Hongyu Liu, Evangelos Milios, Jeannette Janssen: ``Probabilistic Models for Focused Web Crawling'', 6th ACM International Workshop on Web Information and Data Management (WIDM 2004), November 12-13, 2004, Washington, DC, USA (with corrections)

Probabilistic Models for Focused Web Crawling

Hongyu Liu[1], Evangelos Milios[1], Jeannette Janssen [1,2]
[1]Faculty of Computer Science, Dalhousie University
[2]Dept. of Mathematics and Statistics, Dalhousie University
Halifax, NS, Canada B3H 1W5
{hongyu,eem}@cs.dal.ca, janssen@mathstat.dal.ca

ABSTRACT

A Focused crawler must use information gleaned from previously crawled page sequences to estimate the relevance of a newly seen URL. Therefore, good performance depends on powerful modelling of context as well as the current observations. Probabilistic models, such as Hidden Markov Models(HMMs) and Conditional Random Fields(CRFs), can potentially capture both formatting and context. In this paper, we present the use of HMM for focused web crawling, and compare it with Best-First strategy. Furthermore, we discuss the concept of using CRFs to overcome the difficulties with HMMs and support the use of many, arbitrary and overlapping features. Finally, we describe a design of a system applying CRFs for focused web crawling, that is currently being implemented.

Categories and Subject Descriptors

H.5.4 [**Information interfaces and presentation**]: Hypertext/hypermedia; I.5.4 [**Pattern recognition**]: Applications, Text processing; I.2.6 [**Artificial intelligence**]: Learning; I.2.8 [**Artificial intelligence**]: Problem Solving, Control Methods, and Search.

General Terms

Algorithms, performance, measurements, experimentation.

Keywords

Focused crawling, Web Graph, Hidden Markov Models, Conditional Random Fields, World Wide Web

1. INTRODUCTION

The concept of Focused crawler is increasingly seen as a potential solution to the problem of indexing the exponentially growing Web. It is designed to traverse a subset of the Web to only gather documents on a specific topic and aims to identify the promising links that lead to target documents, and avoid off-topic branches. In the Web graph, relevant pages tend to link to other relevant ones. For example, if looking for pages on the topic *Linux*, the crawler should ignore *Health* related web pages. However, off-topic pages may often reliably lead to relevant pages. For example, a University page leads to a Department page, which leads to a People page, Faculty list page, Research and Publications. When looking for research publications on a specific topic, the crawler may have to traverse pages that are irrelevant to the topic, before it reaches highly relevant ones. In this case, a good strategy is to effectively determine which links to follow to get to relevant pages.

Focused crawling is designed to traverse a subset of the Web to only gather documents on a specific topic and aims to identify the promising links that lead to target documents, and avoid off-topic branches. Exhaustive crawling uses Breadth-First search to download as many pages as possible, while a focused crawler aims to selectively choose links leading to targets. The major problem of focused crawlers is to identify the next most important link to follow.

Previous work in focused crawling algorithms can be found in the literature: Breadth-First search [18], genetic algorithm [10], reinforcement learning [21, 9], arbitrary predicate [1], Shark-search [8], focused crawling [4, 3] and others [5, 14, 15]. A framework to evaluate different crawling strategies is described in [17, 16, 23]. They found that the Best-First strategy performed best. An interesting approach proposed in [6] uses the backlink service of certain search engines to construct context graphs. However, it is not realistic for a focused crawler to rely on search engines like Google to obtain backlink information. Furthermore, the assumption that all pages in a certain physical layer from a target document will share terms does not always hold.

This paper proposes probabilistic models for focused crawling that integrate evidence from both content and linkage. Our approach is unique in two respects. One is the way we use Random Markov models for focused crawling. We think of a focused crawler as a random surfer, over an underlying Markov chain of hidden states, defined by the number of hops away from targets, from which the actual topics of the document are generated. Another contribution is modelling the semantic structure of the web by observing the user's behavior on a small training data set and application of this model to guide the actual web crawler in order to find the relevant documents. Experiments show that by learning from user's browsing, the crawler often outperforms Best-First strategy.

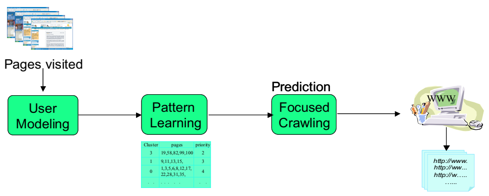
The rest of the paper is organized as follows. Section 2 describes the architecture of our focused crawling system. Section 3 describes an implemented system based on Hidden Markov models and initial experimental results. Section 4 describes the conceptual design of a system based on CRFs, that is currently being implemented. Section 5 discusses the results and future work.

