Learning to extract information from large domain-specific websites using sequential models

Sunita Sarawagi
sunita@iitb.ac.in

V.G.Vinod Vydiswaran
vgvinodv@iitb.ac.in

ABSTRACT

In this article we describe a novel information extraction task on the web and show how it can be solved effectively using the emerging conditional exponential models. The task involves learning to find specific goal pages on large domain-specific websites. An example of such a task is to find computer science publications starting from university root pages. We encode this as a sequential labeling problem solved using Conditional Random Fields (CRFs). These models enable us to exploit a wide variety of features including keywords and patterns extracted from and around hyperlinks and HTML pages, dependency among labels of adjacent pages, and existing databases of named entities in a unified probabilistic framework. This is an important advantage over previous rule-based or generative models for tackling the challenges of diversity on web data.

1. INTRODUCTION

The World Wide Web continues to drive industry and researchers alike to new ways of information gathering and search that go beyond bulk crawling and keyword search. A steady, but not so populist, line of work has been on imposing some form of structure on the largely unstructured web, and then providing a richer search on the extracted structures. Given the scale and diversity of the web, the dream of a full-fledged semantic embedding of the web is far from reality. However, it is possible to design task-oriented extraction primitives that are solvable with current technology and deployable immediately. This has driven a lot of work on web wrapper extraction in the past. However, most previous wrapper systems have been successful only on fairly stylized machine-generated documents. In this article we present a new extraction task that pushes the envelope in this line of work by handling more heterogeneous sources, albeit from a limited domain.

Often websites within a domain are structurally similar to each other. Humans are good at navigating these websites to reach specific information within it. Our goal is to learn the navigation path by observing the user's clicks on a few example websites and then use the learnt model to automatically reach the goal pages using as few redundant page fetches as possible. We start from a listing of related websites and after watching the user find the specific information from a few websites in the list, we automate the search to the goal pages in the remaining.

We present a scenario where such a capability would be useful. Citation portals like Citeseer need to gather publications on a particular discipline from homepages of faculty starting from lists of universities easily obtained from web directories like Dmoz. This requires following a path starting from the root page of the university, to the homepages of departments relevant to the discipline, from there visiting the homepages of faculty members, and then searching for links such as "Papers", "Publications", or "Research Interests" that lead to the publications page, if it exists. Several university websites follow this template, although there is a lot of variety in the details of exact words on pages and anchors and linkage patterns. We expect such a learning based approach to still capture the main structure in a few examples so as to automatically gather all faculty publications from any given list of universities without fetching too many superfluous pages.

Other scenarios where similar path learning problems arise are, extracting contact addresses of companies in a list, extracting computer science talk announcements from university web pages, and finding prices of specific products in a category hierarchy from electronic stores.

1.1 Possible approaches and Related work

We first discuss possible approaches that are based on existing technology and research work.

Keyword search. One approach is to find a set of keywords that can be used in conjunction with a search engine with the domain restricted to each website to get to the right pages. The set of keywords can be selected using a classifier trained to discriminate the goal pages from the non-goal pages. The search engine results can be re-ranked using a more detailed classifier to measure relevance to the goal pages. We claim that this method cannot provide high accuracy for the simple reason that the goal page itself may not hold enough information to correctly identify it as the goal page. The path leading to the goal page is important too. We will need to consider text around the entire path leading to the goal page in order to decide if it is relevant or not. For example, consider the publications scenario explained earlier. If Citeseer wants to get all computer science publications starting from a university root page, then it is necessary to follow a path through computer science and related departments' homepages. A publication page on its own might be hard to classify as holding "computer science publications". Similarly, a list of publications on a course webpage of a computer science department may not qualify

as publications of faculty of that department.

Generic Focused crawlers. Focused crawlers[1] are designed to crawl all webpages on a topic specified through examples of pages related to the topic. The basic property that such crawlers exploit is that pages on a topic are often linked to each other. The crawler consists of two classifiers — a baseline classifier that trains on the page-tokens, and an apprentice that trains adaptively on link-tokens of newly discovered relevant pages to choose the best hyperlink out of the crawl frontier. The focused crawler selects the link to crawl next by first choosing pages on the crawl frontier that are relevant to the goal topic, and then selecting the most relevant link from the hyperlinks out of those pages.

