Improving the Performance of Focused Web Crawlers

Sotiris Batsakis[1], *Euripides G.M. Petrakis*[1], *Evangelos Milios*[2]

[1] Department of Electronic and Computer Engineering

Technical University of Crete (TUC)

Chania, Crete, GR-73100, Greece

batsakis@softnet.tuc.gr, petrakis@intelligence.tuc.gr

[2] Faculty of Computer Science

Dalhousie University

Halifax, Nova Scotia, B3H 1W5, Canada

eem@cs.dal.ca

Abstract

This work addresses issues related to the design and implementation of focused crawlers. Several variants of state-of-the-art crawlers relying on web page content and link information for estimating the relevance of web pages to a given topic are proposed. Particular emphasis is given to crawlers capable of learning not only the content of relevant pages (as classic crawlers do) but also paths leading to relevant pages. A novel learning crawler inspired by a previously proposed Hidden Markov Model (HMM) crawler is described as well. The crawlers have been implemented using the same baseline implementation (only the priority assignment function differs in each crawler) providing an unbiased evaluation framework for a comparative analysis of their performance. All crawlers achieve their maximum performance when a combination of web page content and (link) anchor text is used for assigning download priorities to web pages. Furthermore, the new HMM crawler improved the performance of the original HMM crawler and also outperforms classic focused crawlers in searching for specialized topics.

1. Introduction

Crawlers (also known as Robots or Spiders) are tools for assembling Web content locally [1]. Focused crawlers in particular, have been introduced for satisfying the need of individuals (e.g. domain experts) or organizations to create and maintain subject-specific web portals or web document collections locally or for addressing complex information needs (for which a web search would yield no satisfactory results). Applications of focused crawlers also include guiding intelligent agents on the Web for locating specialized information. Typical requirements of such application users are the need for high quality and up-to-date results, while minimizing the amount of resources (time, space and network bandwidth) to carry-out the search task. Focused crawlers try to download as many pages relevant to the subject as they can, while keeping the amount of not relevant pages downloaded to a minimum [2].

Crawlers are given a starting set of web pages (seed pages) as their input, extract outgoing links appearing in the seed pages and determine what links to visit next based on certain criteria. Web pages pointed to by these links are downloaded, and those satisfying certain relevance criteria are stored in a local repository. Crawlers continue visiting Web pages until a desired number of pages have been downloaded or until local resources (such as storage) are exhausted.

Crawlers used by general purpose search engines retrieve massive numbers of web pages regardless of their topic. Focused crawlers work by combining both the content of the retrieved Web pages and the link structure of the Web for assigning higher visiting priority to pages with higher probability of being relevant to a given topic. Focused crawlers can be categorized as follows:

- **Classic Focused Crawlers** [3] take as input a user query that describes the topic, a set of starting (seed) page URLs and they guide the search towards pages of interest. They incorporate criteria for assigning higher download priorities to links based on their

likelihood to lead to pages on the topic of query. Pages pointed to by links with higher priority are downloaded first. The crawler proceeds recursively on the links contained in the downloaded pages. Typically, download priorities are computed based on the similarity between the topic and the anchor text of a page link or between the topic and text of the page containing the link (most likely, they are related to pages on the topic of the query). Text similarity is computed using an information similarity model such as the Boolean or the Vector Space Model (VSM) [4].

- **Semantic Crawlers** [5] are a variation of classic focused crawlers. Download priorities are assigned to pages by applying semantic similarity criteria for computing page-to-topic relevance: a page and the topic can be relevant if they share conceptually (but not necessarily lexically) similar terms. Conceptual similarity between terms is defined using ontologies [5] [6] [7].
- **Learning Crawlers** [8] apply a training process for assigning visit priorities to web pages and for guiding the crawling process. They are characterized by the way relevant web pages or paths through web links for reaching relevant pages are learned by the crawler. Typically, a learning crawler is supplied with a training set consisting of relevant and not relevant Web pages which is used to train the learning crawler [8] [9]. Higher visit priority is assigned to links extracted from web pages classified as relevant to the topic. Methods based on *Context Graphs* [10] and *Hidden Markov Models* (HMM) [11] take into account not only the page content and the corresponding classification of web pages as relevant or not relevant to the topic but also the link structure of the Web and the probability that a given page (which may be not relevant to the topic) will lead to a relevant page within a small number of steps (hops). Hybrid methods [12] suggest combining ideas from learning crawlers with ideas of classic focused crawlers.

In this paper, state-of-the-art approaches for building topic driven focused crawlers are considered and evaluated, with particular emphasis on learning crawlers. The approach to learning crawlers presented in this work is inspired by previous work on Hidden Markov Model (HMM) crawlers [11] for learning paths leading to relevant pages, and from the most successful designs of classic focused crawlers [3] [13] using information from both page content and link anchor text for estimating download priorities to candidate pages. The crawlers mentioned above (and several variants of them) have all been implemented and their performance has been compared using several example topics.

The contributions of this work are the following:

a) It presents a critical evaluation of various existing approaches to web crawling, including classic and learning focused crawlers (i.e. Breadth-First, Best-First and HMM crawlers).
b) Several novel variants of classic, semantic and learning crawlers are proposed and evaluated. Classic focused crawlers combining page content and anchor text, and semantic crawlers using general purpose term taxonomies (for searching for topics which are not only lexically but also conceptually similar to a given topic) are evaluated.
c) A new approach to learning paths reaching web pages relevant to a given topic is proposed inspired by a recent contribution [11] on an HMM crawler. In particular, this work is enhanced with ideas from classic focused crawlers (for better estimation of page content relevance with the content of the topic) combined with ideas from document clustering for learning promising paths leading to target pages.

The rest of this paper is structured as follows: Section 2 reviews work on focused crawlers emphasizing on state-of-the-art approaches to Best-First, semantic and learning crawling. Issues related to the design and implementation of crawlers, along with the presentation of the new learning crawler proposed in this work, are presented in Section 3. A critical evaluation

of the crawlers considered in this work is presented in Section 4 followed by conclusions and issues for further research in Section 5.

2. Related Work and Background

Commercial search engines use generic topic-independent crawlers for building their index and Web page repository for responding to user queries. GoogleBot[1], Slurp[2], MSNBot[3] and Teoma[4] are examples of such crawler implementations. Methods for implementing search engine crawlers include breadth-first search, and page importance (using back-link counts or *PageRank* [14]).

2.1 Focused Crawlers

Topic oriented crawlers attempt to focus the crawling process on pages relevant to the topic. They keep the overall number of downloaded Web pages for processing [33] [34] to a minimum, while maximizing the percentage of relevant pages. Their performance depends highly on the selection of good starting pages (seed pages). Typically users provide a set of seed pages as input to a crawler or, alternatively, seed pages are selected among the best answers returned by a Web search engine [32], using the topic as query [15] [16] [31]. Good seed pages can be either pages relevant to the topic or pages from which relevant pages can be accessed within a small number of routing hops. For example, if the topic is on scientific publications, a good seed page can be the publications page of an author (laboratory or department). Alternatively, a good seed page can be the personal web page of the author (the

[1] http://www.google.com

[2] http://www.yahoo.com

[3] http://www.msn.com

[4] http://www.ask.com

home page of a laboratory or department respectively); although the last may contain no publications at all, it is known to lead to pages containing publications.