2. SYSTEM OVERVIEW

The Web graph is noisy and multi-topic. Off-topic pages often reliably lead to highly relevant pages and paths leading to targets tend to be longer. Links may lead to unrelated topics within an extremely short radius. At the same time, there exist long paths and large topical subgraphs where topical coherence persists. Webmasters usually follow some general rules to organize web pages semantically, for example, university pages are likely linked to department pages, then to faculty pages and rarely linked to sports canoeing pages. That is, dominant semantic topic hierarchies exist. User surfing behavior is likely to follow her intuitive understanding of such semantic topic hierarchies to locate pages relevant to her information needs efficiently while ignoring areas not linking to targets.

The above observations suggest that the context of the hyperlinks the user follows and marks as relevant reveals the user's information needs. If we can detect such patterns hidden in the surfer's topic-specific browsing, we may be able to build an effective focused crawler.

The system consists of three components: User Modelling, Pattern Learning and Focused Crawling, as shown in Fig.1.



Cluster	pages	priority
3	19,58,82,99,100	2
1	9,11,13,15,	3
0	1,3,5,6,8,12,17, 22,28,31,35,	4
. .	. . . . .	. .

Figure 1: System Architecture.

2.1 User Modelling

In the User Modelling stage, we aim to collect the sequences of pages visited by the user during her topic-specific browsing. If user finds current browsing page is interesting, she can click the ***Useful*** button which is added to page being browsed, then the page will become an annotated target page. In order to analyze the user browsing pattern, we construct a web graph which contains both nodes and edges.

Each node in the web graph presents a HTML page with a unique URL, and an edge is established if a referring document has a hyperlink pointing to a referred document. After completing browsing, user clicks ***Submit*** button to save entire visited pages as training data.

The documents in the web graph are then processed to extract the semantic relationships among words in the collection. The documents are analyzed using Latent Semantic Indexing(LSI) [22, 2]. After LSI is applied, co-occurring concepts are mapped onto the same dimension.

To capture semantic relations between visited web pages during topic-specific browsing, the collection of web pages visited are clustered [13]. We use a combination of K-means to obtain initial clusters, with $K = 5$, and refine the clusters using the EM algorithm.

Our goal is to capture the concept associative relation between different types of documents, i.e. the notion that document type A is more likely to lead to targets than document type B, rather than what exact categories they belong to. After clustering, the associative relationships between groups are captured into a *concept graph* $G = (V, E)$, where each node is assigned to C_i, the label of the cluster it belongs to. Details on the construction of the web and concept graphs are presented in [12].

2.2 Learning Patterns Leading to Targets

Focused crawler can only use information gleaned from previously crawled page sequences to estimate the relevance of a newly seen URL, therefore, good performance relies on powerful modelling of context as well as the current observations. Probabilistic models, such as HMM and CRF, offer a good balance between simplicity and expressiveness of context.

We model focused crawling in the following way: We think of focused crawler as a random surfer and there is an underlying Markov chain of hidden states, defined by the number of hops away from targets, from which the actual topics of the document are generated. Let S be the set of finite hidden states $S = \{T_0, T_1, ..., T_{n-1}\}$, defined by reaching a target page by 0, 1,..., n-1 or more hop(s). Let $s = \{s_1, s_2, ..., s_m\}$ be some sequence of hidden states, let $o = \{o_1, o_2..., o_m\}$ be some observed data sequence, such as a sequence of web pages the crawler has seen, then the sequence of states that was most likely to have generated the observations can be estimated. Apparently, for those pages with lower state subscript such as T_0 will be assign higher visit priorities than those with higher state subscript such as T_2. We will discuss learning and inference details in section 3 for applying HMM and in section 4 for CRF model.

2.3 Focused Crawling

The system uses the learned user browsing pattern in focused crawling. The crawler utilizes a queue, which is initialized with the starting URL of the crawl, to keep all candidate URLs ordered by their visit priority value. The crawler downloads the page pointed to by the top URL of the queue, calculates its reduced dimensionality (LSI) representation, and extracts all the outlinks. Then all children pages will be downloaded and classified using K-Nearest Neighbor algorithm into one of the clusters. The most likely state sequences are calculated for each parent-child pair based on current observations and corresponding HMM/CRF parameters, and the visit priority values will be assigned accordingly Since the queue is always sorted according to the visit priority value associated with each URL, we expect that URLs on the top of the queue will locate targets rapidly.