A problem with this approach is that it cannot exploit patterns of pages along a route leading to a goal page. The only pattern it exploits is relevance to the specific topic whereas often pages on unrelated topics might consistently lead to topics of interest. A solution to this problem is proposed in [3] where a context graph keeps track of path patterns to a goal page. The goal documents are at level 0, all pages linking to these are at level 1, and so on up to a maximum level N. All documents marked by the same level are clubbed together as being of the same class. Each page is independently classified to one of the levels and has an associated value of the certainty of classification. Each level also has a score that is directly proportional to the proximity to the goal state. A threshold function combines classification certainty with queue scores to determine a priority order for page fetches. The paper does not provide details of how these scores and thresholds are determined.

Rennie and McCallum[10] propose a similar solution to learning the path leading to pages of interest. The main difference is that the set of pages over which the classifiers are trained is fixed in advance whereas in the above context graph approach, the classifier is trained on-the-fly with training labels obtained via a backward crawling of pages linking to discovered goal pages. This paper provides a principled method of choosing among the classifiers at various layers using Q values derived from Reinforcement Learning.

While both these strategies provide a good way of recognizing early pages on a route leading to a goal page, the underlying classification framework does not exploit all useful correlation that exists among the pages in a path. Pages along a path are classified independently to the different levels totally ignoring the fact that if a page is assigned to level j then most likely all pages that it links to should be assigned level $j + 1$. We propose to use graphical models to exploit such correlations. Also, they rely on naive Bayes classifiers which are not capable of exploiting the variety of possibly correlated features available from the text and other properties of links and layouts of pages.

The problem we propose in this paper is different from all previous work on focused crawlers in that all focused crawlers have been developed for operating on the general web whereas we are trying to solve more of an information extraction problem from a given list of domain-specific websites like universities, companies and ecommerce websites. We exploit the regularity in the structures of websites in a given domain to build more powerful models than is possible in the case of general-purpose focused crawlers.

Web knowledge bases. One of the earliest projects that perform extraction of structured information from multi-page sources like a website is WebKB[2]. They describe similar learning tasks of recognizing relations by traversing paths through hyperlinks. However, their approach is based on generative classifiers (like naïve Bayes) for recognizing correct hits coupled with first order rules (like FOIL[4]) for finding the right page.

1.2 Problem statement

There are two phases to this task: first is the training phase, where the user teaches the system by clicking through pages showing some paths that end in goal pages and others that do not. The second phase is the foraging phase where the given list of websites are automatically navigated to find all goal pages while fetching as few redundant pages as possible. We consider two different modes of training.

1. The first is a fully supervised case where the user determines a set of milestone states that capture the main structural similarity across websites. As the user is clicking through to teach the system paths to follow, he labels pages with one of these milestones states. At the end of this process, we have a set of classes L, and a set of training paths where a subset of the pages in the path are labeled with a class from L.

2. The second is the partially supervised case where the user only indicates if the path ended in a goal state or not and does not provide any labels to intermittent pages.

1.3 Overview of our approach

We treat this as a sequence tagging problem where the path is a sequence of pages ending in a goal page. We first train a Conditional Random Field (CRF) to recognize such paths. There are several options for encoding the path to states of the CRF and depends among other things on the level of supervision provided by the user. We detail this in Section 3 after presenting a background of the basic CRF technology in Section 2. We then superimpose ideas from reinforcement learning to prioritize the order in which pages should be fetched to reach the goal page. This provides an elegant and unified mechanism of modeling the path learning and foraging problem.

