_{ij} currently present in S, or inserting a new vertex from the layer j into S. Note that we consider adding a vertex from a mandatory attribute layer to S even if it does not improve the weight of subgraph S. Now, S can contain at most $(|\mathbf{A}|-1)$ vertices per page, one vertex from each (non-noise) attribute layer. Thus, $\hat{t}(S \cup \{v_{ij}\})$ can be computed from $\hat{t}(S)$ in time proportional to $|S|$, and the time complexity of each iteration in the second phase is proportional to the number of vertices in $\hat{\mathcal{G}}$.

At the end of each iteration, the algorithm terminates if there is no significant improvement in $\hat{t}(S)$, that is, if the improvement is less than a user-defined threshold parameter ϵ. Finally, nodes n_i with corresponding vertices v_{ij} in S are labeled with the attribute A_j; the remaining nodes are labeled as Noise.

5. EXPERIMENTAL EVALUATION

In this section, we report experimental results with real-life web datasets which demonstrate the effectiveness of our MLN model with graph-based inference. We compare the running time of our inference algorithm with the standard MaxWalkSat (MWS) algorithm [7]. From a modeling perspective, we also compare the labeling accuracy of our MLN model with the hierarchical CRF (HCRF) model [19] which represents the state-of-the-art in web data extraction.

5.1 Experimental Setup

Datasets: We constructed two datasets Restaurants and Books as follows. We picked random web pages that provide detailed information about restaurants from www.citysearch.com, www.frommers.com, www.nymag.com and www.superpages.com, and about books from www.christianbooks.com, www.fishpond.com, www.booksamillion.com, www.libertybooks.com and www.valorebooks.com. The approximate number of pages we collected per site were 100 and 500 for Restaurants and Books, respectively. The number of nodes per page was typically a few hundred. We considered the non-noise attributes Name, Address, Phone, Timing, Review, and Description in the Restaurants dataset. Of these attributes, Review and Description were not mandatory. In the Books case, we considered Name, Price, ISBN10 (10 digits), ISBN13 (13 digits), Date, NumberOfPages, and Description as non-noise attributes. Of these, only Name and Price were mandatory.

Extraction Models: We considered the following extraction models in our experiments: (1) MLN with our graph-based inference, (2) MLN with MWS inference, and (3) HCRF. To build and test the various models, we annotated the nodes in the DOM tree representation of the web pages with attribute labels. To learn the MLN model, we made use of the publicly available package *thebeast* (Markov logic and statistical relational learning software) available at http://code.google.com/p/thebeast/ (see also [13]). This package provides a MIRA type learning algorithm and supports the MWS inference algorithm. We integrated our graph-based inference algorithm (described in Section 4) into this package. Note that the MIRA algorithm is dependent on the inference algorithm to learn formula weights.

The content and structural predicates and formulas for our MLN model are defined as discussed in Section 3. Content predicates are attribute-specific; we give a few examples of content predicates for each of the attributes below.

Books:

•Name: FirstLetterCapital, FontSizeLarge, PageTitleOverlap. The last predicate fires if the text content in the node overlaps with the title of the page.

•Price: ContainsNumber, ContainsPriceSymbol.

•ISBN10, ISBN13: Has10Digits, Has13Digits.

•NumberOfPages: Has3or4Digits.

•Date: Has4Digits.

•Description: In this case, a dictionary is constructed using the textual content of labeled nodes. Since noisy nodes also contain different words, the dictionary words are selected using the mutual information criterion. Then a dictionary based predicate ContainsDescriptionDictWord is defined. The predicate fires if the percentage of words within a node that are present in the dictionary exceeds a certain threshold (say, 10%). Another useful predicate LargeText is defined based on the observation that descriptions are often lengthy. Therefore this predicate fires when the number of words in the text content exceeds a certain threshold.

Restaurants:

•Name: Similar to Name in Books.

•Address: ContainsWordRoad, ContainsZipCode, ContainsStateName. The last predicate is implemented using state name dictionaries.

•Phone: Has9or10Digits, ContainsNumberinBracket.

•Timing: Has1or2Digit, ContainsDay, ContainsAMPM.

•Reviews, Description: Similar to Description in Books.