3. USE OF A HIDDEN MARKOV MODEL

3.1 Hidden Markov Model (HMM)

A HMM [20] is a finite set of states $S = \{s_1, s_2..., s_n\}$ and a finite set of observations $O = \{o_1, o_2..., o_m\}$ associated with two conditional probability distributions $P(s_j|s_i)$ and $P(o|s_j)$. There is also a initial state distribution $P_0(s)$. HMMs, widely used in speech-recognition and information extraction, provide superior pattern recognition capabilities for dynamic patterns. HMMs are useful when one can think of underlying unobservable events probabilistically generating surface events, that are observable.

Let n be the number of hidden states,

- Hidden states: $S = \{T_{n-1}, T_{n-2}, ..., T_1, T_0\}$
  - Reaching a target page by $n-1$ or more, $n-2$,...,1, 0 hop(s).
- Visible states: $O = \{1, 2, 3, 4, 5\}$
  - Cluster number which web pages belong to.
- HMM parameters θ:
  - Initial Probability Distribution Matrix $\pi = \{P(T_0), P(T_1), P(T_2), ..., P(T_{n-1})\}$. Probability of reaching a target by 0, 1, 2,..., n-1 or more hop(s) at time 1, respectively.
  - Matrix of Transition Probabilities: $A = [a_{ij}]_{n\times n}$, where, a_{ij} = probability of being in the T_j state at time *t+1* given that you are in state T_i at time *t*.
  - Matrix of Emission Probabilities: $B = [b_{ij}]_{n\times 5}$, where,
    b_{ij} = probability of seeing cluster j if you are in state T_i.

Once the state-transition structure is determined, the parameters of the model to be estimated are the state transition and the emission probabilities. We use annotated training data – that is, sequences of user visited pages with identified target pages from user modelling stage, and label all nodes in the concept graph as T_0, T_1, T_2,..., T_{n-1} in a Breadth-First search out of the set of target pages (T_0) as shown in Fig.2.

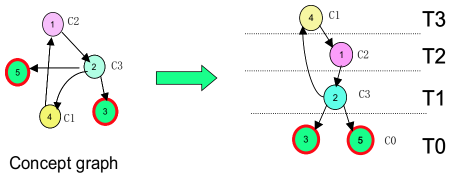


Figure 2: Parameter Estimation of HMM.

Probabilities are estimated by maximum likelihood with ratios of counts, as is traditional. That is,

$$a_{ij} = \frac{|L_{ij}|}{\sum_{k=0}^{n-1} |L_{kj}|} \tag{1}$$

where, $L_{ij} = \{v \in Ti, w \in Tj : (v, w) \in E\}$.

$$b_{ij} = \frac{|N_{ij}|}{\sum_{k=1}^{5} |N_{kj}|} \tag{2}$$

where, $N_{ij} = \{C_i : C_i \in Tj\}$.

An example of learned parameters with topic *linux* is shown in Fig.3. The sum of each column in matrix A and the sum of each row in B are 1.

$$n = 5, C = 5$$

$$\pi = \{0.2, 0.2, 0.2, 0.2, 0.2\}$$

$$A = \left[\begin{array}{c|ccccc} & T0 & T1 & T2 & T3 & T4 \\ T0 & 0.29 & 0.41 & 0.20 & 0.00 & 0.14 \\ T1 & 0.71 & 0.38 & 0.52 & 0.33 & 0.16 \\ T2 & 0.00 & 0.21 & 0.20 & 0.33 & 0.04 \\ T3 & 0.00 & 0.00 & 0.09 & 0.33 & 0.00 \\ T4 & 0.00 & 0.00 & 0.00 & 0.00 & 0.66 \end{array}\right] \quad B = \left[\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ T0 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ T1 & 0.00 & 0.23 & 0.34 & 0.28 & 0.16 \\ T2 & 0.00 & 0.33 & 0.22 & 0.33 & 0.11 \\ T3 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ T4 & 0.00 & 0.26 & 0.30 & 0.28 & 0.16 \end{array}\right]$$

Figure 3: Example. Topic: */Computer/Software/Operating System/Linux.* # of Web pages=200, # of targets=30.

3.2 Efficient Inference

Given the model and all its parameters, the task is to find the most likely state sequence given the observed web page sequence. Determining this sequence is efficiently performed by dynamic programming with the *Viterbi* algorithm [20]. Forward value $\delta(s_i, t)$ is the maximum probability of all sequences ending at state s_i at time t, and the partial best path is the sequence which achieves this maximal probability. A recursive relation holds for these values:

$$\delta(s, t + 1) = \max_{s'} \delta(s, t)P(s|s')P(o_{t+1}|s) \tag{3}$$

where, $P(s|s')$ and $P(o_{t+1}|s)$ are transition probabilities and emission probabilities, respectively.