2. BACKGROUND ON CRFS

A CRF models $\Pr(\mathbf{y}|\mathbf{x})$ as a Markov random field, with nodes corresponding to elements of the structured object $\mathbf{y}$, and potential functions that are conditional on (features of) $\mathbf{x}$. One common use of CRFs is for sequential learning problems like NP chunking[11], POS tagging[5], and named-entity recognition (NER)[8]. For these problems, the Markov field is a chain and $\mathbf{y}$ is a linear sequence of labels from a fixed set $\mathcal{Y}$, and the label at position i depends only on its previous label. For instance, in the NER application, where the task is to identify entity types like people names and organization in plain text, $\mathbf{x}$ might be a sequence of words, and $\mathbf{y}$ might be a sequence in $\{I, O\}^{|\mathbf{x}|}$, where $y_i = I$ indicates "word x_i is inside a name" and $y_i = O$ indicates the opposite.

CRFs have been shown to perform better than other sequential models like hidden Markov models that learn a joint

probability $\Pr(\boldsymbol{x}, \boldsymbol{y})$ of pairs of observation sequences $\boldsymbol{x}$ and label sequences $\boldsymbol{y}$. The parameters of the model are trained to maximize the joint likelihood of the training examples. A major shortcoming of generative models like HMMs is that they maximize the joint probability of sequence and labels. This does not necessarily maximize accuracy. Also, the conditional independence of features is a restrictive assumption. Conditional Random Fields learn a single global conditional model for $\Pr(\mathbf{y}|\mathbf{x})$ and have been found to achieve high accuracy in a number of applications.

Notation: We will use bold-faced symbols to denote vectors and non-bold faced symbols to denote scalars.

Assume a vector $\mathbf{f}$ of *local feature functions* $\mathbf{f} = \langle f^1, \ldots, f^K \rangle$, each of which maps a pair $(\mathbf{x}, \mathbf{y})$ and a position i in the vector $\mathbf{x}$ to a measurement $f^k(i, \mathbf{x}, \mathbf{y}) \in \mathbb{R}$. Let $\mathbf{f}(i, \mathbf{x}, \mathbf{y})$ be the vector of these measurements and let $\mathbf{F}(\mathbf{x}, \mathbf{y}) = \sum_i^{|\mathbf{x}|} \mathbf{f}(i, \mathbf{x}, \mathbf{y})$. For the case of NER, the components of $\mathbf{f}$ might include the measurement $f^{13}(i, \mathbf{x}, \mathbf{y}) = [\![x_i \text{ is capitalized}]\!] \cdot [\![y_i = I]\!]$, where the indicator function $[\![c]\!] = 1$ if c if true and 0 otherwise; this implies that $F^{13}(\mathbf{x}, \mathbf{y})$ would be the number of capitalized words paired with the label I.

For the sake of efficiency, we restrict any feature $f^k(i, \mathbf{x}, \mathbf{y})$ to be **local** in the sense that the feature at a position i will depend only on the previous labels. With a slight abuse of notation, we claim that a local feature $f^k(i, \mathbf{x}, \mathbf{y})$ can be expressed as $f^k(y_i, y_{i-1}, \mathbf{x}, i)$. Some subset of these features can be simplified further to depend only on the current state and are independent of the previous state. We will refer to these as **state features** and denote these by $f^k(y_i, \mathbf{x}, i)$ when we want to make the distinction explicit. The term **transition features** refers to the remaining features that are not independent of the previous state.

A Conditional Random Field (CRF)[5; 11] is an estimator of the form

$$\Pr(\mathbf{y}|\mathbf{x}, \mathbf{W}) = \frac{1}{Z(\mathbf{x})} e^{\mathbf{W} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y})} \tag{1}$$

where $\mathbf{W}$ is a weight vector over the components of $\mathbf{F}$ and the normalizing term $Z(\mathbf{x}) = \sum_{\mathbf{y}'} e^{\mathbf{W} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}')}$.