The Fish-Search approach [17] assigns binary priority values (1 for relevant, 0 for not relevant) to pages candidate for downloading by means of simple keyword matching. Therefore, all relevant pages are assigned the same priority value. The Shark-Search method [13] suggests using Vector Space Model (VSM) [4] for assigning non binary priority values to candidate pages. The priority values are computed by taking into account page content, anchor text, text surrounding the links and the priority value of parent pages (pages pointing to the page containing the links). Additional approaches to focused crawling include InfoSpiders and Best-First Crawler [3]. InfoSpiders use Neural Networks, while Best-First Crawlers assign priority values to candidate pages by computing their text similarity with the topic by applying VSM [4]. Shark-Search can be seen as a variant of Best-First crawler with a more complicated priority assignment function.

Best-First Crawlers use only term frequency (*tf*) vectors for computing topic relevance. The use of inverse document frequency (*idf*) values (as suggested by VSM) is problematic since it not only requires recalculation of all term vectors at every crawling step but also, at the early stages of crawling, *idf* values are highly inaccurate because the number of documents is too small. Best-First crawlers have been shown to outperform InfoSpiders, and Shark-Search as well as other non-focused Breadth-First crawling approaches [3]. Best-First crawling is considered to be the most successful approach to focused crawling due to its simplicity and efficiency.

The *N*-Best First crawler [3] is a generalized version of Best-First crawler: at each step, the *N* pages (instead of just one) with the highest priority are chosen for expansion. Along the same lines, "Intelligent Crawling" [18] suggests combining page content, URL string information, sibling pages and statistics about relevant or not relevant pages for assigning

priorities to candidate pages. This results into a highly effective crawling algorithm that learns to crawl without direct user training.

2.2 Semantic Crawlers

Best-First crawlers estimate page to topic relevance as document similarity. Computing similarity by classical information retrieval models (e.g., using VSM as Best-First crawlers do) is based on lexical term matching: two documents are similar if they share common terms. However, the lack of common terms in two documents does not necessarily mean that the documents are unrelated. For example, two terms can be semantically similar (e.g., can be synonyms or have related meaning) although they are lexically different. Classical crawler methods will fail to associate documents with semantically similar but lexically different terms. Semantic crawlers resolve this problem using term taxonomies or ontologies. In term taxonomies or ontologies conceptually similar terms are related by virtue of IS-A (or other types of) links.

All terms conceptually similar to the terms of the topic are retrieved from the taxonomy or the ontology and are used for enhancing the description of the topic (e.g. by adding synonyms and other conceptually similar terms to the topic). Document similarity can be computed by VSM or by specialized models such as the Semantic Similarity Retrieval Model (SSRM) [6], or the model suggested by Corley et al. [19] which have been shown to outperform VSM [7]. In this work, Best-First crawlers are implemented based on this category of similarity metrics, forming the so called Semantic Crawler methods. Along these lines, topic oriented ontologies have been suggested for finding pages relevant to the topic of interest [5] [20].

2.3 Learning Crawlers

The crawler learns user preferences on the topic from a set of example pages (training set). Specifically the user provides a set of pages and specifies which of them are relevant to the

topic of interest. Training may also involve learning the path leading to relevant pages. The training set may consist either of relevant pages only or of both relevant and not relevant pages. During crawling, each downloaded page is classified as relevant or not relevant and is assigned a priority. Early approaches to learning crawlers use a Naïve Bayesian classifier (trained on web taxonomies such as Yahoo) for distinguishing between relevant and not relevant pages [2]; others suggest using decision trees [9], First Order Logic [21], Neural Networks and Support Vector Machines [8]. In [22] Support Vector Machines are applied to both page content and link context, and their combination is shown to outperform methods using page content or link context alone.

The structure of paths leading to relevant pages is an important factor in focused crawling [23]. Diligenti et al. [10] introduce the concept of "context graphs"; first back links to relevant pages are followed to recover pages leading to relevant pages. These pages along with their path information form the context graph. The original context graph method builds classifiers for sets of pages at distance 1, 2,… from relevant pages in the context graph. The focused crawler uses these classifiers to establish priorities of visited pages, priorities assigned to links extracted from these pages. An extension to the context graph method is the *Hidden Markov Model (HMM)* crawler [11][30]: The user browses the Web looking for relevant pages and indicates if a downloaded page is relevant to the topic or not. The visited sequence is recorded and is used to train the crawler to identify paths leading to relevant pages.

Chakrabarti et al. [24] proposed a two classifier approach. The open directory (DMOZ[5]) Web taxonomy is used to classify downloaded pages as relevant or not, and to feed a second classifier which evaluates the probability that the given page will lead to a target page. An extensive study of Learning Crawlers and the evaluation of several classifiers used

[5] http://www.dmoz.org/

to assign visit priority values to pages is presented in [8]. Classifiers based on Support Vector Machines (SVM) seem to outperform Bayes Classifiers and classifiers based on Neural Networks. Bergmark [25] suggests using document clustering information (e.g., the cluster centroids of training page sets) as topic descriptors.

Hybrid crawlers [12] combine ideas from learning and classic focused crawlers. In the work by Chen [12], the crawler works by acting alternatively either as learning crawler guided by genetic algorithms (for learning the link sequence leading to target pages) or as classic crawler.

3. Crawler Design

Issues related to crawler design are discussed next:

a) Input: Crawlers take as input a number of starting (seed) URLs and (in the case of focused crawlers) the topic description. This description can be a list of keywords for classic and semantic focused crawlers or a training set for learning crawlers.

b) Page downloading: The links in downloaded pages are extracted and placed in a queue. A non focused crawler uses these links and proceeds with downloading new pages in a first in, first out manner. A focused crawler reorders queue entries by applying content relevance or importance criteria or may decide to exclude a link from further expansion (generic crawlers may also apply importance criteria to determine pages that are worth crawling and indexing).

c) Content processing: Downloaded pages are lexically analyzed and reduced into term vectors (all terms are reduced to their morphological roots by applying a stemming algorithm [26] and stop words are removed). Each term in a vector is represented by its term frequency-inverse frequency vector (*tf-idf*) according to VSM. Because computing inverse document frequency (*idf*) weights during crawling can be problematic most Best

First Crawler implementations use only term frequency (*tf*) weights. In this work we used precompiled *idf* weights, provided by the IntelliSearch[6] Web search engine holding *idf* statistics for English terms.

d) Priority assignment: Extracted URLs from downloaded pages are placed in a priority queue where priorities are determined based on the type of crawler and user preferences. They range from simple criteria such as page importance or relevance to query topic (computed by matching the query with page or anchor text) to more involved criteria (e.g. criteria determined by a learning process).

e) Expansion: URLs are selected for further expansion and steps (b) - (e) are repeated until some criteria (e.g. the desired number of pages have been downloaded) are satisfied or system resources are exhausted.

3.1 Best-first crawlers

A best-first crawler assigns download priorities to web pages by computing the document (*d*) similarity between the pages and the topic (*q*). Both the page and the topic are reduced to term vectors and their similarity is computed according to VSM [4]:

$$similarity_{PAGE\_CONTENT}(p) = \frac{\sum_i q_i d_i}{\sqrt{\sum_i q_i^2 \sum_i d_i^2}} \qquad (Eq.1)$$

where q_i and d_i are the *tf-idf* weights of the terms in the vector representations of the topic and the page respectively. The term weights for a document are computed using the following formula:

$$d_i = \underbrace{\frac{f_i}{\max f_d}}_{tf_i} \underbrace{\log \frac{N}{N_i}}_{idf_i} \qquad (Eq.2)$$

[6] http://www.intelligence.tuc.gr/intellisearch/

where d_i is the weight of term i in document d, tf_i *is* the term frequency of term i in document d, idf_i is the inverse document frequency of term i, f_i is the frequency of term i into d, $max\ f_d$ is the maximum frequency of all terms in d, N is the total number of documents and N_i is the number of documents containing term i.