To build the HCRF model, we constructed the visual

Attribute	C	C + S (Intra)	C + S (All)	MWS (WOC)
Name	**82.2**	**82.2**	**82.2**	55.4
Price	53.6	91.0	**95.6**	49.2
Date	**99.0**	**99.0**	98.0	96.3
NumberOfPages	**88.0**	**88.0**	87.0	43.7
ISBN10	99.8	**100.0**	99.8	80.8
ISBN13	**97.8**	**97.8**	**97.8**	72.3
Description	58.3	57.5	**58.8**	53.0
Avg-F1	82.7	87.9	**88.4**	64.4
Name	**94.9**	**94.9**	94.6	83.7
Address	**96.8**	95.6	96.7	66.8
Phone	91.9	95.9	**96.6**	92.4
Timing	70.8	91.3	**92.6**	76.9
Description	66.3	65.9	**73.1**	53.0
Review	**90.2**	90.2	86.8	39.7
Avg-F1	85.2	89.0	**90.1**	68.7

Table 1: F1 scores of different methods on the Books and Restaurants datasets.

tree representation [19] of the web pages and annotated the nodes. We implemented a gradient based learning algorithm and a loopy belief propagation algorithm [17] for marginal probability computation and inference. In the HCRF model, we used equivalent content features as our MLN model but structural features were restricted to only adjacent siblings and parent-child nodes in the tree.

Evaluation Metrics: All the experiments were performed in a leave-one-out (LOO) setting. Here, assume that there are labeled examples available from K sites. We then evaluate the standard precision/recall/F1-score performance measures on the i^{th} site using the labeled examples from the remaining sites as training data. We repeat this $\forall i = 1, \ldots, K$, and report the average performance over the K sites.

Platform: All the experiments were performed on a Linux PC with 4 Intel Xeon 2.33 GHz (dual core) processors and 4GB of RAM.

5.2 Experimental Results

Effect of Content and Structural Features: The first three columns of Table 1 depict the F1 values for the MLN model using our graph-based inference algorithm. To measure the efficacy of the different types of features, we consider three cases. In the first case, we use only content (C) formulas. This implies that there are only $-\infty$ weight edges in the graph for modeling the hard constraints. In the second case, we use both content formulas and *intra-page proximity* and *precedence* structural (S) formulas (denoted as C+S (Intra)). In both these cases, each training and inference instance is a single web page. In the last case, we also include *inter-page alignment* structural formulas (denoted as C+S (All)). In this case, we perform training and inference on groups of three pages. Note that the graph contains additional edges in the models with structural formulas. Thus the model (number of formulas and weights) and the corresponding graph becomes increasingly complex as we go from the first to the third case.

From the first three columns in Table 1 we observe that the F1 performance improves for several attributes as we go from the model with only content formulas to the model that includes intra-page and inter-page structural formulas. This improvement is significant for attributes like Price in the Books dataset, and Phone, Timing and Description attributes in the Restaurants dataset. Essentially structural formulas

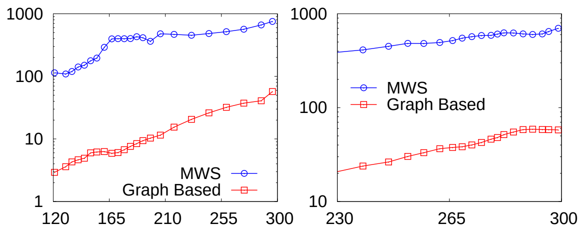

Figure 2: Inference time in seconds versus Number of nodes for the case of C+S (All) on the Books (left) and Restaurants (right) datasets.

help to correct wrong node labels obtained using only content formulas. For example, in many web pages, there are multiple instances of price like list price, discounted price, savings, etc. in the page. With only content formulas, all instances will be labeled as Price. Structural information enables us to distinguish between the various instances, for e.g., by selecting the occurrence of Price that is closest to the Name attribute. Observe that although some performance degradation is seen in attributes like Review and NumberOfPages with the inclusion of inter-page formulas, the average F1 scores are higher when structural formulas are present.

Comparison with MWS (Time): As efficient inference is important during training as well as testing, we measured the inference time taken by the MWS and our graph-based inference algorithms. In Figure 2, we plot the running times for page groups containing different numbers of nodes. The time taken by the MWS algorithm is dependent on the number of random flips. To ensure a certain minimum level of labeling accuracy, we set this number to 20000 and 1000 for the Books and Restaurants datasets, respectively. It is easy to see that our algorithm is an order of magnitude faster. Of course, to obtain labeling accuracy that is competitive to our algorithm, the number of flips has to be increased to a larger value resulting in an even higher speed difference between the algorithms.