2.1 An efficient inference algorithm

The *inference problem* for a CRF is defined as follows: Given $\mathbf{W}$ and $\mathbf{x}$, find the best label sequence, $\arg\max_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}, \mathbf{W})$, where $\Pr(\mathbf{y}|\mathbf{x}, \mathbf{W})$ is defined by equation 1.

$$\begin{aligned} \arg\max_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}, \mathbf{W}) &= \arg\max_{\mathbf{y}} \mathbf{W} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}) \\ &= \arg\max_{\mathbf{y}} \mathbf{W} \cdot \sum_j \mathbf{f}(y_j, y_{j-1}, \mathbf{x}, j) \end{aligned}$$

An efficient inference algorithm is possible because all features are assumed to be local. Let $\mathbf{y}_{i:y}$ denote the set of all partial labels starting from 1 (the first index of the sequence) to i, such that the i-th label is y. Let $\delta(i, y)$ denote the largest value of $W \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}')$ for any $\mathbf{y}' \in \mathbf{y}_{i:y}$. The following recursive calculation implements the usual Viterbi algorithm[9]:

$$\delta(i, y) = \begin{cases} \max_{y'} \ \delta(i-1, y') + \mathbf{W} \cdot \mathbf{f}(y, y', \mathbf{x}, i) & \text{if } i > 0 \\ 0 & \text{if } i = 0 \end{cases} \tag{2}$$

The best label then corresponds to the path traced by $\max_y \delta(|\mathbf{x}|, y)$.

2.2 Training algorithm

Learning is performed by setting parameters to maximize the likelihood of a set of a training set $T = \{(\mathbf{x}_\ell, \mathbf{y}_\ell)\}_{\ell=1}^N$ expressed in logarithmic terms as

$$L(\mathbf{W}) = \sum_\ell \log \Pr(\mathbf{y}_\ell|\mathbf{x}_\ell, \mathbf{W}) = \sum_\ell (\mathbf{W} \cdot \mathbf{F}(\mathbf{x}_\ell, \mathbf{y}_\ell) - \log Z_{\mathbf{W}}(\mathbf{x}_\ell))$$

We wish to find a $\mathbf{W}$ that maximizes $L(\mathbf{W})$. The above equation is convex and can thus be maximized by gradient ascent or one of many related methods. (In our implementation, we use a limited-memory quasi-Newton method[6; 7].) The gradient of $L(\mathbf{W})$ is the following:

$$\begin{aligned} \nabla L(\mathbf{W}) &= \sum_\ell \mathbf{F}(\mathbf{x}_\ell, \mathbf{y}_\ell) - \frac{\sum_{\mathbf{y}'} \mathbf{F}(\mathbf{x}_\ell, \mathbf{y}') e^{\mathbf{W} \cdot \mathbf{F}(\mathbf{x}_\ell, \mathbf{y}')}}{Z_{\mathbf{W}}(\mathbf{x}_\ell)} \\ &= \sum_\ell \mathbf{F}(\mathbf{x}_\ell, \mathbf{y}_\ell) - E_{\Pr(\mathbf{y}'|\mathbf{W})} \mathbf{F}(\mathbf{x}_\ell, \mathbf{y}') \end{aligned}$$

The first set of terms are easy to compute. However, we must use the Markov property of $\mathbf{F}$ and a dynamic programming step to compute the normalizer $Z_{\mathbf{W}}(\mathbf{x}_\ell)$, and the expected value of the features under the current weight vector, $E_{\Pr(\mathbf{y}'|\mathbf{W})} \mathbf{F}(\mathbf{x}_\ell, \mathbf{y}')$. Details of computing these can be found be in [11].

3. OUR APPROACH

We first describe the model training phase where the user provided example positive and negative paths from a few websites are used to train a CRF model. We then describe how this trained model is used to locate goal pages starting from root pages of other websites.

3.1 Model training

During training, we are given examples of several paths of labeled pages where some of the paths end in goal pages and others end with a special "fail" label. We first consider the fully supervised case where the user provides milestone states and later consider the partially supervised case where no intermittent labels are provided.