In particular, in the case of focused crawling, we describe $\delta(i, w)$ as the maximum probability of all web page sequences ending at state T_i when web page w is seen. For each crawled page w, we calculate all partial best paths for each hidden state, each of which has an associated probability δ. We find the overall best path by choosing the state T_i with the maximum partial probability $\delta(i, w)$, i.e. the maximum probability of all partial paths ending at state T_i at page w. As shown in Fig. 4(a), each web page w is associated with it cluster number C_w and information from its parent in the form of a cluster number C_p and partial probabilities $\delta(j, p)$. If page w is a start URL, then δ will be calculated using Initial Probability Distribution Matrix π using equation 4,

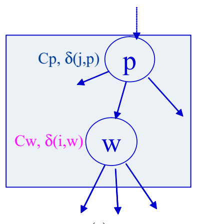
$$\delta(i, w) = \pi(i)b_{ic_w} \tag{4}$$

otherwise, δ will be calculated based on information from its parent and current observations using equation 5,

$$\delta(i, w) = \max_j(\delta(j, p)a_{ji}b_{ic_w}) \tag{5}$$

There is also a back pointer ψ shown in equation 6, which points to the predecessor state T_j that optimally provokes the current state T_i, and the most probable hidden state path will be used to obtain the URL visit order.

$$\psi(i, w) = \arg\max_j (\delta(j, p) a_{ji}) \tag{6}$$



(a)

$$Ulr\_older\ Matrix\ T:$$

$$\begin{bmatrix} & T0 & T1 & T2 & T3 \\ T0 & 1 & 3 & 6 & 10 \\ T1 & 2 & 4 & 7 & 11 \\ T2 & 0 & 5 & 8 & 12 \\ T3 & 0 & 0 & 9 & \frac{n^2+3n-2}{2} \end{bmatrix}$$

(b)

Figure 4: Efficient Inference.

The Viterbi algorithm is used to determine the most likely sequence of underlying hidden states, so we also define a Url_order Matrix to assign visit priority order, as shown in Fig.4(b). For page w, if the most probable hidden state path from page p is $T_k \rightarrow T_m$, the visit priority order would be $Url\_order[T_k, T_m]$. The total number of state transitions $trans$ is a function of the number of states n. i.e., $trans = (n^2 + 3n - 2)/2$. Some state transitions are not possible. For example, there is no $T_3 \rightarrow T_1$, since if you can go from T_3 to T_1 in one hop, then the page is not a true T_3, it is T_2. The same as $T_3 \rightarrow T_0, T_2 \rightarrow T_0$ transitions. It is possible to have $T_j \rightarrow T_j (j = 0..n-1)$ transitions. This will happen if there are cross links at the same distance from the targets.

3.3 Experiments

3.3.1 Evaluation

It is important that the focused crawler return as many interesting pages as possible while minimizing the proportion of uninteresting ones. To evaluate the effectiveness of our approach, we use *precision* measure, which is the percentage of the web pages crawled that are relevant. Since target pages user marked as interesting may belong to different topics, the relevance assessment of a page p is based on the maximal cosine similarity to the set of target pages T with a confidence threshold γ. That is, if

$$\max_{t \in T} \cos(p, t) \geq \gamma$$

then p is considered as relevant, where $\cos(p, t)$ is the standard cosine similarity function and w_{dk} is the TF-IDF weight of reduced vector term k in document d.

3.3.2 Results

We selected a variety of topics from DMOZ [1] which are neither too broad nor too narrow, as shown in Table 3.3.2. For comparison, we chose Best-First Search (BFS) crawler. According to [17], the Best-First crawling is competitive and outperforms other focused crawling strategies. Its visit priority order of URL here is based on maximal cosine similarity between the set of target pages and the page where link was found.

[1]http://dmoz.org/

Table 1. Different kind of Crawls

Topics	# of pages user visited	# of targets user marked	γ	Start URL
/Sports/hockey	207	34	0.7	http://sportsnetwork.com, http://about.com/sports
/Computers/Software/Operating _Systems/Linux	200	30	0.9	http://www.computerhope.com,http://about.com/compute http://comptechdoc.org/os, http://www.informationweek.com/techcenters/sw/
/Health/Condition_Diseases/Diabetes, Heart Diseases	112	18	0.8	http://www.cnn.com
/Health/Condition_Diseases/Diabetes, Heart Diseases	112	18	0.8	http://www.nih.gov/, http://www.healthfinder.gov/ http://www.healthatoz.com/, http://www.healthweb.org/
/Health/Condition_Diseases/Diabetes, Heart Diseases	112	18	0.8	http://chealth.canoe.ca/; http://health.allrefer.com/

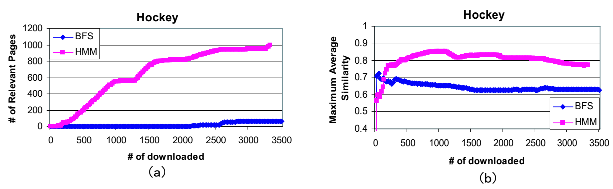


Figure 5: Topic *Hockey*: (a) # of relevant pages retrieved; (b) Maximum Average Similarity of all downloaded pages.