There are several variants of the Best First Crawling strategy, depending on how links in the same page are prioritized:

1) All links in the page receive the same download priority by applying Eq.1 on the topic and page content representations.
2) Priorities are assigned to pages by computing the similarity between the anchor text of the link pointing to the page and the query by applying Eq. 1, leading to a variant of Best-First crawlers (evaluated in this work). In this case, the links from the same page may be assigned different priority values.
3) Combining (1) and (2) is a third alternative (also evaluated in this work): the priority of link i in page p is computed as the average of the two similarities (query to page and query to anchor text):

$$priority(p_i) = \frac{1}{2}(similarity_{PAGE\_CONTENT}(p) + similarity_{ANCHOR\_TEXT}(i)) \quad (Eq.3)$$

The rationale behind this approach is that a page relevant to the topic is more likely to point to a relevant page than to an irrelevant one. On the other hand, anchor text may be regarded as a summary of the content of the page that the link points to. However, anchor text isn't always descriptive of the content of the page the link points to. So the third alternative combines page content and anchor text as Eq.3 suggests.

3.2 Semantic Crawlers

In Semantic Crawlers, all terms conceptually similar to topic terms are retrieved from the ontology. They are used for enhancing the description of the topic and for computing the semantic similarity between the topic and the candidate pages. In this work, WordNet[7] is used for retrieving conceptually similar terms [6] [7]. WordNet is a controlled vocabulary and thesaurus offering a taxonomic hierarchy of natural language terms. It contains around 100,000 terms, organized into taxonomic hierarchies. WordNet provides broad coverage of the English vocabulary and can be used for focused crawling on almost every general-interest topic making our implementation the first general purpose Semantic Crawler.

The similarity between the topic and a candidate page is computed as a function of semantic (conceptual) similarities between the terms they contain. Computing semantic similarity involves computing the similarity between related concepts which are not lexicographically similar [7]. The definition of document similarity depends on the choice of semantic similarity function. In this work the priority of a page *p* is computed as follows:

$$priority_{Semantic}(p) = \frac{\sum_k \sum_l sim(k,l) w_k w_l}{\sum_k \sum_l w_k w_l} \qquad (Eq.4)$$

where *k* and *l* are terms in the topic and candidate page (or anchor text of the link) respectively, w_k, w_l are their term weights and sim*(k,l)* is their semantic similarity. A similar formula was applied to *tf* weight vectors using an ad-hoc term similarity method based on topic specific ontologies in [5]. Topic specific ontologies were also used in [20]. In this work, the semantic similarity between terms is defined, based on WordNet as the underlying taxonomy, as follows:

[7] http://wordnet.princeton.edu

a) Synonym Set Similarity: *sim(k,l)* is 1 if *l* belongs to the synonym set of term *k* into the WordNet taxonomy and 0 otherwise.

b) Synonym, Hypernym/Hyponym Similarity: *sim(k,l)* is 1 if *l* belongs to the synonym set of term *k*, 0.5 if *l* is a hypernym or hyponym of *k* and 0 otherwise.

The download priority of a link is defined as the average of the similarity of the topic description with anchor text and page content respectively, both computed using Eq. 4.

3.3 Learning Crawlers

The Hidden Markov Model (HMM) crawler [11] works by establishing a relation between page content and the path leading to relevant pages. To create a training set of pages, the user browses on a specific topic, starting with a set of seed pages and accumulating a sequence of pages, which he labels as relevant or not. Relevant pages form a cluster (C_0) while non-relevant pages are clustered by applying K-Means [27] (K is user defined) forming clusters C_1 to C_k. A Hidden Markov Model [28] is created based on this clustering: each page is characterized by two states (a) the visible state corresponding to the cluster that the page belongs to according to its content, and (b) the hidden state corresponding to the distance of the page from a target one. Given the cluster the page belongs to, its priority is computed by the probability that crawling it will lead to a target page.

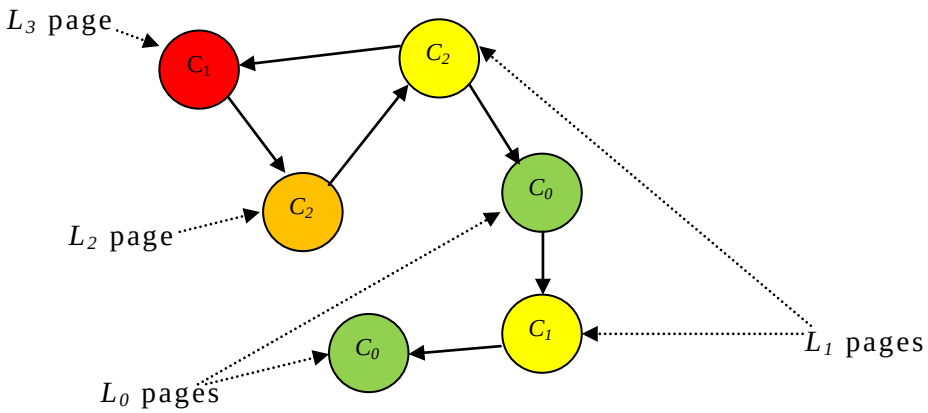

Fig.1:Representation of an HMM training set using distance from target pages (Level L) and clusters of pages in the training set.

A pictorial representation of an HMM training set is shown in Fig.1. L_0 indicates target or level 0 pages, L_1 are level 1 pages (1 link distance from target pages), L_2 are level 2 (2 links away from target pages), and L_3 are 3 or more links away from target pages. C_0, C_1 and C_2 labels correspond to Clusters 0, 1 and 2 respectively. Pages within the same cluster may belong to different levels and (conversely) pages at the same level may belong to different clusters.

The following summarizes the parameters and notation used by an HMM crawler [11]: Web pages are characterized by their level or hidden state L_i, (where i is the level) and by the cluster C_j they belong to (observation). The set of pages hidden states and observations is modeled by a Hidden Markov Model [28].

I. Initial probability matrix: $\pi = \{P(L_0,...,L_{states-1})\}$ where *states* is the number of hidden states and $P(L_i)$ represents the probability of being at hidden state i at time 1. This probability is estimated by assigning to each page a value equal to the percentage of pages with the same hidden state in the training set.

II. Transition Probabilities Matrix: $A = [a_{ij}]_{0 \le i < states, 0 \le j < states}$ where a_{ij} represents the probability of being at state L_j at time t+1 if at state L_i at time t. This probability is assumed independent of t and it is estimated by counting the transitions from state L_i to L_j in the training set, and by normalizing by the overall number of transitions from state L_i.

III. Emission Probabilities Matrix: $B = [b_{ij}]_{0 \le i < states, 0 \le j < clusters}$ where b_{ij} represents the probability of being at cluster C_j given state L_i, and *clusters* is the number of clusters. Probabilities are computed by counting the number of pages in cluster C_j with hidden state L_i and by normalizing by the overall number of pages in hidden state L_i.