Training involves running multiple passes of inference over the training set and we found 10 passes were sufficient. The MWS inference algorithm was extremely slow when contiguity constraints were incorporated. It took more than a day for one pass over the training set to complete for some sites. Also, it ran out of memory in later passes. One reason for the problem is that since contiguity constraints involve node triples, the number of ground formulas becomes very large. Also, randomly flipping the labels to improve the objective function results in slower convergence.

Comparison with MWS (Labeling Accuracy): Due to the reasons mentioned above, we could not obtain labeling accuracy results for the MWS algorithm with contiguity constraints. Therefore, the MWS results reported in the final column of Table 1 are for the case of C+S (Intra) and without contiguity constraints (WOC), with the number of random flips set to 80000 which gave the maximum labeling accuracy. Although the MWS algorithm ran fast in this case, its F1 performance was poor. This is because in the absence of contiguity constraints, two types of errors occur. For attributes like Address, Timing, Review, and Description which can span more than one node and occur contiguously, *recall* is affected when some intermediate nodes are wrongly labeled. On the other hand, when an attribute like Address occurs multiple times in a non-contiguous manner within a restaurant page, *precision* is affected when several of them

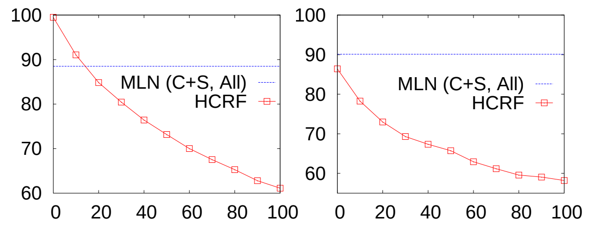

Figure 3: % of pages from unseen websites included in test set versus Average F1 (%) on the Books (left) and Restaurants (right) datasets.

(e.g., for related restaurants) get labeled with the same attribute and only one of them is relevant. This is the reason for the low precision and F1 values for attributes like Name, Price, and NumberOfPages which typically do not require contiguity constraints.

In contrast, our graph-based inference algorithm is efficient even with contiguity constraints because of supervertices, and it outperforms MWS on F1 scores as well.

Comparison With HCRF: When we evaluated the HCRF model in the LOO setting we observed that its performance on both the datasets was quite poor. We will give explanations for this behavior shortly. To understand whether the problem is poor generalization on unseen sites, we conducted an experiment in a modified LOO setting as follows. Let $\mathbf{D}_i = (D_{tr}^{(i)}, D_{tst}^{(i)})$ denote the train and test split of the data of the i-th web site $W_i, i = 1, \ldots, K$. We kept the ratio of the train and test split at 60:40. For each i, we first constructed the training set as $\cup_{j \neq i} D_{tr}^{(j)}$ and built the HCRF model using this set. Then we evaluated the performance on the set $\cup_{j \neq i} D_{tst}^{(j)}$ and this set corresponds to the leftmost x-axis point in Figure 3. Next we progressively *expanded* this test set by adding a fixed percentage of the examples from the set $\mathbf{D}_i$. Thus the rightmost x-axis point in the figure corresponds to the inclusion of all the examples from the unseen set $\mathbf{D}_i$.

Due to space limitations, we report the average F1 score (averaged over the sites and attributes) in Figure 3. From the figure, it is clearly seen that the performance of the HCRF model is quite good to start with. Note that this starting point is equivalent to evaluating the performance within the same sites and is the procedure used in [19]. However, as more examples from the unseen site get added to the test set the performance starts degrading. This demonstrates that while the HCRF model generalizes well within the sites, its generalization is not good on unseen sites. In contrast, the MLN model with structural formulas and contiguity constraints generalizes much better to unseen websites. For reference, we have indicated the performance of MLN (C+S, All) from Table 1 as horizontal lines in Figure 3. Note that, the MLN performance should be considered a lower bound since it was obtained under the more stringent across website LOO setting and adding pages from seen websites in the test set would only improve its performance.

Compared to the HCRF modeling framework, the MLN model with our inference algorithm makes use of domain knowledge quite effectively in the form of (1) hard constraints and (2) structural formulas. This is the key reason for its superior performance. In our experiments, we have observed that the absence of contiguity constraints hurts the performance of the HCRF model. Also, the absence of mandatory constraints (for some attributes) in the HCRF model may result in poor recall with none of the nodes getting labeled for those attributes. Finally, structural formulas like *precedence* used in the MLN model capture long-range dependencies among attributes much better than the HCRF model. This helps the MLN model to cope with noise and generalize well even in the LOO setting.