.

The number of relevant pages against the number of pages downloaded for the topic *Hockey* is shown in Fig.5(a). The system collected a total of 207 user browsed web pages. The results show a significant improvement over Best-First crawling. It appears that Best-First crawling pursued the links that appear the most promising at the expense of longer term loss, whereas our focused crawler explored suboptimal links that eventually lead to longer term gains, in spite of a penalty at the early stage of the crawl, as shown in Fig.5(b).

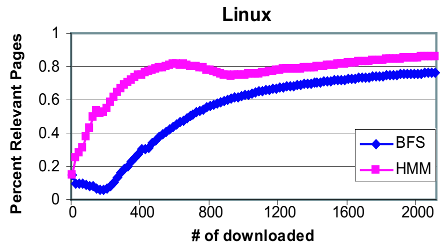


Figure 6: Topic *Linux*: Percentage of relevant pages.

Another topic *Linux*, for which our system showed the least average performance improvement over Best-First crawler, although it still outperforms Best-First crawler, is shown in Fig.6. We conjecture that such topics are those with long paths and large subgraphs where topical coherence persists, so Best-First crawler performs well too. However, our system locates relevant pages quickly at the beginning.

Another advantage of our system is focusing on what the user really wants on specific topics. In other focused crawling systems, target pages are usually predefined or keywords-based and therefore are more general and less specific on user's personal interests. As a result, it will retrieve tons of returning "relevant" pages to the predefined targets and user still needs to investigate further to find more specific information. With the user browsing stage in our system, user's topic-specific interests are effectively captured and they are

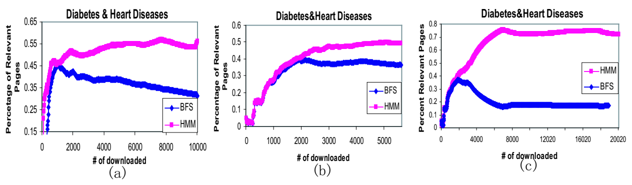


Figure 7: Topic is *Diabetes and Heart Disease* with different start URLs.

not limited to one topic and single domain. Fig.7(a) shows the results of crawling topics *Diabetes* and *Heart Disease* for the user at the same time and the numbers of relevant pages retrieved by our system are 50% more than those by Best-First crawler when crawling 10,000 pages.

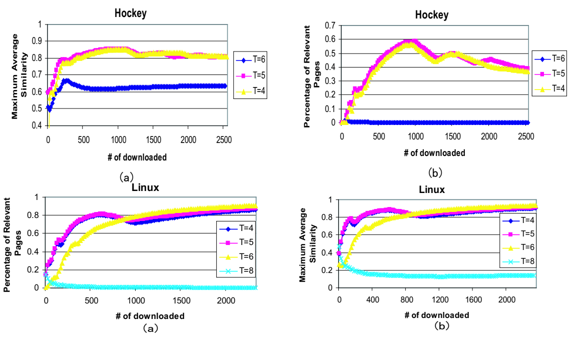


Figure 8: Effect of different HMM states T. First row: Topic is *Hockey*. Second row: Topic is *Linux*.

In order to investigate the effect of the HMM state transition structure, we present experimental results using different number of states on each topic, respectively, shown in Fig.8. There is visible improvement in the performance using the selected number of states and the best number of states may vary depending on topic. For topic *Hockey*, HMM with 4 or 5 states works much better. In particular, with 6 states, the performance drops down to that of Best First crawling, indicating that the typical linkage structure leading to the topic doesn't exist anymore. For topic *Linux*, HMM with 6 performs the best, and the one with 8 states yields the worst. These observations match our initial conjecture that the HMM state transition structure reflects the general hierarchical concept linkage structure of a topic, rather than physical layers of Web graph. On the surface, the Web graph is rapidly mixing with random links from topic to topic within both short and long paths, however, the topical concept linkage structure behind the scene does exist and doesn't vary too much. Back to our examples, for topic Hockey, the general rule could be $Top \rightarrow Sports \rightarrow Hockey \rightarrow Ice - Hockey$, indicating that the level varies between 3-5, and HMM with 6 or more will mislead the focused crawler. For topic *Linux*, the path is longer but should be less than 8, which implies $Top \rightarrow Computers \rightarrow Software \rightarrow OperatingSystem \rightarrow Linux$. Large number of states intensifies the problem of limited training data.

