Using a Hidden Markov Model to Learn User Browsing Patterns for Focused Web Crawling

Hongyu Liu
Jeannette Janssen
Evangelos Milios

Technical Report CS-2005-05

June 3, 2005

Faculty of Computer Science
6050 University Ave., Halifax, Nova Scotia, B3H 1W5, Canada

Using a Hidden Markov Model to Learn User Browsing Patterns for Focused Web Crawling

Hongyu Liu [a], Jeannette Janssen [b], Evangelos Milios [c]

[a,c] Faculty of Computer Science, Dalhousie University, Halifax, Canada

http://www.cs.dal.ca/∼hongyu, http://www.cs.dal.ca/∼eem

[b] Dept. of Mathematics and Statistics, Dalhousie University, Halifax, Canada

http://www.mathstat.dal.ca/∼janssen

Abstract

A focused crawler is designed to traverse the Web to gather documents on a specific topic. It can be used to build domain-specific Web search portals and online personalized search tools. The focused crawler must use information gleaned from previously crawled page sequences to estimate the relevance of a newly seen URL.

In this paper, we present a new approach for prediction of the important links to relevant pages based on a Hidden Markov Model (HMM). The system consists of three stages: user data collection, user modelling via sequential pattern learning and focused crawling. In particular, we first collect the Web pages visited during a user browsing session. These pages are clustered, and the link structure among pages from different clusters is used to learn page sequences that are likely to lead to target pages. The learning is done using HMM. During crawling, the priority of links to follow is based on a learned estimate of how likely the page is to lead to a target page. We compare performance with Context-Graph crawling and Best-First crawling and experiments show that this approach performs better than other strategies.

Contents

List of Tables

List of Figures

1 Introduction

With the exponential growth of information on the World Wide Web, there is great demand for efficient and effective methods to organize and retrieve the information available [1, 2, 3, 4]. The imbalance between rapid growth of the Web and limited network and storage resources poses many challenges for generic crawlers and search engines. For example, search engines may find it hard to refresh their index often enough to keep up with the pace of change of the Web. As well, a topic-oriented keyword search is more and more likely to yield scores of more or less relevant results, which means that a user still needs to go through a selection and browsing process before finding the pages most relevant to her. On the other hand, popular topical pages will not fit every user's interests equally well.

As a result, it is likely that generic search engines will become less appealing to a user with specific information needs that do not fall into the most popular categories. Hence there will be a market for topic-specific search tools such as Web search portals that index pages on a particular topic, or personalized search tools that search the Web for pages that fit a particular user profile.

1.1 Focused crawling

The success of topic-specific search tools will depend on the ability to locate topic-specific pages on the Web while using limited storage and network resources. This can be achieved if the Web is explored by means of a *focused crawler*. A focused crawler is a crawler that is designed to traverse a subset of the Web for gathering only documents on a specific topic, instead of searching the whole Web exhaustively. Focused crawlers were first introduced by Chakrabarti *et al.* [5].

In order to find pages of a particular type or on a particular topic, focused crawlers aim to identify links that are likely to lead to target documents, and avoid links to off-topic branches. Regular crawlers use a Breadth-First search strategy in order to download as many pages as possible, while focused crawlers selectively choose links that are quickly leading to target pages. As a result, with a small investment of network resources, a focused crawler is expected to collect high quality domain-specific documents from the Web with respectable coverage at a rapid rate.

The challenge in designing a focused crawler is to predict which links lead to target pages, since the Web is noisy and multi-topic. Sometimes, relevant pages link to other relevant ones. However, it is very common that pages that are not on topic lead to topical pages indirectly. According to a study [6], most relevant pages are separated by irrelevant pages, the number of which ranges from at least 1, to a maximum of 12, commonly 5. A common approach to

focused crawling is to use information gleaned from previously crawled pages to estimate the relevance of a newly seen URL. The effectiveness of the crawler will depend on the accuracy of this estimation process.

1.2 Literature Review

A variety of methods for focused crawling have been developed. The earliest relevant work is the so-called Fish-Search [7]. Here, the Web is crawled by a team of crawlers, which are viewed as a school of fish. If the "fish" find a relevant page based on specified query keywords, they continue looking by following more links from that page. If the page is not relevant, its child links receive a low preferential value. Shark-Search [8] is a modification of Fish-search which differs in two ways: a child inherits a discounted value of the score of its parent, and this score is combined with a value based on the anchor text that occurs around the link in the Web page.

The focused crawler introduced in [5] uses a canonical topic taxonomy. The user specifies a number of starting points, and browses from these pages. The user matches pages to the best categories while browsing. This categorization is used to train a classifier which makes relevance judgements on a document to the topic. A distiller then determines visit priorities based on hub-authority connectivity analysis.

Intelligent crawling with arbitrary predicates is described in [9]. Their method involves looking for specific features in a page to rank the candidate links. These features include page content, URL names of referred web page, and the nature of the parent and sibling pages. It is a generic framework in that it allows the user to specify the relevant criteria. Also, the system has the ability of self-learning, i.e. to collect statistical information during the crawl and adjust the weight of these features to capture the dominant individual factor at that moment.

Similar methods can be found in [10, 11, 12]. These methods have in common that they use a baseline best-first focused crawling strategy combined with different heuristics based on local features extracted from the parent page, such as similarity to a query, in-degree, PageRank, and relevance feedback. This reflects the belief that in the Web graph, relevant pages are likely to link to other relevant pages.

Other methods used to capture path information leading to targets include reinforcement learning [13], Context Graph [14], and genetic algorithm [15].

A algorithm was presented for learning a mapping performed by Naive Bayes text classifiers from the text surrounding a hyperlink to a value function of sum of rewards, the estimate of the number of relevant pages that can be found as the result of following that

hyperlink [13].

A genetic algorithm is used in [15] to explore the space of potential strategies and evolve good strategies based on the text and link structure of the referring pages. The strategies produce a rank function which is a weighted sum of several scores such as hub, authority and SVM scores of parent pages going back k generations.

The Context Graph method, proposed by Diligenti *et al.* [14] explicitly addressed this problem by using backlinks to estimate the link distance from a page to a target page. Their method starts from seed document, follows backlinks to a certain layer, and then builds up a context graph for the seed page. A classifier is constructed for each layer using Naive Bayes algorithm. As a new document is found, it is classified into a layer. Documents classified into layers closer to the target are crawled first. The experiments showed that this approach maintained a higher level of relevance. However, since the Web graph is rapidly mixing with random links from topic to topic within both short and long paths, the assumption that all pages in a certain layer from a target document belong to the same topic described by a set of terms does not always hold.

1.3 Contribution and Outline of the Paper

Unlike all the systems above, our objective is to capture the hierarchical semantic linkage structure of topic. In the Web graph, 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.

Our approach is to use a combination of semantic content analysis and link structure of paths leading to targets by learning from the user's browsing patterns and emulate them to find more relevant pages. We believe that sequential pattern learning from user's browsing behavior on specific topics is beneficial because humans are very good at discriminating between links based on a variety of clues on a page. Semantic content analysis is performed by a clustering algorithm. Documents are grouped semantically to build a concept graph, from which the learning and prediction are done from both content and linkage by training an HMM to recognize sequences of topics that lead to relevant pages. The use of finite-state Markov models helps represent useful context including not only text content, but also linkage relations. For example, in the path $p1 \rightarrow p2 \rightarrow p3 \rightarrow w$, the prediction of link $p3 \rightarrow w$ is based on the observable text content of w combined with features from its ancestor sequence $p1$, $p2$, and $p3$.

Our approach is unique in the following important ways. One is the way we use Hidden 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. When a new document is seen, the prediction is to estimate how many links this document is away from a target. After training, our system may have learned patterns like "University pages are more likely to lead to Research papers than Sports pages". Note this is different from other systems in that we capture semantic associative relations along paths leading to target pages rather than physical link distances to make predictions. 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 the proposed crawler outperforms Best-First strategy and Context-Graph crawling. The learned model is a general model and parameters of the model are adapted to different users searching different topics, so it is broadly applicable. It can be used to build topic portals and it can help individual users perform personalized online search.

The rest of the paper is organized as follows. Section 2 describes the overview of our focused crawling system and Section 2.1 describes how to collect page sequences user browsed to build a concept graph. Pattern learning and the use of HMM for focused crawling are described in section 3. Section 4 describes the detailed crawling algorithm. Experimental results are shown in Section 5. Conclusion and discussion are in Section 6.

2 System Overview

Consider the way a user searches for information on the Web. Usually the surfer has an idea (topic) in her mind before starting search. Then she will go to a search engine like Google or Yahoo, and type topic keywords to start the search. Keyword search typically results in thousands of hits. She will then select some results returned by the search engine and start browsing there. While browsing, she selectively follows links from page to page to identify what she really wants. If she reaches a page she is interested in, she may take time to read, print or save it. The process continues by choosing another link, or going back to the list of search engine results to follow other returned results, or typing new keywords to start a refined topic search. In short, the user refines keyword-based search by combining searches with further browsing to achieve her topic-specific information need. What the user really seeks goes beyond lexical keyword search.

The Web graph is noisy and multi-topic. Off-topic pages often reliably lead to highly

relevant pages. Due to the small world nature of the Web [16], 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 the pages of a web site semantically. For example, university pages are likely linked to department pages, then to pages of faculty members; they are rarely linked to sports canoeing pages. In other words, dominant semantic topic hierarchies exist. User surfing behavior is likely to follow an intuitive understanding of such semantic topic hierarchies. The decision by the user to follow or ignore a specific link is based on a variety of clues on the page and humans are very good at making such decisions.

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

Focused crawlers can only use information 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 observable document. Probabilistic models, such as HMM, offer a good balance between simplicity and expressiveness of context.

The system consists of three components: User Data Collection, User Modelling via Pattern Learning, and Focused Crawling, as shown in Fig. 1. The following sections describe each of the components.

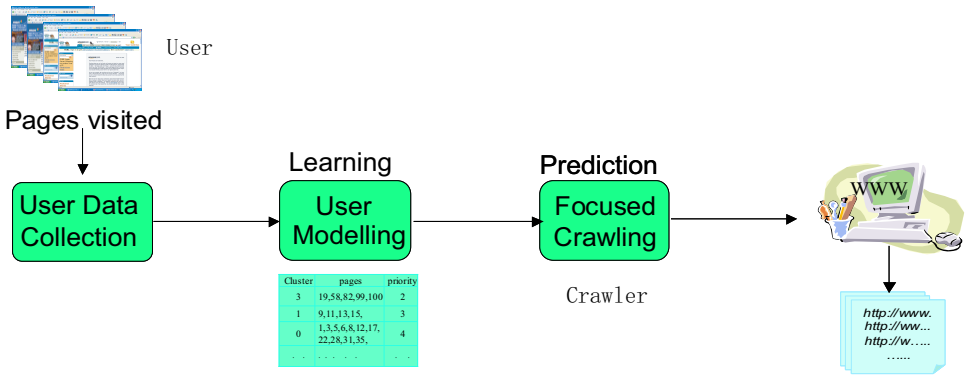

Figure 1: System Architecture: User Data Collection, User Modelling via Pattern Learning, and Focused Crawling.