3.1.1 Supervised case

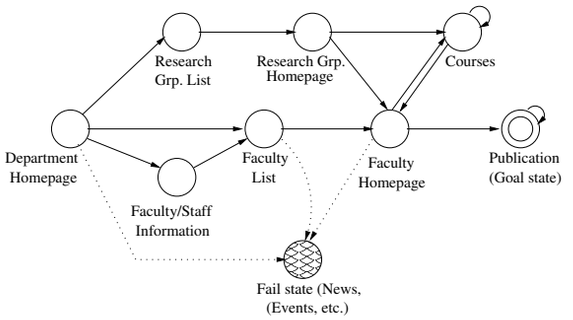
We can treat each path as a sequence of pages denoted by the vector $\mathbf{x}$ and their corresponding milestone labels denoted by $\mathbf{y}$. Each x_i is a webpage represented suitably in terms of features derived from the words in the page, its URL, and anchor text in the link pointing to x_i.

A number of design decisions about the label space and feature space need to be made in constructing a CRF to recognize characteristics of valid paths. One option is to assign a state to each possible label in the set L which consists of the milestone labels and two special labels "goal" and "fail". An example of such a model for the publications scenario is given in Figure 1(a) where each circle represents a label.

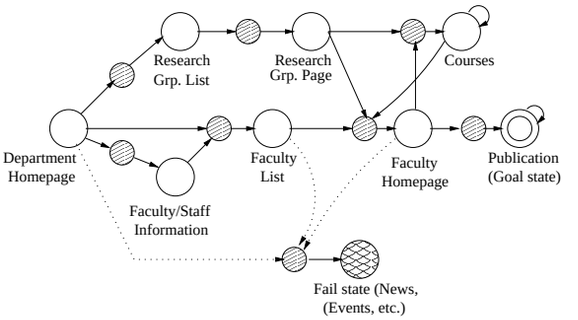
State features are defined on the words or other properties comprising a page. For example, state features derived from words are of the form

$$f^k(i, \mathbf{x}, y_i) = [\![x_i \text{ is "computer" and } y_i = \text{faculty}]\!]$$

. The URL of a page also yields valuable features. For example, a "tilda" in the URL is strongly associated with a personal home page and a link name with word "contact" is strongly associated with an address page. We tokenize each



(a) One state per label with links as transitions



(b) A state for each label and each link

Figure 1: State transition diagram for the Publications domain

URL on delimiters and add a feature corresponding to each token.

Transition features capture the soft precedence order among labels. One set of transition features are of the form: $f^k(i, \mathbf{x}, y_i, y_{i-1}) = [\![y_i \text{ is "faculty" and } y_{i-1} \text{ is "department"}]\!]$. They are independent of x_i and are called **edge features** since they capture dependency among adjacent labels. In this model transition features are also derived from the words in and around the anchor text surrounding the link leading to the next state. Thus, a transition feature could be of the form $f^k(i, \mathbf{x}, y_i, y_{i-1}) = [\![x_i \text{ is an anchor word "advisor", } y_i \text{ is "faculty" and } y_{i-1} \text{ is "student"}]\!]$.

A second option is to model each given label as a dual-state — one for the characteristics of the page itself (**page-states**) and the other for the information around links that lead to such a page (**link-states**). Hence, every path alternates between a page-state and a link-state.

In Figure 1(b), we show the state space corresponding to this option for the publications domain. There are two advantages of this labeling. First, it reduces the sparsity of parameters by making the anchor word features be independent of the label of the source page. In practice, it is often found that the anchor text pointing to the same page are highly similar and this is captured by allowing multiple source labels to point to the same link state of label. Second for the foraging phase, it allows one to easily reason about intermediate probability of a path prefix where only the link is known and the page leading to it has not been fetched.

In this model, the state-features of the page states are the same as in the previous model, the state features of the link states are derived from the anchor text. Thus, the anchor-text transition features of the previous model, become state features of the link state. Thus the only transition features in this model are the edge features that capture the precedence order between labels.

3.1.2 Partially supervised case

In this case, apart from the start and goal pages none of the other pages have a label. A simple solution is to assign to each intermittent page a label equal to the distance from the goal or start state and then map it to the above problem by associating a label for each distance value. A similar labeling scheme was followed in [3]. The main limitation of this approach is that pages with similar content may not be exactly the same distance away for different websites and mapping them to different states might make it hard to exploit commonality across websites. In [10] the labeling scheme is discretized to coarser levels, but then it becomes difficult to determine a level of discretization that will neither generalize too much nor make it as specific as in [3]. We propose the following solution. Choose N, the path length from the start to the goal state that covers most except any outlying long paths. State i is assigned to pages distance i from the goal. This is similar to the method of [3] but with two crucial differences.