4. USE OF CONDITIONAL RANDOM FIELD

In the HMM we proposed above, we try to capture semantic relations along paths leading to targets. Keywords are used implicitly in the construction of clusters and classification of pages into them. Furthermore, features associated with anchor text, title or metadata, URL token, are not accounted for. It may, therefore, be useful if a composite criterion were formed, that directly included keywords and other features. The hope is that with multiple features, even if only some of them are present, relevant paths will be identified. Linear-chain Conditional Random Fields [11] is a formalism that generalizes HMMs, and may help provide a more flexible framework, where multiple features are used.

4.1 Conditional Random Fields(CRFs)

Linear-chain CRFs were first introduced in [11]. Its effectiveness has been demonstrated in the tasks of Table Extraction [19] and Labelling Sequence Data [11].

Similar to the HMM formulation, let S be the set of hidden states $S = \{T_0, T_1, ..., T_{n-1}\}$, defined by reaching a target page by $0, 1, ..., n-1$ or more hop(s). Let $s = \{s_1, s_2, ..., s_m\}$ be a sequence of (hidden) states, and $o = \{o_1, o_2 ..., o_m\}$ the corresponding observed web page sequence. CRFs specify the probability of possible state sequences given an observation sequence $p(s|o)$ rather than joint probability $p(s, o)$, as in HMMs, as follows:

$$p(s|o) = \frac{1}{Z_0} \exp(\sum_{t=1}^{m} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t)) \qquad (7)$$

where s is a state sequence, o is an observed data sequence, k is the number of features, $\theta = \{\lambda_1, \lambda_2, ..., \lambda_k\}$ are parameters to be estimated from the training data set, f_k are predefined feature functions. Z_0 is an observation-dependent normalization factor over all data sequences which can be calculated as follows:

$$Z_0 = \sum_{s \epsilon S^m} \exp(\sum_{t=1}^{m} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t)) \qquad (8)$$

where S^m represents the set of all data sequences.

4.2 Features

4.2.1 Feature set

In HMM, we focus on capturing semantic content and linkage relations leading to targets by applying LSI and clustering algorithms. Normally, a document relevant to a specific topic frequently contains explicitly a set of topic-specific keywords. For example, a TCP/IP document often contains keywords "tcp, ip, header, protocol", etc. Therefore, the lexical keywords are a significant factor and it may be beneficial to use them directly.

To build good topic-specific keywords, we extend the User Modelling stage in Section 2.1 by adding a Keyword Acquisition step: we let the user specify 3-5 keywords before or after her topic-specific browsing. Additional keywords are extracted from the text in the title and meta tags of the pages annotated as relevant. For example, keywords for topic *Linux* could be "Linux, software, operating system, OS, Redhat, LinuxPro, OpenLinux, manual, documentation project".

When defining feature functions, we construct a set of real-valued features to capture whether the observed web pages have specific characteristics. The notation for feature k and observed web page o is $b_k(o)$. We propose nine features ($k = 1..9$).

1. **Cluster Feature**: For each cluster $j = 1..5$, we define a feature capturing whether the observed web page o belongs to the cluster.

$$b_j(o) = \begin{cases} 1 & \text{if } o \text{ is in cluster } j; \\ 0 & \text{otherwise.} \end{cases}$$

2.**Text Feature**: Maximal similarity value between the content of a given candidate page and the set of targets. We define it as the 6th features.

$$b_6(o) = \begin{cases} 1 & \text{if sim(target, o)} \geq \text{a threshold;} \\ 0 & \text{otherwise.} \end{cases}$$

3. **URL Token Feature**: The tokens in the URL of an observed page may contain valuable information about predicting whether or not a page is a target page or potentially leading to a target. For example, a URL containing "linux" is more likely to be a web page about linux related information, and a URL which contains the word "operating system" or "OS" indicates that with high probability, it may lead to a Linux page. We first parse the URL into tokens, then compute the similarity between tokens and topic keywords.

$$b_7(o) = \begin{cases} 1 & \text{if sim(keywords, URL tokens of } o) \geq \text{a threshold;} \\ 0 & \text{otherwise.} \end{cases}$$

4. **Anchor Text Feature**: The anchor text around the link pointing to an observed page o often is closely related to the topic of the page. Human's skills and knowledge of discriminating between links when they browse mostly rely on the anchor texts.

$$b_8(o) = \begin{cases} 1 & \text{if sim(keywords, anchor text of page pointing to } o) \\ & \geq \text{a threshold;} \\ 0 & \text{otherwise.} \end{cases}$$

5. **Meta Data Feature**: Meta Data and title often capture the content of the observed web page.

$$b_9(o) = \begin{cases} 1 & \text{if sim(keywords, Title and Meta Data of } o) \\ & \geq \text{a threshold;} \\ 0 & \text{otherwise.} \end{cases}$$

4.2.2 *Feature Representation*

In CRFs, each feature can be represented as either a binary or a real value. For binary value, a 1 indicates the presence of the feature, whereas, a 0 means the lack of the feature, as shown in last section. With CRFs, we also can use real-valued features directly and in this case, the actual similarity values are the inputs.