2.1 User Data Collection

In the User Data Collection stage, we aim to collect the sequences of pages visited by the user during her topic-specific browsing. The user starts browsing with a given topic in mind and is intended to stay on the topic. If the user considers the current page interesting, she can click the ***Useful*** button which is added to each page (Fig. 2a), and the page will become an annotated target page. In order to represent the user browsing pattern, we construct a web graph (Fig. 2b). Web graph is a directed graph whose nodes correspond to pages on the

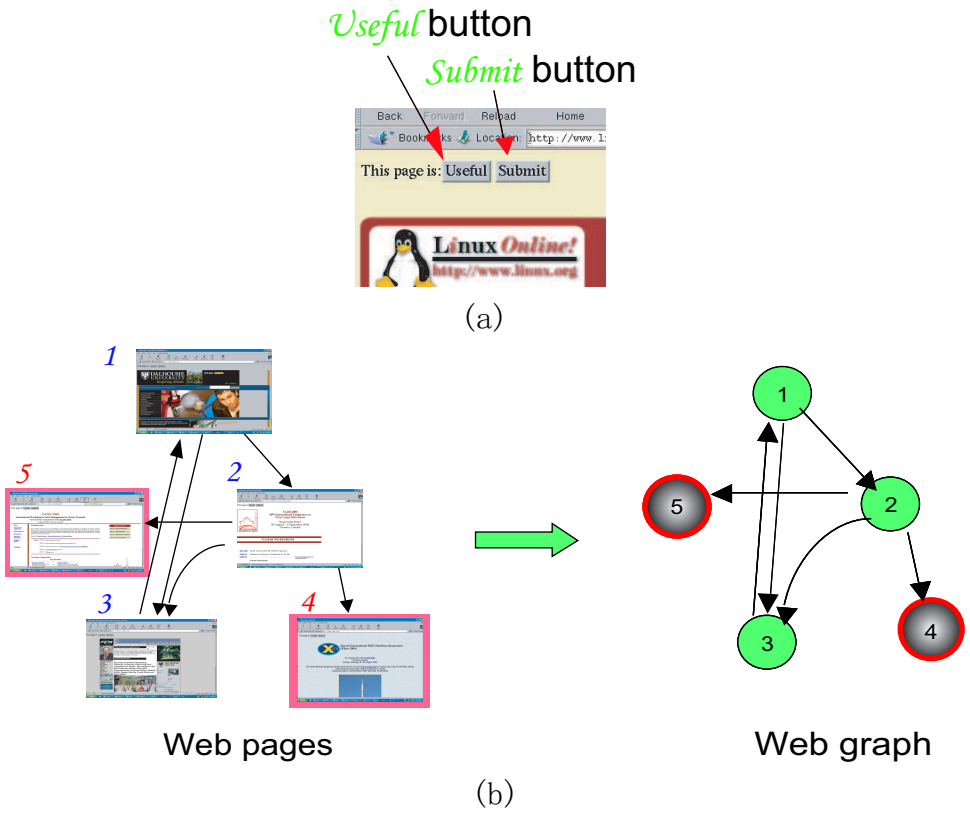

Figure 2: User Data Collection. Red circles represent target pages. (a) Useful and Submit buttons. (b) User visited pages form web graph.

Web, and whose arcs correspond to links between these pages. To construct a web graph, each browsed page is represented by a node, and all hyperlinks between pages are added as edges between the nodes (Fig. 2b). When a user requests a web page by typing a URL in the location bar, choosing a URL from a bookmark, or following a link to reach a new web page, a new node will be created and the corresponding links will be added into the web graph. For example, if the user follows the hyperlink in page 2 to access page 3 in Fig. 2, node 3 is created and edges from $2 \rightarrow 3$ is added. Each node is associated with a browsed

HTML page with a unique URL, and an edge is established if a referring document has a hyperlink pointing to a referred document. Nodes corresponding to targets are marked as red-circled nodes in the web graph, for instance nodes 4 and 5. In the case of web pages with frames, when a frame page is being requested, all of its children pages will be automatically downloaded by the browser. Therefore, instead of recording the frame page, all of its pages will be recorded in the web graph.

The web graph thus captures the link structure of the pages visited by the user. The way the user browses reflects the content and linkage structure leading to her targets, which is the information we try to capture. For example, path $1 \rightarrow 2 \rightarrow 4$ in Fig. 2(b). Sometimes, the user may follow some links to unrelated pages and go 'back' from them, for example, using the "Back" button on the web browser. In this case, the page information, not path information, is still stored in the graph. For example, user follows path $1 \rightarrow 2 \rightarrow 3$ in Fig. 2(b), then finds that page 3 is totally unrelated, so she goes 'back' to page 2. In this case, page 3 is still saved as part of the training data, but the move from page 3 to page 2 does not translate into a link between these pages.

Sometimes, the user may not make the best decision and follow a longer path to reach a target page, so we store other possible links between nodes. Continuing the example above, if the user follows a hyperlink in page 2 to access page 3, and if page 1 links to page 3, then a link from node 1 to node 3 will be added to the web graph as well.

The reasons for considering only visited pages in defining a web graph is to reject noisy hyperlinks that may mislead the focused crawler, and to extract the dominant structure the user follows towards her topic. On the other hand, in addition to sequences of pages which the user visited, recording links between nodes in the web graph reflects linkage structures in the real world, where focused crawling will take place. We argue that the behaviors of the user during topic-specific browsing reveal the user's real information needs.

After completing the browsing session, the user can then submit the entire session by clicking the ***Submit*** button on each browsed page (Fig. 2a). The entire web graph consisting of all user visited pages, together with the user's annotated target information, will be saved as training data.

In our implementation, we use the Algorithmic Solutions Software LEDA package [16] to save the graph with a hash table for URLs and their corresponding document ID numbers.

2.2 Concept Graph

To capture the semantic relations from the user's browsing session, we first categorize the web pages visited by the user into different types. To do this, we first use Latent Semantic

Indexing (LSI) [17, 18] to identify semantic content, then apply a clustering algorithm, then use cluster information and the user web graph to build a concept graph. Finally a HMM is built to learn sequential patterns leading to targets from the concept graph.

2.2.1 LSI – Identification of Semantic Content

A topic the user browses on can be characterized by the keywords contained in web pages on the topic. The same words may appear in different pages related to different topics, so we bring context into consideration by representing documents using LSI. The documents in the web graph are processed to extract the semantic relationships among words in the collection. Latent Semantic Indexing (LSI) [17, 18] exploits the semantic relationships among different words on the basis of co-occurrences in the document collection. It uses the singular value decomposition (SVD) to reduce the dimensions of the term-document space. Its effectiveness has been demonstrated empirically in many information retrieval applications leading to increased average retrieval precision. The main drawback of LSI is that the computation cost for large document collections is high. However, in our system, user visited pages form a small collection with limited topic categories, thus LSI performance limitations are less of an issue.

In detail, we implement document representation using the following steps:

1. Form the dictionary of the collection by extracting all meaningful content words from each document, ignoring stop words, stemming, converting all characters to lower case, and truncating words with more than 20 characters;

2. For each document and term from the dictionary, calculate its normalized TF-IDF (Term Frequency, Inverse Document Frequency) score;

3. Construct the term-document matrix using the TF-IDF scores as entries;

4. Apply the SVD algorithm to obtain a low dimensional LSI space representation for the document collection.

The normalization in the weighting step 2 is a scaling step designed to keep large documents with many keywords from overwhelming smaller documents. The normalized term weights for term i in document k is

$$W_{ik} = \frac{tf_{ik}(\log_2 \frac{N}{df_i} + 1)}{\sqrt{\sum_{k=1}^{T}(tf_{ik})^2(\log_2 \frac{N}{df_i} + 1)^2}} \tag{1}$$

where,

tf_{ik} = the frequency of term i in document k

N = the number of documents in the collection

df_i = the number of documents in which term i appears

T = the number of terms in the collection

In step 4, singular value decomposition breaks the term-document matrix A down into the product of three matrices,

$$A = U\Sigma V^T, \tag{2}$$

where the columns of matrices U and V are the left and right singular vectors, respectively, and Σ is a diagonal matrix of the singular values of A sorted in decreasing order. The closest rank-k approximation to the original matrix A is constructed from the k-largest singular values. The number of reduced dimensions k and the associated singular values and matrices U_k and V_k are determined by means of an iterative process using Lanczos algorithm [19, 20] with tolerance of the approximation 10^{-6}. The implementation provided in [17] is used. Within the reduced space, semantically related terms and documents presumably lie near each other so that clusters of terms and documents are more readily identified.

2.2.2 Clustering

After applying LSI, pages are grouped together according to types and a concept graph is created. In order to do this, a clustering algorithm is applied.

An important question is to find the right number of clusters in the data, therefore, we apply the X-means algorithm [21], which extends k-means with efficient estimation of the number of clusters within a given interval. We first normalize the reduced LSI document vectors, then we cluster all documents, except the pages marked as target by the user, using the X-means algorithm, where the number of clusters is in the interval [3, 8]. All user marked target pages form a separate target cluster (cluster 0). They are not part of the clustering process. By clustering documents we try to capture the semantic relation between different web pages (clusters) leading to the target pages (cluster 0).

To give an idea of the type of pages contained in each of the clusters, we have extracted the 30 words most highly associated with each cluster, in terms of the log odds ratio and mutual information. The results are given in Appendix.

We might use topic categorization instead of clustering based on pre-existing categories

available online in the Open Directory Project(ODP)[1] or Yahoo! [2]. However, compared to user visited pages produced by topic-specific browsing, the categories in the ODP are too general. Furthermore, standard categorization in this way lacks page context information. In fact, our goal is to be able to capture the concept associative relation between different types of documents, i.e., 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, as shown in Fig. 3(b). In the concept graph, we did not merge different nodes with the same cluster number, since doing that would cause linkage information between nodes to become lost.