6. RELATED WORK

Existing approaches for attribute extraction from web pages rely on *page structure* and *attribute content* to varying degrees. (Comprehensive surveys of existing web extraction techniques can be found in [3, 14].) One of the early methods, *wrapper induction* [8, 10], utilizes manually labeled data to learn extraction rules on the HTML tag structure of pages. Unfortunately, wrappers require human editors to annotate example pages from each site, and are thus impractical for thousands of sites. Unsupervised approaches like RoadRunner [5], DEPTA [18] and tag path clustering [9] eliminate human overhead by looking for repetitive patterns in the HTML tags of a page to extract records from the page. However, all of the above unsupervised methods do not associate attribute labels with HTML elements which is the focus of our work.

Among recently proposed statistical extraction approaches [2, 1, 6, 19, 16], [19, 16] address the problem of extracting structured data from web pages. However, as mentioned in the previous section, a limitation of HCRFs [19] is that they cannot model long-range dependencies like precedence, proximity, alignment, etc. between non-adjacent elements in web pages. Also, our work differs from [16] in two key aspects. First, we consider the web data extraction problem in its full generality, and leverage new structural features like precedence, proximity, contiguity, and a more general version of alignment that have not been considered before by statistical extraction methods. Second, we develop a novel graph-based inference procedure for MLNs that is tailored to our extraction setting and is much faster than existing inference algorithms that rely on generic satisfiability solvers.

Singla and Domingos [15] propose an optimization to reduce memory requirements of satisfiability solvers by lazily grounding clauses. Poon and Domingos [11] exploit the lazy grounding idea to reduce both memory requirement and time of a Markov chain based inference algorithm (MC-SAT). Riedel [13] proposes a cutting plane inference algorithm (related to [11]) that can use any MAP inference algorithm like MaxWalkSat as a base solver. However, none of these works alter the execution logic of satisfiability solvers.

7. CONCLUSION

In this paper, we proposed an MLN-based approach for extracting structured data from web pages. We showed that the inference algorithm can be posed as a maximum weight subgraph problem and presented an efficient greedy algorithm to improve scalability. Experimental results with real-life datasets clearly demonstrate that (1) our graph-based inference algorithm runs significantly faster than the standard MWS algorithm and (2) our MLN model delivers higher labeling accuracy than the state-of-the-art HCRF model. An interesting direction for future work is to develop MLN models and fast inference algorithms for other extraction problems like extracting (multiple) records from list pages, and joint segmentation and attribute extraction from pages.

8. REFERENCES

[1] E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *ACM SIGKDD*, 2004.

[2] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In *ACM SIGMOD*, 2001.

[3] C.-H. Chang, M. Kayed, M. R. Girgis, and K. Shaalan. A survey of web information extraction systems. *IEEE transactions on KDE*, 18:1411–1428, 2006.

[4] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

[5] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, 2001.

[6] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstrcutured lists on the web. In *VLDB*, 2009.

[7] H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. In *The satisfiability problem: theory and applications. AMS*, 1997.

[8] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *IJCAI*, 1997.

[9] G. Miao, J. Tatemura, W. Hsiung, A. Sawires, and L. Moser. Extracting data records from the web using tag path clustering. In *WWW*, 2009.

[10] I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 1(2), 2001.

[11] H. Poon, P. Domingos, and M. Sumner. A general method for reducing the complexity of relational inference and its application to mcmc. In *AAAI*, pages 1075–1080, 2008.

[12] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[13] S. Riedel. Improving the accuracy and efficiency of map inference for Markov logic. In *UAI*, 2008.

[14] S. Sarawagi. Information extraction. *Foundations and trends in databases*, 1(3):261–377, 2008.

[15] P. Singla and P. Domingos. Memory-efficient inference in relational domains. In *21st NCAI*, 2006.

[16] J.-M. Yang, R. Cai, Y. Wang, J. Zhu, L. Zhang, and W.-Y. Ma. Incorporating site-level knowledge to extract structured data from web forums. In *WWW*, 2009.

[17] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, 2000.

[18] Y. Zhai and B. Liu. Web data extraction based on partial tree assignment. In *WWW*, 2005.

[19] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. Simultaneous record detection and attribute labeling in web data extraction. In *ACM SIGKDD*, 2006.