Each feature function $f_k(s_{t-1}, s_t, o, t)$ is defined to have a boolean output and to take as inputs the value of one of the features and particular values on current state s_t and previous state s_{t-1}.

4.3 Parameter Estimation

Given a set of features and the conditional probability function, Eq. 7, the training task is to estimate the parameters $\theta = \{\lambda_1, \lambda_2, ..., \lambda_k\}$. As we have done with HMM, we label all nodes in the concept graph as T_0, T_1, T_2,..., T_{n-1} in a Breadth-First search out of the set of target pages (T_0) as shown in Fig.2. Unlike in HMM, we extract all annotated page sequences the user visited to form the training data set. In our case, since we focus on goal-directed browsing, a sequence of pages terminates on reaching a target page (T_0). For example, we have both $4 \rightarrow 1 \rightarrow 2 \rightarrow 3$ and $4 \rightarrow 1 \rightarrow 2 \rightarrow 5$. The training procedure is summarized as follows:

- Input: training data $D = \{< o, s >_{(i)}\}$, while $i = 1..N$ labelled sequences.
- Output: parameters $\theta = \{\lambda_1, \lambda_2, ..., \lambda_k\}$
- Maximize: the log-likelihood objective function L [19]:
$$L = \sum_{i=1}^{N} \log(p(s_i|o_i)) - \sum_k \frac{\lambda_k^2}{2\sigma^2}$$
where σ^2 is the variance of a Gaussian prior that provides smoothing.
- Method: L-BFGS. It has been shown [19, 7] that this method is faster than iterative scaling and gradient based method and the optimization procedure only requires the first derivative of the log-likelihood function L:
$$\frac{\delta L}{\delta \lambda_k} = (\sum_{i=1}^{N} C_k(s_i|o_i)) - (\sum_{i=1}^{N} \sum_s p(s|o_i) C_k(s|o_i)) - \frac{\lambda_k}{\sigma^2}$$
where C_k is the count of all feature k values given s and o.

4.4 Efficient Inference

As in HMMs, efficient inference in CRFs can also be calculated by the *Viterbi* algorithm with slightly modified forward value δ as in Eq. 3:

$$\delta(s, t+1) = \max_{s'} \delta(s, t) \exp(\sum_k \lambda_k f_k(s', s, o, t))$$

The task in our case is to find the most likely state sequence given the observable web page sequence.

The normalization factor Z_0 can also be efficiently calculated by dynamic programming. Let $\alpha(s_i, t)$ be the forward value, representing the probability of all sequences ending at state s_i at time t, and recurse:

$$\alpha(s, t+1) = \sum_{s'} \alpha(s', t) \exp(\sum_k \lambda_k f_k(s', s, o, t)) \qquad (9)$$

and Z_0 in equation 8 is then $\sum_s \alpha(s, m)$.

In this case, CRFs can be roughly understood as conditionally-trained Hidden Markov Models with great freedom to combine arbitrary complex and non-independent feature sets by applying efficient training and decoding algorithms. As we have seen, it is easy to integrate it into our current system and we expect that better predictions should result in from this robust model [19].

For example, if we are looking for topic *Linux*, several sequences of pages ending at a target page will be extracted from the Web graph built in the User Modeling stage. These sequences contain linkage and content information leading to targets that we are trying to capture. The parameters $\theta = \{\lambda_1, \lambda_2, ..., \lambda_9\}$ will be estimated as shown in Section 4.3, e.g., $\theta = \{0.25, 0.15, 0.1, 0.05, 0.0, 0.10, 0.15, 0.15, 0.05\}$, indicating that with high probability, cluster 1 and 2 are more likely lead to targets, and keywords are more likely to appear in Url tokens and Anchor text for topic *Linux*, therefore higher weights are given to parameters $\lambda_1, \lambda_2, \lambda_7, \lambda_8$. When a new page is seen during Focused Crawling, features will be calculated, the *Viterbi* algorithm is used to find the most likely state sequence, and higher weights make the corresponding features play a more important role in inference. For example, if a page belonging to cluster 1 or 2, and/or its Url or anchor text contains keywords "linux" or "operating system", or "OS", it is more likely to lead to targets.