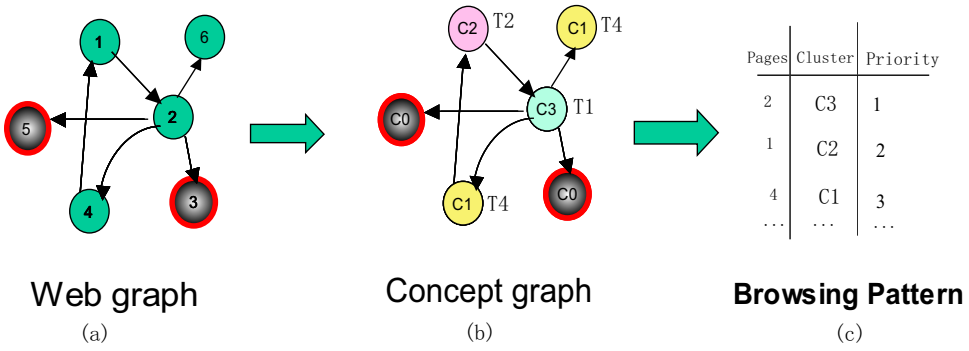

Figure 3: User Modelling via Sequential Pattern Learning. C_i is the label of cluster i, T_j is the estimated hidden state.

3 User Modelling

We model focused crawling in the following way: We assume that there is an underlying Markov chain of hidden states, from which the actual topics of the document are generated. Each state is defined by the distance, in number of hops, of the current page to a target page. In other words, hidden states represent the closeness of a page to a target, and the actual document is the observation that is probabilistically generated. Given a sequence of observations, we predict the next state the page will be in based on the observations so far. During the crawling, those pages with lower state subscript such as T_1 will be assigned higher visit priorities than those with higher state subscript such as T_2, and those pages with

[1]http://dmoz.org
[2]http://www.yahoo.com

the same state are then sorted by the prediction scores, as shown in Fig. 3(c). The pattern learning procedure involves the estimation of the parameters of the underlying probabilistic model (HMM), and details on HMM will be discussed in Section 3.2.

A HMM [22] is defined by 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 $P(o_k|s_j)$. There is also an initial state distribution $P_0(s)$. HMMs, widely used in speech-recognition and information extraction, provide superior pattern recognition capabilities for sequential patterns. HMMs are useful when one can think of underlying unobservable events probabilistically generating surface events, that are observable.

3.1 Structure of the HMM for Focused Crawling

In the HMM used for focused crawling, the hidden states reflect the topology of the Web graph with respect to the target pages, while the observations reflect the content of the Web pages. The model aims to estimate the likelihood of topics leading to a target topic directly or indirectly. The structure of the Hidden Markov Model is shown in Fig. 4.

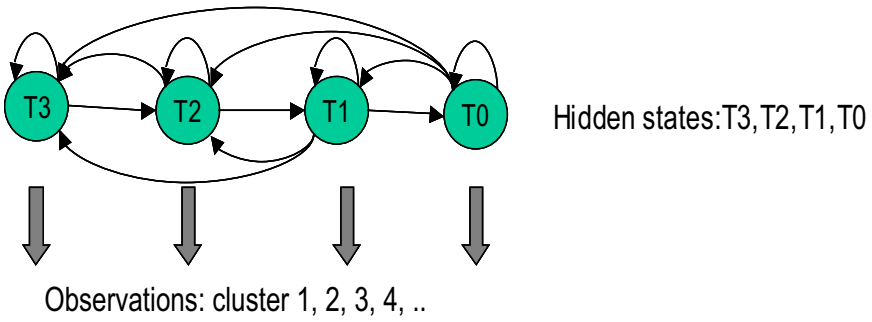

Figure 4: The structure of a Hidden Markov Model with 4 hidden states.

Let n be the number of hidden states. The key quantities associated with the model are the hidden states, observations, and the parameters (initial probability distribution, transition and emission probabilities).

- Hidden states: $S = \{T_{n-1}, T_{n-2}, ..., T_1, T_0\}$
  - The focused crawler is assigned to be in state T_i if the current page is i hops away from a target. The state T_{n-1} represent "$n-1$" or more hops to a target page. The number of hidden states n is a parameter of the model.

- Observations: $O = \{1, 2, 3, 4, ..., x\}$
  - Cluster number $(1..x)$ of an observed web page. The number of clusters x is determined by the X-means algorithm.
- Set of HMM parameters θ:
  - Initial Probability Distribution
    $\pi = \{P(T_0),\ P(T_1),\ P(T_2),\ ...,\ P(T_{n-1})\}$.
    The probabilities of being $0, 1, ...n-1$ hops away from a target page at the start of the process. Due to lack of prior information, we use the uniform distribution $\pi = \{1/n, 1/n, \ldots, 1/n\}$.
  - Matrix of Transition Probabilities: $A = [a_{ij}]_{n \times n}$, where,
    a_{ij} = probability of being in state T_j at time *t+1* given that the system is in state T_i at time *t*.
  - Matrix of Emission Probabilities: $B = [b_{ij}]_{n \times x}$, where,
    b_{ij} = probability that the current page belongs to cluster j if the system is in state T_i.

3.2 Parameter Estimation

Once the state-transition structure is defined, the model parameters are the state transition and the emission probabilities. We use annotated training data, i.e. the concept graph with the identified target pages (see Section 2.2). 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. 5.

Probabilities are estimated by maximum likelihood as ratios of counts, as follows:

$$a_{ij} = \frac{|L_{ij}|}{\sum_{k=0}^{n-1} |L_{kj}|}, \tag{3}$$

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

$$b_{ij} = \frac{|N_{ij}|}{\sum_{k=1}^{x} |N_{kj}|}, \tag{4}$$

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

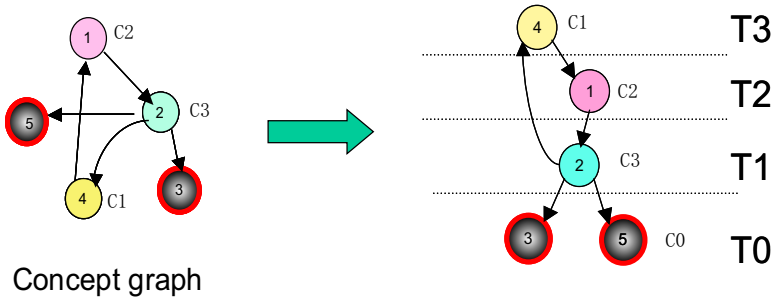

Figure 5: Parameter Estimation of HMM.

3.3 Efficient Inference

The task during the crawling phase is to associate a priority value with each URL that is being added to the queue. Inference using the HMM model is performed for this task. Given a downloaded web page w_t, the URLs associated with its outgoing links are inserted into the queue and they all receive the same priority value, computed on the basis of the prediction of the state of the children of w_t.

The estimated distribution of the state of w_t, $P(S_t|O_{1..t})$, is a probability distribution over all possible state values $T_0, T_1, T_2, .., T_{n-1}$, given observations $O_{1..t}$ and the distribution at step $t-1$, corresponding to the parent w_{t-1} of w_t. This estimation problem is known as filtering. Parent w_{t-1} of w_t is the web page containing the URL of w_t, which was inserted earlier into the queue. The sequence $1..t$ refers to the visited pages along the shortest path from the seed page to the current page. The sequence does not necessarily reflect the temporal order in which pages have been visited.

Based on $P(S_t|O_{1..t})$, a prediction step is carried out to estimate the distribution of the state of the children $t+1$, which have not been seen yet, and therefore there is no observation O_{t+1} associated with them.

The probability distribution $P(S_{t+1}|O_{1..t})$ of the state at the next step $t+1$ given observations $O_{1..t}$ can be calculated using the following equations:

$$P(S_{t+1}|O_{1..t}) = P(S_{t+1}|S_t)P(S_t|O_{1..t}) \tag{5}$$

$$P(S_t|O_{1..t}) = P(O_t|S_t)P(S_t|S_{t-1})P(S_{t-1}|O_{1..t-1}) \tag{6}$$

where $P(S_{t+1}|S_t)$ are the transition probabilities, and $P(O_t|S_t)$ are the emission probabilities of the model.

For the prediction step, we associate values $\alpha(s_j, t)$, $j = 1..n$, with each visited page. Forward value $\alpha(s_j, t)$ is the probability that the system is in state s_j at time t, based on all

observations made thus far. Hence the values $\alpha(s_j, t)$ are the calculated values of $P(S_t|O_{1..t})$. Given the values $\alpha(s_j, t-1)$, $j = 1..n$ of the parent page, we can calculate the values $\alpha(s_j, t)$ using the following recursion, derived from Equation 6:

$$\alpha(s_j, t) = b_{jk_t} \sum_{s_i} (\alpha(s_i, t-1) a_{ij}) \tag{7}$$

where a_{ij} is the transition probability of state T_i to T_j from matrix A and b_{jk_t} is the emission probability of O_{k_t} from state T_j from matrix B. All the $\alpha(s_j, t)$ values for all $j = 1..n$ are stored for each item of the priority queue.

From $\alpha(s_j, t)$, we can calculate the prediction for the future state S_{t+1} using Equation 5. Let $\alpha'(s_j, t+1)$ be the probability that the system will be in state s_j at the next time step, given the observations thus far.

$$\alpha'(s_j, t+1) = \sum_{s_i} (\alpha(s_i, t) a_{ij}) \tag{8}$$

The prediction for next step involves only the transition probabilities because there is no additional evidence.

4 Focused Crawling

After the learning phase, the system is ready to start focused crawling. An overview of the crawling algorithm is shown in Fig. 6. The crawler utilizes a queue, which is initialized with the starting URL of the crawl, and keeps all candidate URLs ordered by their visit priority value. We use a timeout of 10 seconds for web downloads and filter out all but pages with text/html content. The crawler downloads the page pointed to by the URL at the head of the queue, calculates its reduced dimensionality (LSI) representation, and extracts all the outlinks. Then all child pages are downloaded and classified using the K-Nearest Neighbor algorithm into one of the clusters. The prediction of future state is calculated for each parent-child pair based on the current observations and corresponding HMM parameters, and the visit priority values are assigned accordingly. The crawling algorithm with efficient inference using HMM is described in detail in section 4.3.

Since the queue is always sorted according to the visit priority value associated with each URL, we expect that URLs at the head of the queue will locate targets more rapidly.

We also set a relevant threshold γ for determining if a web page is relevant to the user's interests. If its maximal Cosine similarity to the target set is greater than γ, the URL is stored and presented to the user as a relevant page.

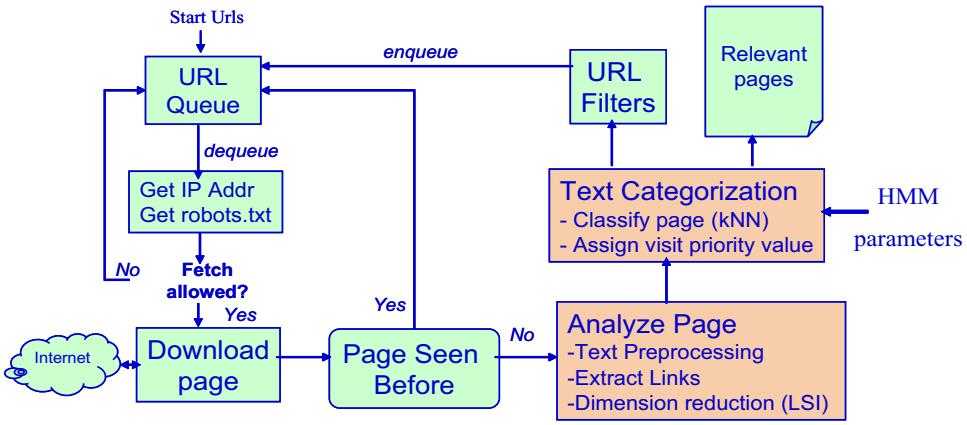

Figure 6: Flow chart of focused crawling.