Components π, A and B are computed prior to crawling based on the training set. The crawler downloads pages, extracts their vector representations (according to VSM) and assigns each page to a cluster using the *K-Nearest Neighbors* algorithm [29]. Given the cluster of a page, the probability that this page will lead to a target page is computed using the HMM model parameters. This probability corresponds to the visit priority of a link contained in that page. Using the HMM model parameters a prediction of the state in the next time step is computed, given the sequence of web pages observed. In order to calculate the prediction value, each visited page is associated with values $a(L_j,t)$, $j=0,1,..,states$. Value $a(L_j,t)$ is the probability that the crawler downloads a page with hidden state L_j at time t. Given values $a(L_j,t-1)$ of parent pages, values $a(L_j,t)$ are computed using the following recursion:

$$a(L_j,t) = b_{jc_t} \sum_{i=0}^{states} (a(L_i,t-1) \cdot a_{ij}) \quad \text{(Eq.5)}$$

where a_{ij} is the transition probability from state L_i to L_j in matrix A and b_{jc_t} is the emission probability of cluster c_t from hidden state L_j in matrix B. Values $a(L_j,0)$ at the final recursion step are taken from initial probability matrix π. Given values $a(L_j,t)$ the probability that in the next time step the selected page will be in state L_j is computed as follows:

$$a(L_j,t+1) = \sum_{i=0}^{states} (a(L_i,t) * a_{ij}) \quad \text{(Eq.6)}$$

The probability of being at state L_0 (relevant page) in the next step is the priority assigned to pages by the HMM crawler [11]. If two clusters yield identical probabilities (i.e., the difference of their probabilities is below a predefined threshold ε) then higher priority is assigned to the cluster leading with higher probability to target pages in two steps (also computed by applying Eq.5 and Eq.6).

Notice that the work in [11] suggests pages associated with the same cluster sequence in the path leading to them are assigned the same priority. In the proposed crawler the priority

score of a page is defined as the average of priorities computed by the HMM crawler [11] and by the similarity of the centroid vector representing the cluster of relevant (positive) pages with the term vector representation of the page. Two variations of the proposed HMM crawler are proposed, using page content alone, or using both page content and anchor text. Fig. 2 summarizes the operation stages of the proposed crawler:

Input: Training set, candidate page (p).

Output: priority value $priority_{learningHMM}(p)$ assigned to candidate page p.

1. Cluster training set using K-Means (K is user defined).
2. Compute π, A, B matrixes and centroid c_r of relevant pages.
3. Classify candidate page p to a cluster c_t using K-Nearest Neighbor algorithm
4. Compute hidden state probabilities for current step:
$$a(L_j,t) = b_{jc_t} \sum_{i=0}^{states} (a(L_i,t-1) * a_{ij})$$
5. Compute hidden state probabilities estimation for next step:
$$a(L_j,t+1) = \sum_{i=0}^{states} (a(L_i,t) * a_{ij})$$
6. Compute priority $priority_{HMM}(p) = a(L_0,t+1)$.
7. Compute $similarity(p,c_r)$ between page content (or page content and anchor text) and the centroid c_r of relevant pages using VSM.
8. Assign priority to page:
$$priority_{learningHMM}(p,c_r) = \frac{similarity(p,c_r) + priority_{HMM}(p)}{2}$$

Fig. 2: Proposed HMM crawler page priority estimation

The operation of the focused crawler is illustrated in Fig. 3. Pages P_1 (in cluster C_1) and P_2 (in cluster C_2) are candidates for downloading. Both clusters C_1 and C_2 have identical probability of leading to cluster C_0 (target pages) in one step; however, both clusters may lead to target pages into two link steps. The crawler in [11] would assign higher probability to page P_1 (page P_2 may also reach C_0 in two link steps the same as non-target pages in C_3 and is assigned lower priority). The proposed crawler (Fig. 2) would rather prefer P_2 instead, because of its proximity with the centroid C_r of cluster C_0 (which is more correct intuitively).

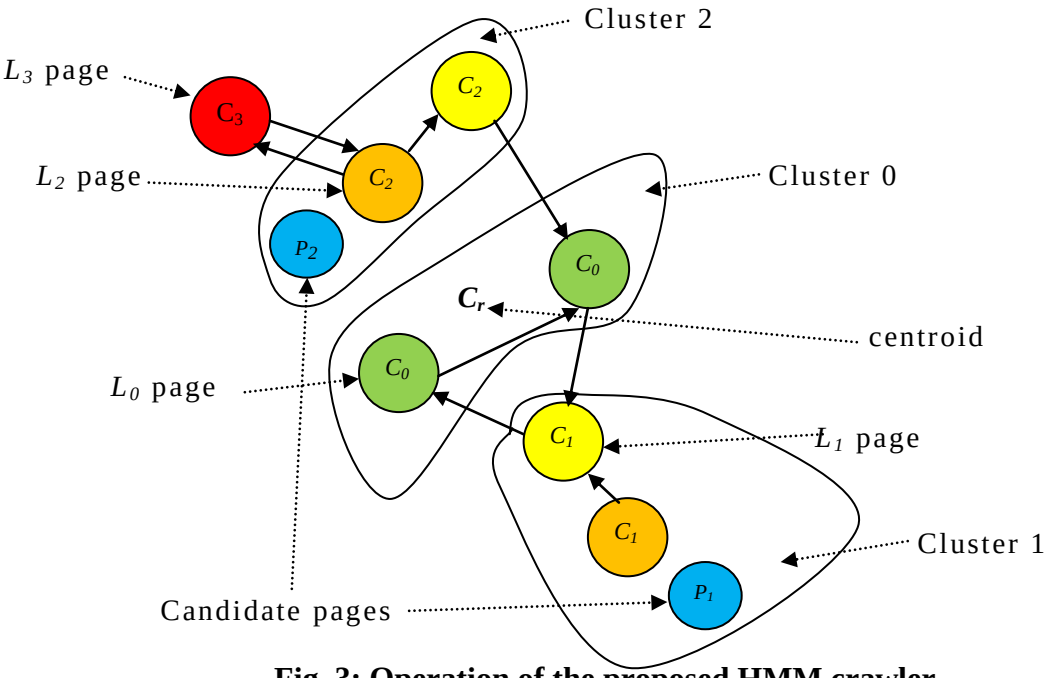

Fig. 3: Operation of the proposed HMM crawler.

4. Experimental Results

4.1 Experiment setup

All Crawlers are implemented in Java[8] . The downloaded pages must be in text/html format and their content size must not exceed 100KB. Restrictions are also imposed on connection timeout and downloading times for performance reasons. These restrictions apply to all implemented crawlers. The crawling process is repeated until the predefined number of pages (10,000 in this work) is retrieved. All crawlers presented in Section 3 are implemented and evaluated below based on results obtained on the following 10 topics: “linux”, “asthma”, “robotics”, “dengue fever”, “java programming”, “first aid”, “graph algorithms”, “optical nerve”, “champions league” and “Olympic games”.

Crawler performance is typically measured by the percentage of downloaded pages that are relevant to the topic (i.e. pages with similarity greater than a predefined threshold

[8] http://java.sun.com/

which is 0.75 in this work). This measure is referred as “harvest rate” [2] [18]. Harvest rate can be adjusted (by using higher threshold) to measure the ability of the crawler to download pages highly relevant to the topic. In the following, for clarity we present average numbers (over all topics) of relevant pages.