- First, we do not independently classify each page in a path but choose the best set of labels that allows only consistent transition from states $i+1$ to i and finally from state 1 to the goal state.

- Second, we define an additional hierarchy of features to capture commonality among adjacent labels. Normally in CRFs each state has its own set of features and their corresponding weight parameters w. We create a binary hierarchy over state indices and for each feature type at the leaf level we define features for groups of states at the internal nodes. For example consider a feature f called "word `paper` appears in anchor text". Each state from 1 to N will have a weight term corresponding to f. We will define additional features for groups of states of the form {1,2}, {3,4}...{$N-1, N$}, {1,2,3,4} and so on. Alternatively we could have chosen a sliding window of varying size to define groups of states but we advocate the hierarchy-based approach since it increases the number of state features only by a factor of two and captures a lot of commonality across adjacent states. An important reason why we advocate this method of feature sharing over a fixed discretization as in [10] is because the weights of these features are learnt via the normal training process and if there is little similarity across grouped adjacent states these weights will automatically be set to a low value. Thus, this provides a supervised method of exploiting commonality when it exists and preserving the specificity of the different states when it does not.

3.2 Path Foraging

Given the trained sequential model M and a list of starting

pages of websites, our goal is to find all paths from the list that lead to the "goal" state in M while fetching as few unrelated pages.

The key issue in solving this is to be able to score from a prefix of a path already fetched, all the set of outgoing links with a value that is inversely proportional to the expected work involved in reaching the goal pages. Consider a path prefix of the form $P_1L_2P_3 \ldots L_i$ where L_{i-1} is a link to page P_i in the path. We need to find for link L_i a score value that would indicate the desirability of fetching the page pointed to by L_i. This score is computed in two parts. First, we estimate for each state y, the proximity of the state to the goal state. We call this the reward associated with the state. Then we compute for the link L_i, the probability of its being in state y.

3.2.0.1 Reward of a state.

We apply techniques from Reinforcement Learning to compute the reward score that captures the probability of a partially-observed sequence to end-up in a goal state of the CRF model M. Reinforcement Learning is a machine learning paradigm that helps in choosing the optimal action at each state to reach the goal states. The goal states are associated with rewards that start to depreciate as the goal states get farther from the current state. The actions are chosen so as to maximize the cumulative discounted reward. We estimate this probability based on the training data by learning a reward function $\boldsymbol{\mathcal{R}}$ for each state. For each position i of a given sequence $\mathbf{x}$ we estimate the expected proximity to the goal state from a state y $\mathcal{R}_i^{\mathbf{x}}(y)$ recursively as follows:

$$\mathcal{R}_i^{\mathbf{x}}(y) = \begin{cases} \sum_{y'} e^{\mathbf{W}\cdot\mathbf{f}(y',y,\mathbf{x},i+1)}\mathcal{R}_{i+1}^{\mathbf{x}}(y') & 1 \le i < n \\ [\![y == goal]\!] & i = n \end{cases} \quad (3)$$

When $i = n$, the reward is 1 for the goal state and 0 for every other label. Otherwise the values are computed recursively from the proximity of the next state and the probability of transition to the next state from the current state.

We then compute a weighted sum of these positioned reward values to get a position independent reward values. The weight are controlled via γ, a discount factor that captures the desirability of preferring states that are closer to the goal state as follows:

$$\boldsymbol{\mathcal{R}}^{\mathbf{x}} = \frac{\sum_{k=1}^{n} \gamma^k \cdot \boldsymbol{\mathcal{R}}_{n-k}^{\mathbf{x}}}{\sum_{k=1}^{n} \gamma^k} \quad (4)$$

where n is the length of the sequence.