4.1 Calculation of the Priority

The visit priority is determined based on the estimated future state distribution $P(S_{t+1}|O_{1..t})$, defining a vector key $(P(T_0),\ P(T_1),\ P(T_2),\ ...,P(T_{n-1}))$, where n is the number of hidden states.

If there are two or more items of approximately equal value in the first key, the key data items are ordered in decreasing order according to the second data item to break the tie. In our system, we use a threshold $\varepsilon = 0.001$ to define the equality of two key data items, i.e., for two state distributions $X[P(T_0),\ P(T_1),\ P(T_2),\ ...,P(T_n)]$ and $Y[P(T_0),\ P(T_1),\ P(T_2),\ ...,P(T_n)]$, if $|X[P(T_0)] - Y[P(T_0)]| < \varepsilon$, the sorting process would use the second key data items pair $X[P(T_1)]$ and $Y[P(T_1)]$.

4.2 Priority Queue Data Structure

The priority queue contains all the URLs of pages w_t to be visited sorted by the priority of the parent page w_{t-1} of page w_t, defined as the page through which the URL of w_t was placed on the queue.

Each queue element consists of:

- the URL of page w_t
- cluster C_{t-1} of the parent page w_{t-1} of page w_t
- the visit *priority* of w_t

Algorithm Focused_Crawler(HMM, γ, n)
    $urlQueue$:= {Seed URLs};
    WHILE(not(termination)) DO
        w_t := dequeue head of $urlQueue$;
        extract from w_t its URL, parent cluster C_{t-1}, and
            $\alpha(j, t-1)$ for all possible states j;
        Download contents of w_t;
        Parse and classify page w_t to cluster C_t;
        IF $cos(w_t, TargetSet) > \gamma$, THEN store w_t as relevant;
        Calculate $\alpha(i,t)$ and *priority* for w_t's children
            IF w_t is start Url,
                THEN $\alpha(i,t) = \pi(i)\ b_{iC_t}$;
                ELSE $\alpha(i,t) = \sum_j(\alpha(j,t)\ a_{ji}\ )b_{iC_t}$;
            calculate prediction $[P(T_0), P(T_1), P(T_2), ..., P(T_n)]$;
            calculate the visit *priority*;
        FOR EACH outlink w_{t+1} of w_t with url (has the same priority)
            $urlQueueEntry\ :=\ (priority,\ url,\ C_t,\ \alpha(i,t))$;
            Enqueue ($urlQueueEntry$); // entries sorted by *priority*

Figure 7: Pseudocode of Crawling Algorithm.

- probabilities $\alpha(j, t-1)$ that page w_{t-1} is in hidden state j, for all j, capturing $P(S_{t-1}|O_{1..t-1})$.

4.3 The Algorithm

The basic crawling algorithm is summarized in Fig. 7. Each web page w_t is associated with information about parent in the form of cluster number C_{t-1} and partial probabilities $\alpha(j, t-1)$. If page w_t is a start URL, its $\alpha(.,.)$ will be calculated using the initial matrix, otherwise, it will be calculated based on information of its parent and current observations.

5 Evaluation

5.1 Performance Metrics

It is important that the focused crawler return as many interesting pages as possible while minimizing the portion of uninteresting ones. The standard measures used to evaluate the performance of a crawler are *Precision* and *Recall*.

To evaluate the effectiveness of our approach, we use *precision*, which is the *percentage*

of the web pages crawled that are relevant. Since in general there are multiple target pages marked by the user as interesting, the relevance assessment of a page p we are using is based on 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.

The *recall* of the standard evaluation measure is the ratio of the relevant pages found by the crawler to all relevant pages on the entire Web. It is not possible to get the exact total number of relevant pages on the Web so *recall* cannot be directly measured. However, it is reasonable for the purpose of comparing different retrieval algorithms on a document corpus to estimate *recall* using all relevant pages returned by all versions of crawls strategies and start URLs.

The ability of the crawler to remain focused on on-topic web pages during crawling can also be measured by the average relevance of the downloaded documents [23]. Thus we define another crawling performance measure, the *Maximum Average Similarity* σ.

$$\sigma \quad = \quad \max_{t \in T} \frac{\sum_{p \epsilon S} \cos(p,t)}{|S|} \tag{9}$$

where T is the set of target pages found, S is the set of pages crawled, and

$$\cos(p,t) = \frac{\sum_{k \epsilon p \cap t} w_{pk} \cdot w_{tk}}{\sqrt{\sum_{k \epsilon p} w_{pk}^2 \sum_{k \epsilon t} w_{tk}^2}}$$

is the standard Cosine similarity function, and w_{dk} is the TF-IDF weight of reduced vector term k in document d.

5.2 Experiments

To collect training data, we selected topics from Open Directory Project (ODP)[3] categories, which are neither too broad nor too narrow, as shown in Table. 1. We also compare our focused crawler with Best-First Search (BFS) crawler and Context-Graph (CGS) crawler.

5.2.1 Algorithms for Comparison

We now briefly describe the focused crawling algorithms against which we compare our focused crawler.

1. Best-First Search crawler (BFS):
   This crawler assigns priorities all children of the current page using lexical Cosine sim-

[3]http://dmoz.org/

Table 1: Different Kinds of Crawls

Topics (0)	(1)	(2)	γ	Start URL for the crawler
/Sports/hockey	207	34	0.7	sportsnetwork.com,about.com/sports
/Computers/Software/ Operating_Systems/ Linux	200	30	0.9	www.computerhope.com comptechdoc.org/os,about.com/compute www.informationweek.com/techcenters/sw/
/Sports/biking	697	1	0.5	www.imba.com mountainbikeabout.com
/Computers/Artificial Intelligence/conference/ calls_for_papers	153	4	0.7	www.informatik.uni-trier.de/ ley/db/ conf/index.a.html directory.google.com/Top/Computers/ Computer_Science/Conferences dir.yahoo.com/Science/Computer_Science/ Conferences/

(0): user searches for these topics in Google and starts browsing from the results
(1): # of pages user visited , (2): # of target pages user marked

ilarity between the set of target pages and the content of current page. Since we have multiple targets, the visit priority order of a URL here is based on maximal Cosine similarity, and the priority queue is sorted by this value.

2. Context-Graph crawler (CGS):
   We implemented the focused crawler which is based on the Context Graph [14], in which Naïve Bayes classifiers are built for each layer, and a category "other". We use the likelihood with a threshold 0.5 to assign weakly matching page to "other". When a new page is seen, it will be classified to the winning layer j it belongs to and its visit priority order is sorted by the link distance to layer 0, that is, the closer the layer j is to layer 0, the higher its visit priority. Pages in the same layer are sorted by the likelihood score. For fair comparison of the learning performance, during the focused crawling, we didn't use the extra back-crawling feature which obtains and includes all immediate parents of every relevant page into the queue when it is discovered.

5.2.2 Training data

We now describe the details of collecting training data for the three focused crawlers being compared.

1. Build Web graph using user visited pages for HMM crawling:
   Data are collected from User Data Collection session as described in Section 2.1. Tar-

gets are all pages the user marked as relevant during browsing.

To collect training data, user is given a topic such as *Hockey* and is asked to search for web pages she is interested in related to this topic. To better capture the topical hierarchies for training, the user started browsing from 1 or 2 levels above the given topic in the ODP or from results returned by Google using keywords on the topics 1 or 2 levels above the desired topic in ODP. For example, for topic *Hockey*, she may start from *Sports* level URLs in the ODP, or type *Sports* keywords in Google and start from some of returned URLs. During browsing, user clicks *Useful* button to mark web pages she is interested in, and clicks *Submit* button to save the training data anytime.

2. Build Context Graph using backlinks for Context-Graph crawling:
   Data are collected using Google [24] backlink service, starting from target pages and proceeding up to 4 layers with maximum 300 pages in a single layer. Target pages form layer 0, and layer i contains all the parents of the nodes in layer $i-1$, as shown in Fig. 8. We do not check for links between nodes at the same layer and links between nodes in different layers except neighboring layers, as shown in Fig. 8.

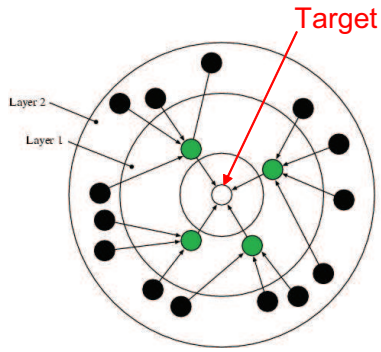

Figure 8: Build context graph.

3. Build Web graph using all nodes from Context Graph for HMM crawling:
   Combination of above two. We use all nodes collected in the context graph to build a web graph with complete links between the nodes.

For HMM crawling, we performed two sets of experiments using the two kinds of training data (no. 1 and 3 above) for comparison. of training data for comparison (No. 1 and 3 above). The motivation is to observe the effect of user topic-specific browsing against the complete web graph on the performance.

5.3 Results

The percentage of relevant pages against the number of pages downloaded for the topic *Linux* is shown in Fig. 9, and the average similarity over all downloaded pages on the same topic is shown in Fig. 10.

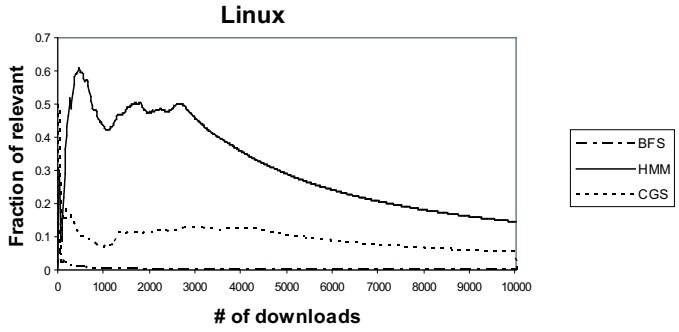

Figure 9: Topic *Linux*: Fraction of relevant pages within the set of downloaded pages.