Each topic is issued as query to Google and the results are inspected by the user. Pages considered relevant by the user form the ground truth for the topic; the size of ground truth set is 50 pages for each topic. The results of a crawler on each topic are compared with the ground truth: for each page returned by the crawler, its document similarity (using VSM) with all pages in the ground truth set is computed. If the maximum of these similarity values is greater than a user defined threshold, the page is marked as a positive result (according to the method). The more the positive results of a crawler are, the more successful the crawler is (the higher the probability that the crawler retrieves results similar to the topic). The performance of a crawler is computed as the average number of positive results over all topics.

The following crawlers are compared:

1) Non Focused Crawlers:

   a) Breadth First Crawler

2) Classic Focused Crawlers:

   b) Best First Crawler with page content.

   c) Best First Crawler with anchor text.

   d) Best First Crawler with page content and anchor text.

3) Semantic Crawlers:

   e) Semantic Crawler based on Synonym Set similarity.

   f) Semantic Crawler based on Synonym, Hypernym & Hyponym similarity.

4) Learning Crawlers:

g) HMM Crawler as in [11].

h) HMM Crawler using page content similarity with relevant pages cluster centroid.

i) HMM with page content & anchor text similarity with relevant pages cluster centroid.

The crawlers were evaluated using the same 10 topics referred to above. Table 2 in appendix illustrates these topics along with their corresponding seed pages.

All crawlers where initialized using the same set of seed pages. Each crawler downloaded 10,000 pages on each topic[9]. Experiments were conducted between 20/8/2008 and 22/10/2008. The results for each method are represented by a plot showing the average number of relevant pages returned by the method as a function of the total number of downloaded pages. Each point in such a plot corresponds to the average over all topics. Classic and semantic crawlers accept the topic (or a description of the topic) as input. Instead, learning crawlers accept a set of target Web pages describing the topic.

4.2 Crawler Evaluation

We first present a comparative evaluation of the classic crawlers:

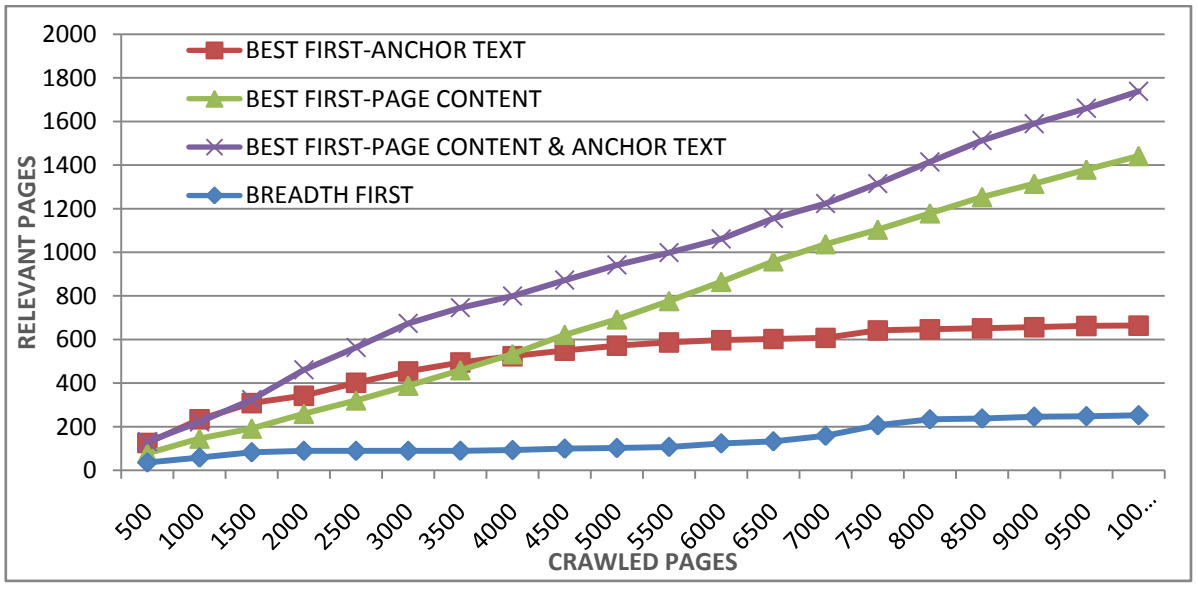

Fig. 4: Performance of classic crawlers (average values for topics of Table 2).

[9] We have made the datasets available on the internet at: http://www.intelligence.tuc.gr/~batsakis/crawler_dataset/

The comparison in Fig. 4 demonstrates the poor performance of Breadth First Crawler, as expected for a non focused crawler. Best First Crawler using anchor text outperforms the crawler using page content at the initial stages of crawling, indicating the value of anchor text for computing page to topic relevance. The crawler combining page and anchor text demonstrated superior performance. This result indicates that Web content relevance cannot be determined using page or anchor text alone. Instead, the combination of page content and anchor text forms a more reliable page content descriptor.

The second experiment compares the performance of semantic crawlers using the topics of Table 2 (as in the previous experiment) against the Best First-page content and anchor text crawler, which is the best classic crawler, as determined earlier.

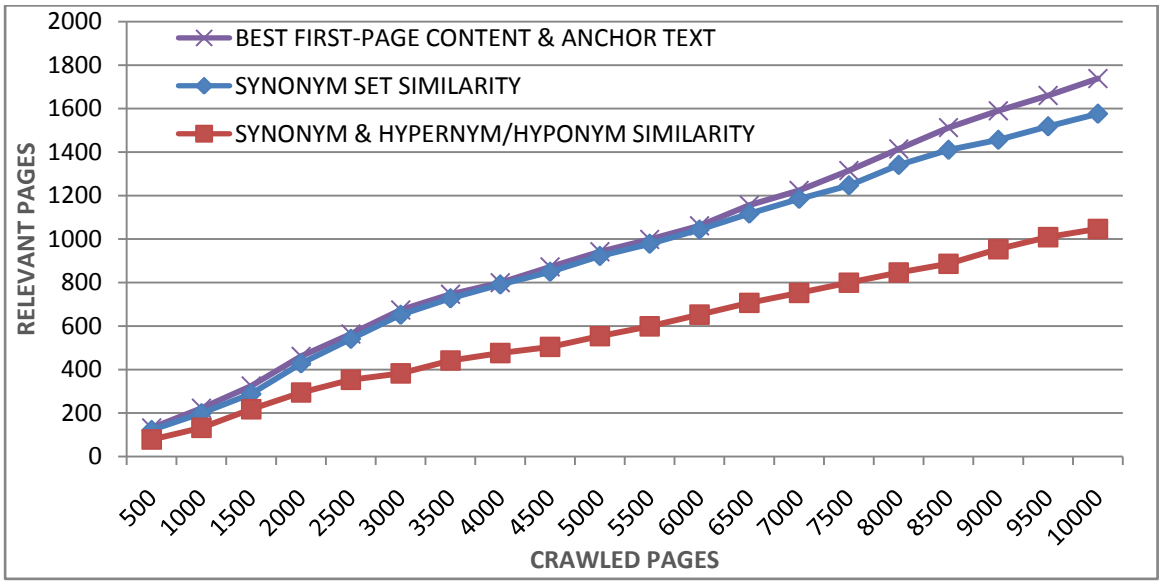

Fig. 5 : Performance of Semantic Crawlers (average values for topics of Table 2).