The final reward value of a state is computed by averaging over all training sequences $\mathbf{x}_1 \ldots \mathbf{x}_N$ as

$$\boldsymbol{\mathcal{R}} = \frac{\sum_{\ell=1}^{N} \boldsymbol{\mathcal{R}}^{\mathbf{x}_\ell}}{N} \quad (5)$$

3.2.0.2 Probability of being in a state.

Consider a path prefix of the form $P_1L_2P_3 \ldots L_i$ where L_{i-1} is a link to page P_i in the path. We need to find for link L_i the probability of its being in any one of the link states. We provide a method for computing this. Let $\alpha_i(y)$ denote the total weight of ending in state y after i states. We thus define $\alpha_i(y)$ as the value of $\sum_{\mathbf{y}' \in \mathbf{y}_{i:y}} e^{\mathbf{W}\cdot\mathbf{F}(\mathbf{y}',\mathbf{x})}$ where $\mathbf{y}_{i:y}$ denotes all label sequences from 1 to i with i-th position labeled y. For $i > 0$, this can be expressed recursively as

$$\alpha_i(y) = \sum_{y' \in \mathcal{Y}} \alpha_{i-1}(y') e^{\mathbf{W}\cdot\mathbf{f}(y,y',\mathbf{x},i)} \quad (6)$$

with the base cases defined as $\alpha_0(y) = 1$.

The probability of L_i being in the link state y is then $\frac{\alpha_i(y)}{\sum_{y' \in \mathcal{Y}_\mathcal{L}} \alpha_i(y')}$ where $\mathcal{Y}_\mathcal{L}$ denotes the link states.

3.2.0.3 Score of a link.

Finally, the score of a link L_i after i steps is calculated as the sum of the product of reaching a state y and the static reward at state y.

$$\text{Score}(L_i) = \sum_y \frac{\alpha_i(y)}{\sum_{y' \in \mathcal{Y}_\mathcal{L}} \alpha_i(y')} \mathcal{R}(y) \quad (7)$$

If a link appears in multiple paths, we sum over its score from each path.

Thus, at any give snapshot of the crawl we have a set of unfetched links whose scores we compute and maintain in a priority queue. We pick the link with the highest score to fetch next. The links in the newly fetched page are added to the queue. We stop when no more unfetched links have score above a threshold value.

4. EXPERIENCE AND SUMMARY

We refer the reader to [12] for a detailed experimental study of the fully supervised case. We experimented on two different tasks: fetching publications starting from computer science department home pages of various universities and extracting pages containing contact addresses of companies. We first evaluated the accuracy of path classification using the CRF-based sequential learning-based approach and compared it with the previously proposed method of independent classifiers using naive Bayes and maximum entropy classifiers. The sequential models obtained F1-values 10 to 20 percentage points higher than the best independent per class classifiers. During foraging, we were able to achieve harvest rates close to 90%.

We are continuing our experiments in the semi-supervised setting where the user does not need to specify milestone states. We plan to integrate such website level extraction tasks with finer grained information extraction engines that can find more specific information like the exact address from the goal page.

5. REFERENCES

[1] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In *WWW, Hawaii*. ACM, May 2002.

[2] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*, 118(1/2):69–113, 2000.

[3] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using Context graphs. In *26th International Conference on Very Large*

Databases, VLDB 2000, pages 527–534, Cairo, Egypt, 10–14 September 2000.

[4] J.R.Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[5] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[6] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large-scale optimization. *Mathematic Programming*, 45:503–528, 1989.

[7] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of The Sixth Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55, 2002.

[8] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of The Seventh Conference on Natural Language Learning (CoNLL-2003)*, Edmonton, Canada, 2003.

[9] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–286, February 1989.

[10] Jason Rennie and Andrew Kachites McCallum. Using reinforcement learning to spider the Web efficiently. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 335–343, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[11] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL 2003*, pages 213–220. Association for Computational Linguistics, 2003.

[12] V.G.Vinod Vydiswaran and Sunita Sarawagi. Learning to extract information from large websites using sequential models. In *COMAD*, 2005.