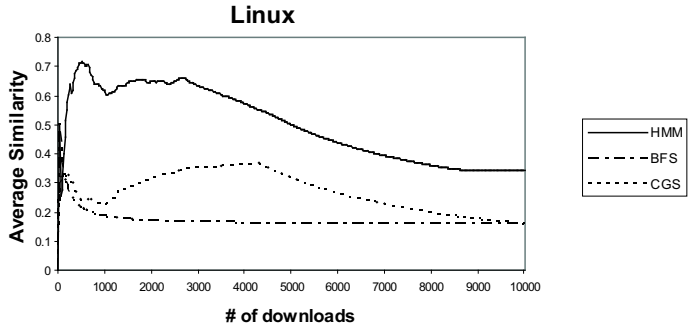

Figure 10: Topic *Linux*: Average maximal similarities to the set of target pages of all downloaded pages.

The system collected the total of 200 user browsed web pages including 30 marked target pages. For the CGS crawler, we used the same target pages as layer 0 and constructed the context graph of depth 4. To evaluate the algorithms, we used the threshold $\gamma = 0.8$. The results show that our system performs significantly better than Context-Graph crawling and Best-First crawling.

In this case, Best-First crawling performs very poorly and when we examine closely Fig. 10, at the very earlier stage, Best-First crawling pursued the links that appear the

most promising (with higher similarity score) at the expense of large longer term loss, whereas our focused crawler explored suboptimal links that eventually lead to larger longer term gains, in spite of penalty at the early stage of the crawl. A typical example during the crawls is that starting from `http://www.comptechdoc.org`, our system follows links quickly leading to `http://www.comptechdoc.org/os/linux/manual` and crawls a lot of links during the same domains, whereas Best-First crawler picks the branch leading to `http://www.iceteks.com/forums/` and crawls many links under the branches that are far away from relevant pages. The URL visit order in Context-Graph crawling is based on the classification score on each layer, so the followed links include `http://www.comptechdoc.org /os/linux/`, `http://www.comptechdoc.org/os/windows/`, `http://www.comptechdoc.org /independent/`.

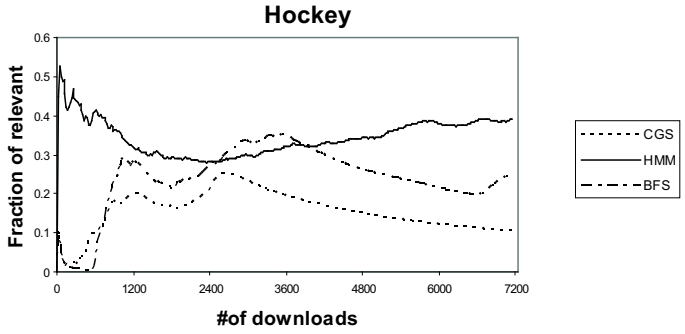

Figure 11: Topic is *Hockey*: Fraction of relevant pages within the set of downloaded pages.

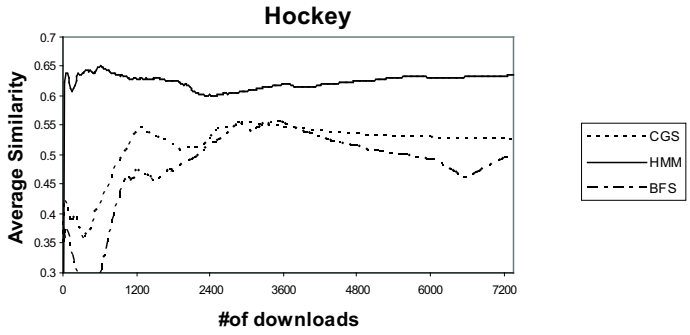

Figure 12: Topic is *Hockey*: Average maximal similarities to the set of target pages of all downloaded pages.

A different topic, *Hockey*, is shown in Fig. 11 and Fig. 12, for which our system still showed very good performance compared to the other two, although Best-First crawler performs

better than Context-Graph crawler overall. The system collected a total of 207 user browsed web pages and 34 marked target pages with the threshold $\gamma = 0.7$. The basic property behind Best-First crawler is linkage locality of the Web, that is web pages on a given topic are more likely to link to those on the same topic. The Context-Graph crawler captures physical layers leading to targets, however, if common hierarchies do not exist for each layer due to mixing links (cross links between topics) in the Web graph, it performs poorly, which reflects our arguments before. In contrast, our model takes both content and linkage structure into account during the learning procedure in order to learn semantic relation to targets, and is shown here to work well on both cases.

Other topics, *Biking* and *Call for papers*, are shown in Fig. 13 and Fig. 14.

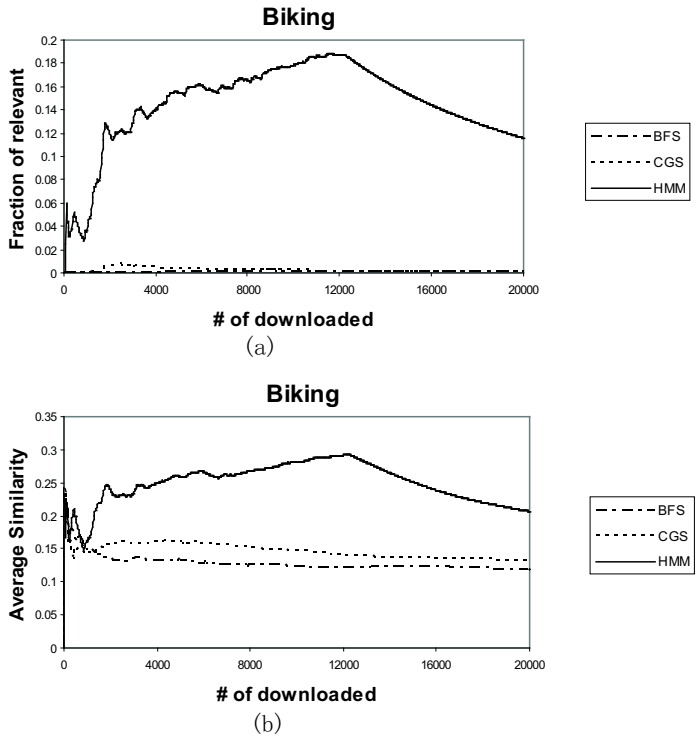

Figure 13: Topic is *Biking*: (a) Fraction of relevant pages within the set of downloaded pages. (b) Average maximal similarities to the set of target pages of all downloaded pages.

It is also worth mentioning that in the original work on Context Graph [14], it was demonstrated that the linkage locality of the Web does boost the performance when including the back-crawling feature: when a page is discovered as a relevant page, its immediate parents are obtained from the search engine and added to the crawling queue automatically. We omitted this feature in order to obtain a fair comparison, since the other two methods

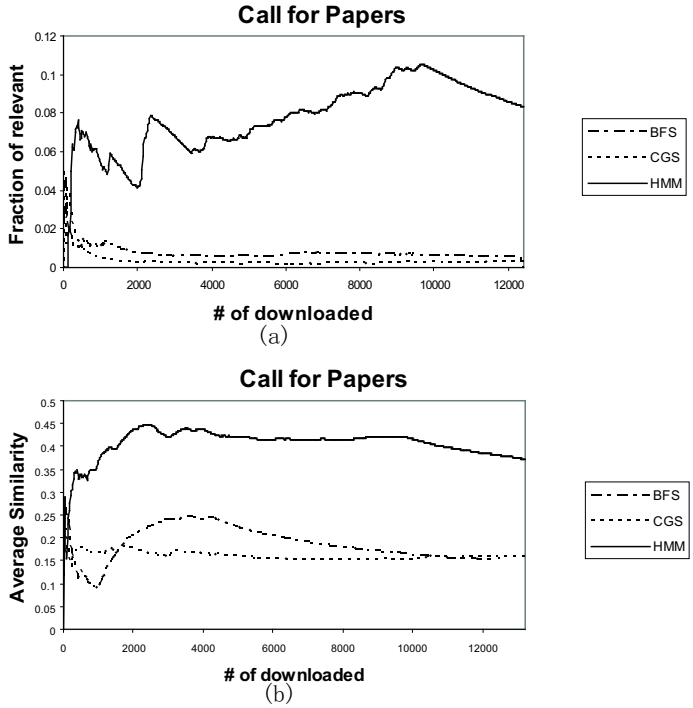

Figure 14: Topic is *Call for papers*: (a) Fraction of relevant pages within the set of downloaded pages. (b) Average maximal similarities to the set of target pages of all downloaded pages.

do not have access to the feature. Apparently our system could be further enriched by combining such features.

Next we observe the effect of the training data on the learning performance. We compared the performance on two types of training data described in Section 5.2.2 for HMM-learning: HMM using user data (HMM), and HMM using all data (HMM_alldata), shown in Fig. 15 and Fig. 16.

In the figures, we also include Context-Graph crawling (CGS) for comparison. The results show that using user visited pages as training data performs best. As we expected, building the web graph from the user's browsing pattern not only extracts the dominant link structure leading to target pages, but also incorporates the judgment of the user about the nature of the web pages. We found that constructing the web graph using all nodes can bring too many noisy links into the learning process, resulting in bad performance.

Now we estimate the recall using all relevant pages returned by all different crawling strategies and start URLs. This is under the assumption that all relevant pages would be found within the combined output of all crawls. The recall scores for all crawls are shown

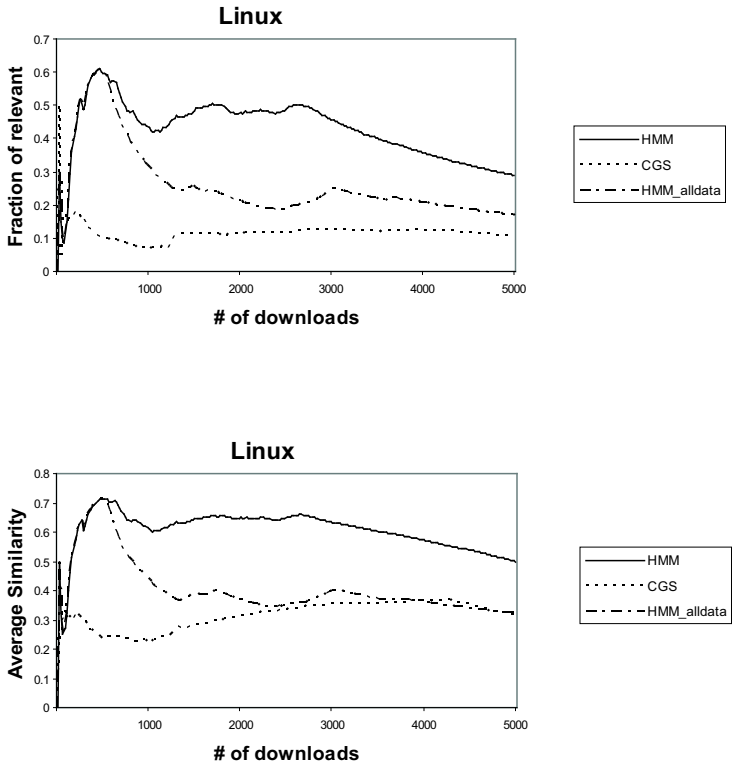

Figure 15: Topic *Linux* with different training data. Top: Fraction of relevant pages. Bottom: Average maximal similarities to the set of target pages of all downloaded pages. HMM: HMM-learning with the web graph using user visited pages (training data No.1); HMM_alldata: HMM-learning with the web graph using all nodes from Context Graph (training data No.3); CGS: Context-Graph learning with context graph using backlinks (training data No.2).

in Table. 2.