The Semantic crawler with synonym set similarity proposed in this work achieves almost identical performance with the Best-First crawler during the initial stages of the crawling and slightly inferior performance when more than 7000 pages are downloaded. Notice the inferior performance of the method using topic expansion by hyponyms and hypernyms in Wordnet (Fig. 5). Presumably, this should not be regarded as a failure of semantic crawlers but rather as a failure of WordNet to provide terms conceptually similar to the topic. WordNet is a

general taxonomy with many thousands of English terms; not all linked terms are in fact similar, implying that the results can be improved by using topic specific ontologies. Topic specific ontologies on several diverse topics were not available to us for these experiments.

Finally the performance of the HMM crawler [11] is compared with the performance of the proposed learning crawlers. The HMM crawler page content-centroid similarity prioritizes page links using the similarity of the page containing the links with the centroid of the relevant cluster in the training set. This is computed as document (term vector) similarity using VSM. The HMM Crawler with page content and anchor text-centroid similarity, combines anchor text and page content similarities with the centroid of the target cluster.

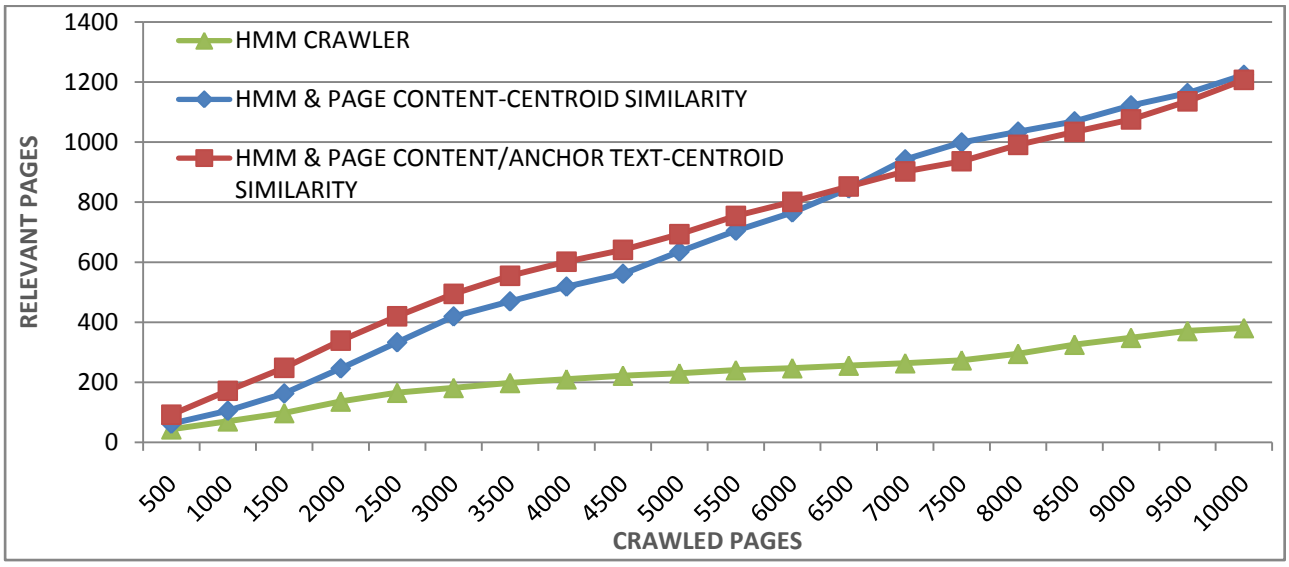

Fig. 6 : Performance of HMM based Crawlers (average values for topics of Table 2).

Both implementations of the proposed learning crawler outperform the state-of-the-art HMM crawler [11] as they allowed for pages associated with the same cluster sequence in the path leading to them (even for links within a page when using anchor text) to be assigned different priorities depending on their content.

4.3 Overall Comparison

As Fig. 4 and Fig. 6 indicate, the best average performance is achieved by the Best First crawler using page content and anchor text. However, average performance is not fully

representative of the strengths and weaknesses of the evaluated crawlers since the results obtained differ considerably among various topics. A closer examination of the results indicate that topics can be divided into two categories: the first category is composed of topics unambiguously defined by a list of keywords. These topics are "asthma", "java programming", "linux", "optical nerve" and "robotics" (topics numbered 1-5 in Table 2). All crawlers downloaded a large number of relevant pages and, at the same time, most of these pages are likely to point to additional relevant pages too. The results obtained in these topics are shown in Fig. 7:

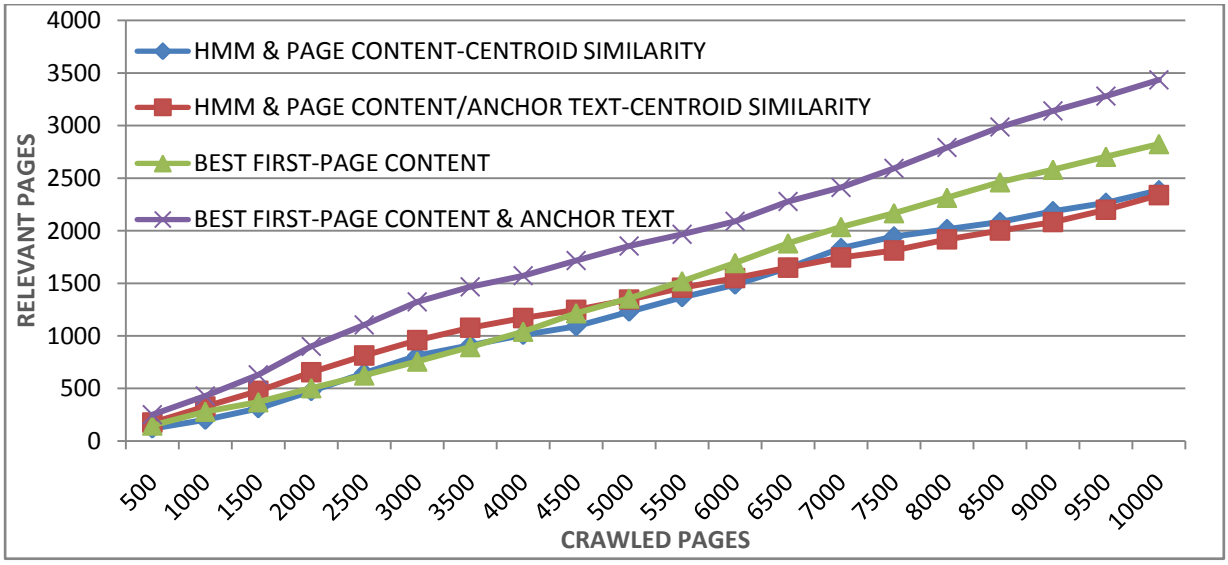

Fig. 7：Average performance of topics 1-5 of Table 2.

The Best-First crawler using page content and anchor text clearly achieves the best overall performance. The superior performance of classic over learning crawlers can be justified by the fact that topics 1-5 of Table 2 are easier to describe using keywords and relevant pages can easily be retrieved using content criteria alone.