5. DISCUSSION

In this paper we present our focused crawler system using two probabilistic models, HMMs and CRFs, to model the link structure and content of documents leading to target pages by learning from user's topic-specific browsing. Our system is unique in several respects. The proposed way of capturing and exploiting user's personalized interests can potentially lead to more effective focused crawlers.

While the architecture using HMM we presented already shows promising performance, the CRF design needs to be implemented and evaluated. Furthermore, the effect of clustering/classification algorithms on the performance of the system must be investigated. We are in the process of empirically evaluating the effectiveness of augmenting features using CRFs.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support from the MITACS Network of Centres of Excellence, IT Interactive Services Inc. and the Natural Sciences and Engineering Research Council of Canada.

7. REFERENCES

[1] C. Aggarwal, F. Al-Garawi, and P. Yu. Intelligent Crawling on the World Wide Web with Arbitrary Predicates. In *Proceedings of the 10th International WWW Conference*, Hong Kong, May 2001.

[2] M. W. Berry. LSI: Latent Semantic Indexing Web Site. *http://www.cs.utk.edu/~lsi/*, Accessed May 15, 2004.

[3] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In *Proceedings of the 11th International WWW Conference*, Hawaii, 2002, USA.

[4] S. Chakrabarti, M. van den Berg, and B. Dom. Focused Crawling: a new approach to topic-specific web resource discovery. In *Proceedings of the Eighth International WWW Conference*, Toronto, Canada,1999.

[5] J. Dean and M. Henzinger. Finding Related Pages in the World Wide Web. In *Proceedings of the 8th International WWW Conference*, pages 389–401, 1999.

[6] M. Diligenti, F. Coetzee, S. Lawrence, C. Giles, and M. Gori. Focused Crawling Using Context Graphs. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB 2000)*, Cairo, Egypt, September 2000.

[7] F.Sha and F. Pereira. Shallow Parsing with Conditional Random Fields. In *Proceedings of Human Language Technology*, NAACL, 2003.

[8] M. Hersovici, M. Jacovi, Y. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The Shark-Search Algorithm - An Application: Tailored Web Site Mapping. In *Proceedings of the Seventh International WWW Conference*, Brisbane, Australia, April 1998.

[9] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of IJCAI97*, August 1997. Accessed May 2004. http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/project-home.html.

[10] J. Johnson, K. Tsioutsiouliklis, and C. L. Giles. Evolving Strategies for Focused Web Crawling. In *The Twentieth International Conference on Machine Learning (ICML-2003)*, August 21-24, 2003 Washington, DC USA.

[11] J. Lafferty, A. McCallum, and F.Pereira. Conditional random fields: Probabilistic models for segmenting an labeling sequence data. In *International Conference on Macvhine Learning(ICML-2001)*, 2001.

[12] H. Liu, E. Milios, and J. Janssen. Focused crawling by learning hmm from user's topic-specific browsing. In *Proceedings of 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 732–735, Beijing, China, September 20-24 2004. http://www.maebashi-it.org/WI04/.

[13] D. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[14] F. Menczer and R. Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the web. *Machine Learning*, 39(2/3):203–242, 2000.

[15] F. Menczer, G. Pant, and P. Srinivasan. Myspiders: Evolve your own intelligent web crawlers. In *Autonomous Agents and Multi-Agent Systems*, pages 5, 241–249, 2003.

[16] F. Menczer, G. Pant, and P. Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. In *To appear in ACM TOIT* , Accessed May 2004. http://www.informatics.indiana.edu/fil/Papers/TOIT.pdf.

[17] F. Menczer, G. Pant, P. Srinivasan, and M. Ruiz. Evaluating Topic-Driven Web Crawlers. In *Proceedings of the 24th Annual International ACM/SIGIR Conference*, New Orleans, USA, 2001.

[18] M. Najork and I. N. Wiener. Breadth-first Search crawling yields high-quality pages. In *Proceedings of the 10th International WWW Conference*, Hong Kong, May 2001.

[19] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table Extraction Using Conditional Random Fields. In *26th Annual International ACM SIGIR conference*, pages Jul 28 – Aug. 1, 2003.

[20] L. R. Rabiner. A Tutorial on Hidden Markov Model and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[21] J. Rennie and A. McCallum. Using Reinforcement Learning to Spider the Web Efficiently. In *Proceedings of the Sixteenth International Conference on Machine Learning(ICML-99)*, pages 335–343, 1999.

[22] S.C.Deerwester, S.T.Dumais, T.K.Landauer, G.W.Furnas, and R.A.Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[23] P. Srinivasan, F. Menczer, and G. Pant. A general evaluation framework for topical web crawlers. In *Information Retrieval, Submitted*, Accessed May 2004. http://www.informatics.indiana.edu/fil/Papers/crawl-framework.pdf.