6 Conclusion and Discussion

Our contribution is to use HMM to model the link structure and content of documents leading to target pages. Our system is unique in several respects. 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. The prediction is based on hidden semantic content and linkage hierarchy leading to targets instead of only physical link distance. Our experiments indicate that our system

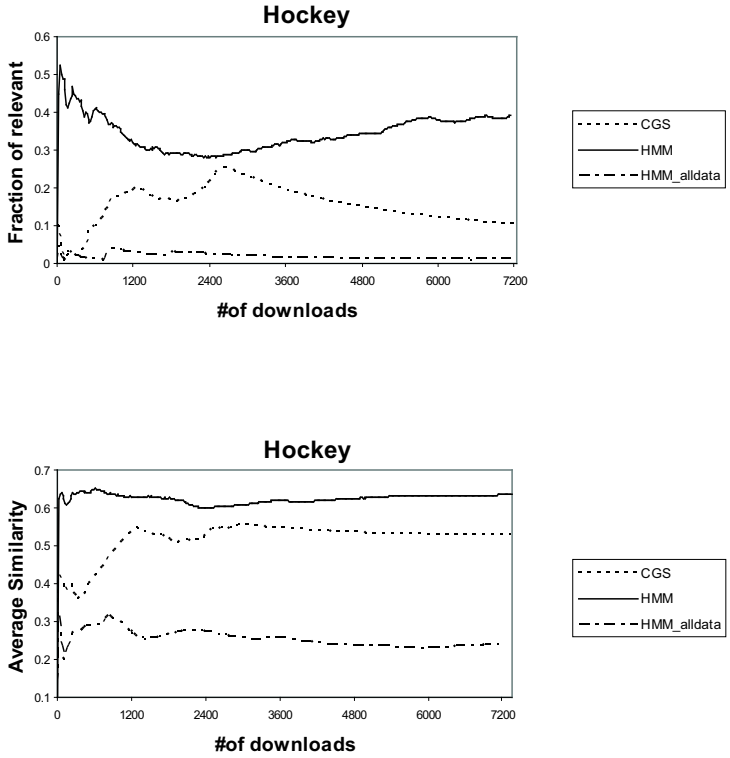

Figure 16: Topic *Hockey* with different training data. Top: Fraction of relevant pages. Bottom: Average maximal similarities to the set of target pages of all downloaded pages. HMM: HMM-learning with the web graph using user visited pages (training data No.1); HMM_alldata: HMM-learning with the web graph using all nodes from Context Graph (training data No.3); CGS: Context-Graph learning with context graph using backlinks (training data No.2).

locates relevant pages quickly and is superior to standard focused crawler.

Our user learning model is versatile, and potentially suitable for different applications. The system does not rely on other search engines and the parameters can be tailored to different users and topics. The proposed approach can potentially lead to focused crawlers that retrieve the information which is the most relevant to the user's interests. The system could be applied to collect user browsing data during daytime, and crawl relevant pages for the user during night time. The model can be used to build topic specific portals, while individual users can rely on it for personal online search.

We are exploring several directions in ongoing work. The proposed way of capturing and exploiting user's personalized interests can potentially lead to more effective focused

Table 2: Recall Evaluation

	topic *Linux*	topic *Hockey*	topic *Biking*	topic *Call for Papers*
HMM	73.25%	52.14%	98%	91.27%
CGS	26.55%	13.58%	0.96%	3.14%
BFS	0.2%	34.28%	1.04%	5.59%

crawlers. Experiments showed that learning from user visited training data works better than using the complete web graph of all nodes. It can be improved by including more and longer path sequences into the training data. In the current user browsing session, to obtain the hierarchy of topics, the user has to browse from the upper level of topical hierarchy and the paths leading to targets are short. It will be nice if the user just starts from Google with the topic keywords as usual, and the system automatically builds the upper topical hierarchy connected with user's browsing data. To make the User Modelling session more practical, we could combine backlinks and user visited pages to better present the topical hierarchy. The user will browse towards her topic in a usual way, typing keywords in Google to start browsing and marking targets as before. When the user marks a target, the system will automatically trace back the path to top page, to collect all backlink sequences of these top pages. The Web graph can be constructed from the top pages using Breadth-First search to targets, or be built from the targets using backlink service of Google API. These top pages are either Google returned results or URLs the user types in the URL bar. We expect that the combination can enhance user's browsing with topical hierarchy structure without much burden for the user. Furthermore, target pages can be updated automatically during the crawl.

The system can be extended for multiple topics. During surfing, the user marks target pages she is interested in without limiting herself to one topic. Training data can be collected from a number of separate browsing sessions. In this case, the web graph can be comprised of several loosely connected separate subgraphs (topics), since the user is doing topic-specific browsing. Accordingly, each subgraph can be clustered separately and patterns can be learned individually for a topic or across topics.

A major area we are investigating is the extension of the feature space. In addition to semantic content, many useful kinds of data such as anchor texts, title/meta data, URL token, and user's query keywords could be included into the learning model. Words "linux", and "hockey" in anchor texts and URLs such as `http://proicehockey.about.com` and `http://justlinux.com/` are important features and provide potential information. We are conducting further improvements and currently exploring more sophisticated models, such

as conditional random fields (CRFs) [25, 26] to richly integrate more features. Including these into the current learning model should be expected to boost the performance.

7 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.

References

[1] P. Lyman, H. Varian, J. Dunn, A. Strygin, K. Swearingen, How Much Information? 2003. Available from <http://www.sims.berkeley.edu/research/projects/how-much-info-2003/>. Link checked March 15, 2005.

[2] R. Zakon, Hobbes' Internet Timeline v7.0, Availabe from <http://www.zakon.org/robert/internet/timeline/>. Link checked March 15, 2005.

[3] Search Engine Sizes, Available from <http://searchenginewatch.com/reports/article.php/2156481>. Link checked March 15, 2005.

[4] Site Position and Coverage, Available from <http://www.silurian.com/sitepos/coverage.htm>. Link checked March 15, 2005.

[5] S. Chakrabarti, M. van den Berg, B. Dom, Focused Crawling: A New Approach to Topic-specific Web Resource Discovery, in: Proceedings of the Eighth International WWW Conference, Toronto, Canada, 1999.

[6] D. Bergmark, C. Lagoze, A. Sbityakov, Focused Crawls, Tunneling, and Digital Libraries, in: Proceedings of the 6th European Conference on Digital Libraries, Rome, Italy, 2002.

[7] P. D. Bra, R. Post, Information Retrieval in the World Wide Web: Making Client-base Searching Feasible, in: Proceedings of the 1th International WWW Conference, Geneva, Switzerland, 1994.

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

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

[10] S. Chakrabarti, K. Punera, M. Subramanyam, Accelerated Focused Crawling through Online Relevance Feedback, in: Proceedings of the 11th International WWW Conference, Hawaii, USA, 1999.

[11] J. Cho, H. Garcia-Molina, L. Page, Efficient Crawling through URL Ordering, in: Proceedings of the 7th World Wide Web Conference, Brisbane, Australia, 1998.

[12] K. Stamatakis, V. Karkaletsis, G. Paliouras, J. Horlock, C. Grover, J. R. Curran, S. Dingare, Domain-Specific Web Site Identification: The CROSSMARC Focused Web Crawler, in: Proceedings of the Second International Workshop on Web Document Analysis (WDA2003), Edinburgh, UK, 2003, Available from < http://www.csc.liv.ac.uk/ wda2003/Papers/Section_V/Paper_17.pdf>. Link checked Mar. 15, 2005.

[13] J. Rennie, A. McCallum, Using Reinforcement Learning to Spider the Web Efficiently, in: Proceedings of the Sixteenth International Conference on Machine Learning(ICML-99), Bled, Slovenia, 1999.

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

[15] J. Johnson, K. Tsioutsiouliklis, C. L. Giles, Evolving Strategies for Focused Web Crawling, in: Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington D.C., USA, 2003.

[16] Algorithmic Solutions,
Available from <http://www.algorithmic-solutions.com/enledabeschreibung.htm>.
Link checked March 15, 2005.

[17] M. W. Berry, LSI: Latent Semantic Indexing Web Site. Available from <http://www.cs.utk.edu/Ĩsi/>. Link checked March 15, 2005.

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

[19] M. W. Berry, Large Scale Singular Value Computations, International Journal of Supercomputer Applications 6 (1992) 13–49.

[20] M. W. Berry, et al., SVDPACKC: Version 1.0 User's Guide, Technical Report CS-93-194, University of Tennessee, Knoxville, TN (October 1993).

[21] D. Pelleg, A. Moore, X-means: Extending K-means with Efficient Estimation of the Number of Clusters, in: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, San Francisco, 2000.

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

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

[24] Google, <`http://www.google.com`>. Link checked March 15, 2005.

[25] J. Lafferty, A. McCallum, F.Pereira, Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, in: Proceedings of the International Conference on Machine Learning (ICML-2001), Williams College, USA, 2001.

[26] D. Pinto, A. McCallum, X. Wei, W. B. Croft, Table Extraction Using Conditional Random Fields, in: Proceedings of the 26th Annual International ACM SIGIR conference, Toronto, Canada, 2003.

A APPENDIX

The following tables show the top 30 words most highly associated with each cluster, in terms of the log odds ratio.