The second category of topics consists of ambiguous (e.g. "champions league", "first aid") topics which are not clearly defined by keywords, topics containing terms that appear very frequently in non relevant pages (such as the term games in "Olympic games"), or very specialized topics (e.g. "dengue fever" and "graph algorithms"). Fig. 8 shows the performance

of classic and learning crawlers for ambiguous or specialized topics (i.e., topics numbered 6-10 in Table 2):

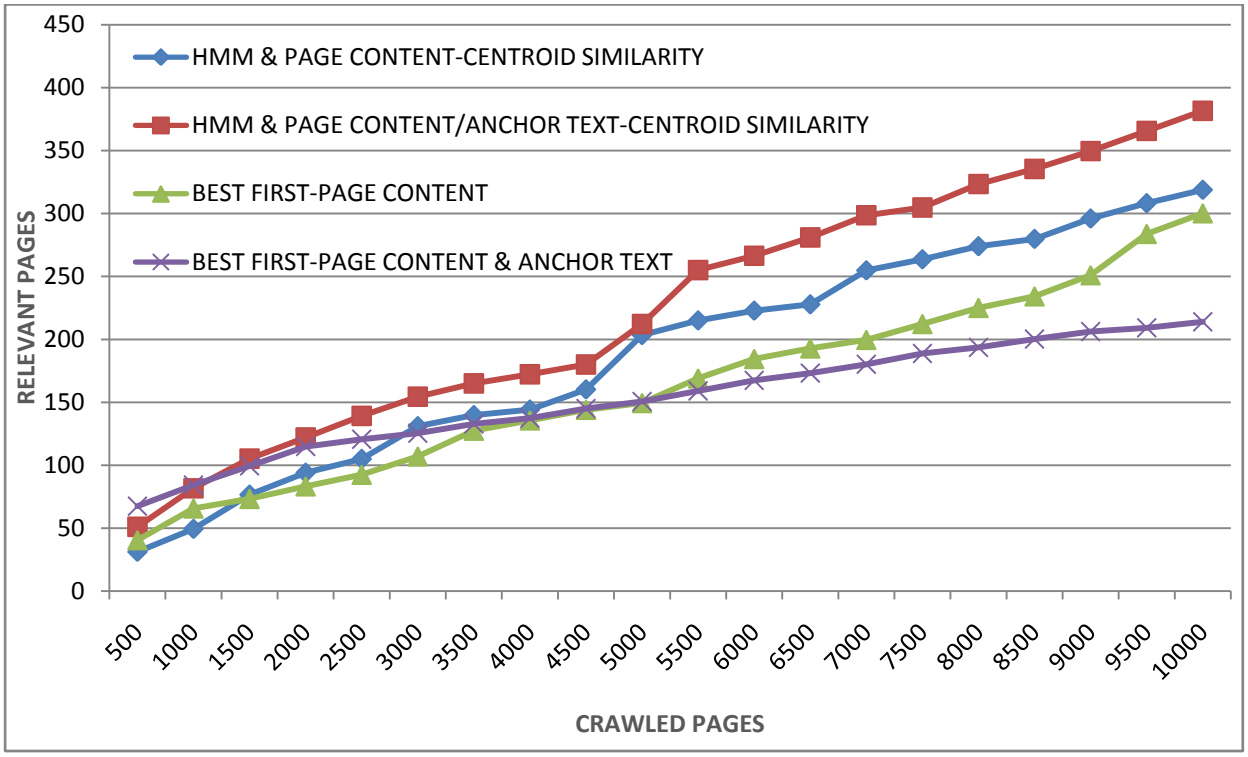

Fig. 8: Average performance of topics 6-10 of Table 2.

In this set of experiments, the proposed learning crawlers clearly outperform Best-First (classic) crawlers. Methods using combination of Web page content and anchor text are (in the case of learning crawlers) most effective than methods using Web page content alone. The effectiveness of learning crawlers in the case of vague or very specific topics can be justified as follows: user search preferences or topic descriptions can more effectively be modeled using a set of example pages, than a few keywords. At the same time, the number of relevant pages can be very small, and learning access paths can result in faster downloading of relevant pages than methods using mere content criteria as all classic crawlers do.

Table 1 illustrates average running times (in minutes) for crawling 10,000 pages along with average (over all topics for 10,000 pages) harvest rate for the methods considered in this work. Running times are only indicative of the time performance of each method and are presented only for comparison:

CRAWLER	Average running time (minutes)	Average harvest rate (percentage of relevant pages)
BREADTH FIRST	610,8	2,52%
BEST FIRST- PAGE CONTENT	741,2	14,42%
BEST FIRST –ANCHOR TEXT	813,9	6,64%
BEST FIRST – CONTENT & ANCHOR TEXT	969	17,38%
SYNONYM SET SIMILARITY	967,7	15,77%
SYNONYM & HYPERNYM/HYPONYM SIMILARITY	855,7	10,47%
HMM CRAWLER	1415,6	3,81%
HMM & PAGE CONTENT –CENTROID SIMILARITY	1571,8	12,24%
HMM & PAGE CONTENT /ANCHOR TEXT –CENTROID SIMILARITY	1644,4	12,07%

Table 1: Average crawling time and harvest rate.

Running time statistics are affected by several factors, such as network load, average downloaded file size, amount of parallelism (i.e., number of crawlers running in parallel), crawler dependent computations and platform capabilities. In general, simpler crawlers such as Breadth First are faster than more involved ones such as learning crawlers. When high performance systems are used (rather than low-end PCs as in this work) we expect that computational time will be minimized, and that network load will be the major factor affecting running time performance.

5. Conclusions and future work

In this work, several variants of focused crawlers were implemented and evaluated using several topics and based on well established performance criteria (harvest rate). These

include variants of classic, semantic and learning crawlers. Particular emphasis is given to learning crawlers based on the Hidden Markov Model (HMM) capable of learning not only the content of target pages (as classic focused crawlers do) but also paths leading to target pages.

Building upon previous work [11] a new HMM crawler is proposed. The strength of this crawler over [11] relies on its ability to distinguish between pages associated with the same cluster sequence in the path leading to them (in [11] such pages are all treated equally and are assigned the same priority value). This is achieved by comparing the content of each candidate page with the centroid of pages matching the query topic (positive examples) in the training set. All crawlers achieve best performance when a combination of page content and link anchor text is taken as the description of the candidate page.

The addition of semantic relations (suggested by semantic crawlers) didn't improve performance over classic crawlers, with better results obtained when semantic relations are restricted to synonym terms. However, it is plausible to expect that their performance can be improved by using application specific ontologies (related to the topic), instead of general purpose ontologies such as WordNet.

Learning crawlers take as input user selected pages (rather than a query or topic described by keywords as other focused crawlers do). In fact, learning crawlers perform a very difficult task: they attempt to learn web crawling patterns leading to relevant pages possibly through other not relevant pages thus increasing the probability of failure (since web structures cannot always be modeled by such link patterns). The experimental results indicate that the performance of HMM crawlers is promising overall (especially when the search topic is vague or very specific) and may lead to even more successful implementations of learning crawlers in the future. The present work can be regarded as a contribution towards that direction.

References

[1] **Cho, J.** "Crawling the Web: Discovery and Maintenance of a Large-Scale Web Data.". *Ph.D.thesis, Stanford University.* 2001.

[2] **Chakrabarti, S., van den Berg, M. and Dom, B.** "Focused Crawling: A New Approach for Topic Specific Resource Discovery". *In Proceedings of the 8th International World Wide Web Conference (WWW8).* 1999.

[3] **Menczer, F., Pant, G. and Srinivasan, P.** "Topical Web Crawlers: Evaluating Adaptive Algorithms". *ACM Transactions on Internet Technology (TOIT). 4(4):378–419, Nov. 2004.*

[4] **Salton, G., Wong, A. and Yang, C.S.** "A Vector Space Model for Automatic Indexing". *Communications of the ACM.18(11):613–620,* 1975.

[5] **Ehrig, M. and Maedche, A.** "Ontology-Focused Crawling of Web Documents". *Proc. of the Symposium on Applied Computing (SAC 2003).* March 9-12, 2003.