Topic *Hockey*

Cluster 0			
0.01395 hockey	0.00791 team	0.00761 march	0.00735 ice
0.00562 sortable	0.00526 report	0.00486 hl	0.00447 nhl
0.00430 player	0.00391 stick	0.00364 league	0.00334 stats
0.00314 message	0.00313 goalie	0.00285 jhl	0.00269 puck
0.00266 board	0.00230 skater	0.00227 ahca	0.00217 rost
0.00214 wire	0.00210 jose	0.00206 watch	0.00197 junior
0.00183 home	0.00175 shark	0.00175 check	0.00167 star
0.00164 parent	0.00154 cornell		
Cluster 1			
0.05240 bullet	0.00935 chart	0.00923 flow	0.00872 association
0.00852 federation	0.00849 international	0.00844 orange	0.00840 bbc
0.00766 sport	0.00570 line	0.00503 game	0.00447 cricket
0.00422 nation	0.00391 olympics	0.00379 race	0.00340 radio
0.00320 homepage	0.00319 world	0.00299 defence	0.00285 science
0.00269 ahl	0.00266 rugby	0.00261 united	0.00242 state
0.00237 win	0.00234 scotland	0.00222 network	0.00216 wale
0.00216 scca	0.00212 think		
Cluster 2			
0.02319 sport	0.01581 acsm	0.01151 www	0.01137 http
0.00924 description	0.00898 medicine	0.00891 kb	0.00225 fit
0.00223 online	0.00186 query	0.00173 org	0.00166 exercise
0.00164 membership	0.00152 card	0.00149 clinic	0.00147student
0.00139 member	0.00138 outdoor	0.00125 sponsor	0.00122 brochure
0.00121 health	0.00114 public	0.00113 whitewater	0.00113 chapter
0.00106 video	0.00104 raft	0.00101 profession	0.00099 youth
0.00097 result	0.00097 scientific		

Cluster 3			
0.00954 variety	0.00942 final	0.00817 docwrite	0.00807 determine
0.00399 month	0.00398 total	0.00345 boston	0.00333 minw
0.00333 atlantic	0.00318 document	0.00316 write	0.00310 chicago
0.00310 nyra	0.00310 columbia	0.00299 tampa	0.00298 nashville
0.00288 pittsburgh	0.00288 slou	0.00276 vancouver	0.00276 anaheim
0.00276 florida	0.00266 washington	0.00265 edmonton	0.00265 losa
0.00265 dallas	0.00265 calgary	0.00265 ottawa	0.00255 philadelphia
0.00243 phoenix	0.00235 statistics		
Cluster 4			
0.02215 browse	0.00857 item	0.00690 gift	0.00607 hacker
0.00516 supplier	0.00516 ship	0.00495 desk	0.00443 certified
0.00429 affiliate	0.00428 online	0.00347 depart	0.00320 memorabilia
0.00307 ness	0.00307 kentex	0.00285 jewelry	0.00268 mitchell
0.00267 checkout	0.00267 furnish	0.00267 louisvill	0.00267 biederlack
0.00248 clearance	0.00248 riddell	0.00248 steiner	0.00248 vince
0.00248 slugger	0.00232 neil	0.00232 bryant	0.00229 helmet
0.00219 shaquille	0.00219 patriots		

Topic *Linux*

Cluster 0			
0.02608 linux	0.01733 howto	0.01298 lilo	0.01064 boot
0.00992 update	0.00912 kernel	0.00784 install	0.00696 soft
0.00618 lx	0.00596 raid	0.00449 usd	0.00431 configuration
0.00345 root	0.00340 minix	0.00280 system	0.00278 distribution
0.00277 disk	0.00273 describe	0.00246 mode	0.00245 device
0.00224 module	0.00218 hda	0.00216 rdev	0.00212 filesystem
0.00201 section	0.00184 org	0.00184 version	0.00182 hard
0.00180 sysrq	0.00178 emacs		
Cluster 1			
0.00777 linux	0.00719 book	0.00547 partition	0.00463 develop
0.00418 thing	0.00404 author	0.00389 system	0.00345 history
0.00328 unix	0.00320 software	0.00295 command	0.00279 found
0.00264 kernel	0.00259 calendar	0.00241 directory	0.00236 subject
0.00228 consult	0.00215 list	0.00204 computer	0.00201 search
0.00200 program	0.00198 denounce	0.00193 administration	0.00187 base
0.00180 organization	0.00179 inform	0.00174 page	0.00173 fdisk
0.00170 various	0.00169 home		
Cluster 2			
0.01278 talkbacks	0.00997 utc	0.00738 mar	0.00634 feedback
0.00444 read	0.00325 subproject	0.00281 supercomputer	0.00275 microsoft
0.00243 migrating	0.00219 city	0.00202 munich	0.00162 open
0.00156 madison	0.00147 byte	0.00144 march	0.00141 decision
0.00136 cisco	0.00131 fsf	0.00128 grow	0.00128 limux
0.00126 mysql	0.00123 vlans	0.00123 loaf	0.00116 cost
0.00114 concept	0.00114 monday	0.00114 router	0.00114 nt
0.00113 municipal	0.00111 units		
Cluster 3			
0.03064 ximian	0.01571 devel	0.00469 system	0.00447 page
0.00389 gnu	0.00388 bcn	0.00300 web	0.00281 sun
0.00280 internet	0.00264 computer	0.00251 buy	0.00246 network
0.00233 security	0.00230 software	0.00215 gnome	0.00212 csl
0.00208 center	0.00202 kde	0.00199 design	0.00198 user
0.00196 archives	0.00194 operating	0.00192 red	0.00188 complete
0.00186 think	0.00183 project	0.00183 http	0.00178 hat
0.00174 post	0.00173 submit		

Topic *Biking*

Cluster 0			
0.00194 car	0.00185 collision	0.00154 hit	0.00150 avoid
0.00103 left	0.00087 pass	0.00074 turn	0.00069 invisible
0.00067 motorist	0.00065 approach	0.00058 mirror	0.00056 unexpectedly
0.00054 slow	0.00054 ahead	0.00049 driveway	0.00047 sidewalk
0.00045 cross	0.00044 slam	0.00042 diagram	0.00041 headlight
0.00041 move	0.00039 street	0.00036 vest	0.00035 side
0.00034 door	0.00030 lane	0.00027 night	0.00026 wrong
0.00025 absolute	0.00025 scenario		
Cluster 1			
0.01258 device	0.00828 article	0.00806 adventure	0.00771 thing
0.00696 cyp	0.00670 austin	0.00669 rail	0.00603 gift
0.00578 spacer	0.00549 online	0.00538 electric	0.00523 line
0.00523 their	0.00508 shell	0.00502 engine	0.00487 hand
0.00481 www	0.00478 electronic	0.00476 submit	0.00462 model
0.00453 map	0.00440 post	0.00386 large	0.00377 association
0.00372 http	0.00357 tour	0.00353 search	0.00352 yellow
0.00349 people	0.00345 retail		
Cluster 2			
0.07911 helmet	0.01772 law	0.01159 bicycle	0.00856 standard
0.00799 safety	0.00612 astm	0.00513 subcommitte	0.00506 institute
0.00486 pamphlet	0.00462 age	0.00454 children	0.00448 cpsc
0.00423 promotion	0.00397 injury	0.00338 wear	0.00328 require
0.00299 effect	0.00296 sitemap	0.00294 protect	0.00283 test
0.00272 head	0.00263 study	0.00249 quick	0.00239 child
0.00235 statistics	0.00225 mandatory	0.00210 consumer	0.00206 passenger
0.00204 fine	0.00195 manufacturer		

Cluster 3			
0.06200 bicycle	0.03920 club	0.02193 bike	0.01853 coalition
0.01711 cycling	0.01704 san	0.01485 california	0.01253 county
0.01002 association	0.01000 trail	0.00878 diego	0.00861 valley
0.00859 santa	0.00788 mass	0.00730 kiril	0.00723 kunduraziefff
0.00672 cyclist	0.00672 park	0.00671 ride	0.00644 bicyclist
0.00546 beach	0.00540 critic	0.00540 dude	0.00538 alliance
0.00526 match	0.00512 orange	0.00486 caucus	0.00485 wheelmen
0.00473 angel	0.00463 az		
Cluster 4			
0.01402 day	0.01402 wg	0.01254 december	0.01207 october
0.01206 november	0.01190 february	0.01176 january	0.01078 infobong
0.01057 march	0.00826 rss	0.00784 august	0.00770 september
0.00769 april	0.00713 june	0.00663 july	0.00583 blog
0.00570 bloglin	0.00566 anger	0.00546 meta	0.00539 post
0.00535 movable	0.00497 crap	0.00487 recipe	0.00476 syndic
0.00462 preview	0.00451 tech	0.00384 remember	0.00383 yes
0.00380 comment	0.00380 prais		

Topic *Call for papers*

Cluster 0			
0.01066 poster	0.00742 paper	0.00626 submission	0.00477 submit
0.00469 author	0.00457 registration	0.00432 coordinate	0.00420 call
0.00408 ir	0.00399 area	0.00292 accept	0.00216 review
0.00216 format	0.00179 tutor	0.00168 camera	0.00160 electron
0.00150 ready	0.00143 cikm	0.00132 notify	0.00127 template
0.00121 travel	0.00120 ijcai	0.00118 aaai	0.00118 cst
0.00114 require	0.00109 track	0.00109 instruction	0.00097 exhibit
0.00096 noon	0.00093 sig		
Cluster 1			
0.00731 hfo	0.00464 intelligence	0.00420 artificial	0.00414 robot
0.00309 research	0.00297 ai	0.00287 aaai	0.00278 greater
0.00268 problem	0.00265 page	0.00265 unsat	0.00257 family
0.00253 brain	0.00249 conference	0.00233 infinite	0.00199 waterloo
0.00193 download	0.00185 knowledge	0.00180 image	0.00176 kb
0.00174 generic	0.00169 event	0.00167 think	0.00163 consortium
0.00162 neuromaz	0.00161 satisfy	0.00160 project	0.00160 learn
0.00156 data	0.00156 intern		
Cluster 2			
0.01849 ucsc	0.01706 bf	0.01130 conference	0.00983 system
0.00775 computer	0.00770 intern	0.00568 workshop	0.00523 ibm
0.00462 application	0.00444 dunnion	0.00426 intelligent	0.00417 autonomy
0.00415 ssa	0.00412 technology	0.00400 pp	0.00398 deadline
0.00390 university	0.00363 software	0.00351 kb	0.00349 base
0.00328 learn	0.00316 velev	0.00315 proceedings	0.00312 xx
0.00289 download	0.00289 fischer	0.00279 engine	0.00279 autom
0.00265 may	0.00254 verify		
Cluster 3			
0.02805 sat	0.00185 ntab	0.00179 satz	0.00167 ssa
0.00136 power	0.00132 relsat	0.00132 modoc	0.00124 benchmark
0.00123 sato	0.00113 rank	0.00091 mysql	0.00089 script
0.00081 laurent	0.00079 php	0.00079 slow	0.00072 nsat
0.00070 eqsatz	0.00070 csat	0.00066 modify	0.00066 grasp
0.00066 calcr	0.00066 zre	0.00065 back	0.00062 variable
0.00062 heerhugo	0.00060 simon	0.00060 zchaff	0.00058 original
0.00058 file	0.00056 asat		

The following tables show the top 30 words most highly associated with each cluster, in terms of the Mutual information.

Topic *Hockey*

Cluster 0			
0.01152 hockey	0.00671 team	0.00593 march	0.00529 ice
0.00431 report	0.00385 nhl	0.00372 player	0.00365 hl
0.00314 league	0.00282 sortable	0.00258 stats	0.00247 stick
0.00240 message	0.00218 jhl	0.00216 goalie	0.00213 board
0.00192 puck	0.00185 home	0.00183 watch	0.00169 game
0.00164 jose	0.00158 wire	0.00157 junior	0.00150 rost
0.00149 skater	0.00144 star	0.00142 check	0.00141 ahca
0.00141 shark	0.00129 day		
Cluster 1			
0.01242 bullet	0.00303 association	0.00276 international	0.00261 bbc
0.00249 federation	0.00235 orange	0.00229 chart	0.00226 flow
0.00199 line	0.00155 game	0.00155 nation	0.00143 olympics
0.00141 cricket	0.00141 race	0.00120 radio	0.00116 homepage
0.00116 world	0.00105 de	0.00105 science	0.00097 united
0.00096 rugby	0.00093 ahl	0.00091 states	0.00088 sport
0.00085 network	0.00084 win	0.00081 made	0.00081 scotland
0.00077 wale	0.00077 scca		
Cluster 2			
0.04250 sport	0.01524 acsm	0.01119 www	0.01075 http
0.01014 medicine	0.00825 description	0.00800 kb	0.00403 online
0.00357 fit	0.00270 exercise	0.00265 health	0.00248 org
0.00232 result	0.00229 inform	0.00228 membership	0.00221 link
0.00208 member	0.00207 query	0.00200 video	0.00200 outdoor
0.00198 card	0.00195 athlete	0.00194 student	0.00194 public
0.00182 clinic	0.00177 sponsor	0.00176 profession	0.00171 program
0.00169 page	0.00162 youth		

Cluster 3			
0.00365 final	0.00246 variety	0.00223 determine	0.00218 docwrite
0.00132 total	0.00129 month	0.00123 write	0.00115 document
0.00112 boston	0.00109 atlantic	0.00109 minw	0.00103 columbia
0.00103 nyra	0.00103 chicago	0.00103 nashville	0.00100 tampa
0.00100 statistics	0.00098 philadelphia	0.00097 pittsburgh	0.00097 slou
0.00095 florida	0.00095 anah	0.00095 vancouver	0.00095 washington
0.00092 chri	0.00092 edmonton	0.00092 losa	0.00092 dallas
0.00092 calgary	0.00092 ottawa		
Cluster 4			
0.01931 browse	0.01003 item	0.00887 sport	0.00854 online
0.00834 gift	0.00608 hacker	0.00581 ship	0.00581 supplier
0.00572 desk	0.00550 certified	0.00543 affiliate	0.00496 depart
0.00449 memorabilia	0.00375 email	0.00358 ness	0.00358 kentex
0.00348 jewelry	0.00339 mitchell	0.00322 biederlack	0.00322 louisvill
0.00322 furnish	0.00322 checkout	0.00316 helmet	0.00312 clearance
0.00312 riddell	0.00312 steiner	0.00312 vince	0.00312 slugger
0.00304 career	0.00303 neil		

Topic *Linux*

Cluster 0			
0.02083 linux	0.01083 howto	0.00751 boot	0.00735 lilo
0.00693 kernel	0.00678 update	0.00589 install	0.00343 soft
0.00323 configuration	0.00320 lx	0.00311 raid	0.00262 usd
0.00249 root	0.00241 system	0.00228 mini	0.00222 distribution
0.00216 disk	0.00195 describe	0.00184 device	0.00174 mode
0.00164 filesystem	0.00161 module	0.00160 hda	0.00156 section
0.00149 version	0.00149 org	0.00147 think	0.00141 hard
0.00141 rdev	0.00136 available		
Cluster 1			
0.00280 book	0.00218 partition	0.00160 author	0.00158 develop
0.00135 history	0.00116 unix	0.00115 found	0.00112 command
0.00104 calendar	0.00099 directory	0.00097 subject	0.00094 consult
0.00079 administration	0.00077 list	0.00077 denounce	0.00077 organization
0.00073 fdisk	0.00072 apply	0.00070 emul	0.00070 type
0.00070 icon	0.00068 policy	0.00068 library	0.00068 various
0.00067 chapter	0.00065 base	0.00064 think	0.00061 september
0.00060 item	0.00059 pm		
Cluster 2			
0.00767 talkback	0.00725 utc	0.00708 linux	0.00659 mar
0.00577 feedback	0.00529 read	0.00341 microsoft	0.00266 subproject
0.00243 open	0.00241 supercomputer	0.00233 migrating	0.00207 city
0.00187 march	0.00186 munich	0.00159 madison	0.00153 mysql
0.00151 article	0.00150 byte	0.00149 product	0.00147 nt
0.00146 decision	0.00144 fsf	0.00138 grow	0.00138 cisco
0.00135 today	0.00134 gnome	0.00134 train	0.00132 cost
0.00132 source	0.00131 limux		
Cluster 3			
0.00632 ximian	0.00384 devel	0.00145 gnu	0.00143 bcn
0.00131 page	0.00106 sun	0.00097 web	0.00096 buy
0.00093 internet	0.00086 gnome	0.00083 csl	0.00083 center
0.00081 kde	0.00080 security	0.00076 complete	0.00074 design
0.00071 post	0.00070 http	0.00070 network	0.00070 submit
0.00062 red	0.00060 group	0.00059 question	0.00058 xfree
0.00058 printer	0.00057 camera	0.00056 hat	0.00056 boulder
0.00055 project	0.00053 community		

Topic *Biking*

Cluster 0			
0.00920 car	0.00471 hit	0.00466 ride	0.00395 collision
0.00381 left	0.00381 avoid	0.00360 pass	0.00355 turn
0.00301 street	0.00269 they	0.00256 lane	0.00247 make
0.00206 cross	0.00204 motorist	0.00203 side	0.00203 light
0.00203 traffic	0.00202 move	0.00190 way	0.00181 approach
0.00179 done	0.00167 night	0.00164 slow	0.00153 mirror
0.00150 sidewalk	0.00148 invisible	0.00142 because	0.00141 ll
0.00138 door	0.00136 thing		
Cluster 1			
0.00103 device	0.00068 adventure	0.00060 austin	0.00060 rail
0.00053 article	0.00049 online	0.00047 cyp	0.00046 line
0.00043 shell	0.00041 model	0.00041 www	0.00040 map
0.00039 electronic	0.00037 gift	0.00036 submit	0.00036 spacer
0.00036 electric	0.00035 engine	0.00034 association	0.00033 they
0.00033 http	0.00032 hand	0.00031 yellow	0.00031 large
0.00030 tour	0.00029 retail	0.00029 unit	0.00028 people
0.00027 light	0.00027 inform		
Cluster 2			
0.07383 helmet	0.01826 bicycle	0.01659 law	0.00917 standard
0.00892 safety	0.00517 astm	0.00515 institute	0.00473 children
0.00472 cpsc	0.00472 age	0.00424 injury	0.00418 pamphlet
0.00399 promotion	0.00370 subcommitte	0.00367 require	0.00351 wear
0.00329 head	0.00318 protect	0.00313 effect	0.00298 test
0.00275 sitemap	0.00274 study	0.00272 child	0.00263 statistics
0.00254 quick	0.00242 consumer	0.00232 fit	0.00231 passenger
0.00222 manufacturer	0.00219 buy		
Cluster 3			
0.04567 bicycle	0.02370 club	0.01758 bike	0.01363 cycling
0.01035 san	0.00949 coalition	0.00867 county	0.00817 car
0.00734 trail	0.00576 ride	0.00573 cyclist	0.00492 valley
0.00485 santa	0.00480 mass	0.00478 association	0.00452 diego
0.00449 bicyclist	0.00407 park	0.00383 critic	0.00332 orange
0.00323 california	0.00317 beach	0.00309 dude	0.00306 ma
0.00294 kiril	0.00291 kunduraziefF	0.00289 alliance	0.00275 southern
0.00267 angel	0.00264 wheelmen		

Cluster 4			
0.01738 wg	0.01738 dd	0.01451 december	0.01446 november
0.01425 october	0.01418 january	0.01414 february	0.01304 march
0.01081 infobong	0.01016 post	0.00995 april	0.00954 rss
0.00943 august	0.00931 september	0.00907 june	0.00849 july
0.00779 comment	0.00622 may	0.00565 blog	0.00559 bloglin
0.00558 anger	0.00549 meta	0.00545 movable	0.00538 crap
0.00521 recipe	0.00515 syndic	0.00507 preview	0.00505 tech
0.00489 remember	0.00463 image		

Topic *Call for papers*

Cluster 0			
0.01359 paper	0.01226 poster	0.00988 submission	0.00686 call
0.00663 author	0.00658 area	0.00642 submit	0.00547 registration
0.00517 coordin	0.00516 aaai	0.00474 ir	0.00435 accept
0.00363 review	0.00340 format	0.00339 data	0.00315 electron
0.00299 ar	0.00298 tutorial	0.00248 camera	0.00244 program
0.00236 ijcai	0.00223 ready	0.00216 track	0.00215 university
0.00206 present	0.00203 mine	0.00203 cikm	0.00198 notify
0.00197 deadline	0.00192 retrieve		
Cluster 1			
0.00238 hfo	0.00097 robot	0.00094 greater	0.00093 unsat
0.00093 artificial	0.00091 family	0.00087 brain	0.00080 problem
0.00080 infinite	0.00079 page	0.00078 ai	0.00074 intelligence
0.00074 waterloo	0.00067 image	0.00063 consortium	0.00059 qg
0.00057 aaai	0.00057 phase	0.00056 vision	0.00055 neuromas
0.00055 project	0.00054 transition	0.00053 member	0.00051 event
0.00049 satisfy	0.00049 instance	0.00049 acf	0.00049 rb
0.00049 rousu	0.00049 gabri		

Cluster 2			
0.00375 ucsc	0.00300 bf	0.00196 system	0.00119 conference
0.00114 ibm	0.00114 computer	0.00107 international	0.00095 ssa
0.00093 workshop	0.00089 dunnion	0.00089 deadline	0.00089 technology
0.00082 software	0.00082 pp	0.00080 autonom	0.00074 velev
0.00074 application	0.00072 base	0.00069 xx	0.00067 fischer
0.00063 proceedings	0.00062 autom	0.00056 engine	0.00056 verify
0.00056 unit	0.00055 technic	0.00055 state	0.00053 integrity
0.00053 university	0.00052 john		
Cluster 3			
0.06279 sat	0.00697 ntab	0.00691 satz	0.00680 ssa
0.00656 ucsc	0.00641 benchmark	0.00485 power	0.00480 modoc
0.00480 relsat	0.00473 sato	0.00464 rank	0.00418 back
0.00388 solver	0.00294 gener	0.00278 mysql	0.00275 script
0.00268 laurent	0.00265 slow	0.00265 php	0.00259 nsat
0.00257 csat	0.00257 eqsatz	0.00253 zre	0.00253 calcr
0.00253 grasp	0.00253 modify	0.00249 variable	0.00249 heerhugo
0.00247 simon	0.00247 zchaff		