[6] **Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E.G.M., Milios, E.** "Semantic Similarity Methods in WordNet and their Application to Information Retrieval on the Web". *7th ACM International Workshop on Web Information and Data Management (WIDM 2005), Bremen Germany.* 2005.

[7] **Hliaoutakis, A., Varelas, G., Voutsakis, E., Petrakis, E.G.M., Milios, E.** "Information Retrieval by Semantic Similarity". *International Journal on Semantic Web and Information Systems (IJSWIS). Special Issue of Multimedia Semantics, 2006, Vol. 3 July/September, No.3, pp. 55-73*.

[8] **Pant, G. and Srinivasan, P.** "Learning to Crawl: Comparing Classification Schemes". *ACM Transactions on Information Systems (TOIS). 23(4), 430-462.* 2005.

[9] **Li, Jun, Furuse, K. and Yamaguchi, K.** "Focused Crawling by Exploiting Anchor Text Using Decision Tree". *Proceedings of the 14th International World Wide Web Conference. 2005, pp. 1190-1191*.

[10] **Diligenti, M., Coetzee, F., Lawrence, S., Giles, C., and Gori., M.** "Focused Crawling Using Context Graphs.". *Proc. 26th International Conference on Very Large Databases (VLDB 2000).* 2000, pp. 527–534.

[11] **Liu, H., Janssen, J. and Milios, E.** "Using HMM to Learn User Browsing Patterns for Focused Web Crawling". *Data & Knowledge Engineering. 59(2), 270–29.* 2006.

[12] **Chen, Y.** "A Novel Hybrid Focused Crawling Algorithm to Build Domain-Specific Collections". *PhD thesis, Virginia Polytechnic Institute and State University.* 2007.

[13] **Hersovici, M., Jacovi, M., Maarek, Y.S., Pelleg, D., Shtalhaim, M., and Ur, S.** "The Shark-Search Algorithm - An Application:Tailored Web Site Mapping". *Computer Networks and ISDN Systems.* 1998, *Vol. 30, No 1-7, pp. 317-26.*

[14] **Brin, S. and Page, L.** "The Anatomy of a Large-Scale Hypertextual Web Search Engine". *In Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, Apr. 14 –18. pp. 107–117.* 1998.

[15] **Kraft, R. and Stata, R.** "Finding buying guides with a web carnivore". *First Latin American Web Congress (LA-WEB'03).* 2003, pages 84–92.

[16] **Pant, G., Tsjoutsiouliklis, K., Johnson, J., and Giles, C. L.** "Panorama: Extending digital libraries with topical crawlers". *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries.* 2004b, p. 142–150.

[17] **De Bra, P., Houben, G., Kornatzky, Y., and Post, R.** "Information Retrieval in Distributed Hypertexts". *Proceedings of RIAO'94, Intelligent Multimedia, Information Retrieval Systems and Management, pages 481–491,New York,* 1994.

[18] **Aggarwal, C., Al-Garawi, F. and Yu, P.** "Intelligent Crawling on the World Wide Web with Arbitrary Predicates.". *Proc. 10th Intl. World Wide Web Conference.* 2001, pp. 96–105.

[19] **Corley, C. and Mihalcea, R.** "Measuring the Semantic Similarity of Texts.",. *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, pages 13–18, Ann Arbor,MI,* June 2005.

[20] **Badia, A., Muezzinoglu, T. and Nasraoui, O.** "Focused crawling: experiences in a real world project". *Proc. of the 15th International Conference on World Wide Web, Edinburgh,*2006, p. 1043-1044.

[21] **Xu, Qingyang and Zuo, Wanli.** "First-order Focused Crawling". Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada, Pages: 1159 – 1160, *2007.*

[22] **Pant, G. and Srinivasan, P.** "Link contexts in classifier-guided topical crawlers". *IEEE Transactions on Knowledge and Data Engineering.vol.* 18 (01), *p*. 107-122, 2006.

[23] **Bergmark, D., Lagoze, C. and Sbityakov, A.** "Focused crawls,tunneling, and digital libraries". *6th European Conference on Digital Libraries, Rome,* 2002.

[24] **Chakrabarti, S., Punera, K. and Subramanyam, M.** "Accelerated Focused Crawling through Online Relevance Feedback". *Proceedings of the eleventh international conference on World Wide Web(WWW2002).* 2002, pp. 148-159.

[25] **Bergmark, D.** "Collection synthesis". *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries, Portland OR.* 2002.

[26] **Porter, M.F.** "An Algorithm for Suffix Stripping". *Program.* 1980, Vol. 14(3), pp. 130- 137.

[27] **Steinbach, M., Karypis, G. and Kumar, V.** "A comparison of document clustering techniques". *6th ACM SIGKDD, World Text Mining Conference, Boston, MA* . 2000.

[28] **Rabiner, L. and Juang, B.** "An Introduction to Hidden Markov Models". *ASSP Magazine, IEEE. Vol. 3, Issue: 1, Part 1,p. 4- 16,*1986.

[29] **Cover, T. and Hart, P.** "Nearest Neighbor Pattern Classification". *Information Theory, IEEE Transactions on. Vol.13, Issue: 1,p. 21- 27,* 1967.

[30] **Liu, H., Milios, E. and Janssen, J.** "Probabilistic models for focused web crawling". *In Proceedings of 6th ACM International Workshop on Web Information and Data Management (WIDM 2004), pages 16–22.* 2004.

[31] **Bao, S., Li, R., Yu, Y. and Cao, Y.** "Competitor Mining with the Web Knowledge". *IEEE Transactions on Data Engineering*, Volume: 20, Issue: 10, page(s): 1297-1310, Oct. 2008.

[32] **McCown, F. and Nelson, M.** "Agreeing to Disagree: Search Engines and their Public Interfaces". *ACM IEEE Joint Conference on Digital Libraries (JCDL 2007)*. Vancouver, British Columbia, Canada. p. 309-318. June 17-23, 2007.

[33] **Pivk, A., Cimiano, P., Sure, Y., Gams, M., Rajkovic, V. and Studer, R.** "Transforming arbitrary tables into logical form with TARTAR". *Data & Knowledge Engineering*, Volume 60, Issue 3, Pages 567-595, March 2007.

[34] **Chang, C., Kayed, M., Girgis, MR. and Shaalan, KF.** "A Survey of Web Information Extraction Systems". *IEEE Transactions on Knowledge and Data Engineering*, TKDE-0475-1104.R3, 2006.

Appendix

Topic	Seed (starting) pages
1.Linux	http://dir.yahoo.com/Computers_and_Internet/Software/Operating_Systems/UNIX/Linux/ http://www.ask.com/web?q=linux&qsrc=0&o=0&l=dir http://www.dmoz.org/Computers/Software/Operating_Systems/Linux/
2.Asthma	http://dir.yahoo.com/Health/Diseases_and_Conditions/Asthma/ http://www.ask.com/web?q=asthma http://www.dmoz.org/Health/Conditions_and_Diseases/Respiratory_Disorders/Asthma/
3.Robotics	http://dir.yahoo.com/Science/Engineering/Mechanical_Engineering/Robotics/ http://www.ask.com/web?q=robotics http://www.dmoz.org/Computers/Robotics/
4.Optical nerve	http://www.dmoz.org/Health/Conditions_and_Diseases/Eye_Disorders/Optic_Nerve/ http://www.ask.com/web?q=optical+nerve http://www.google.com/search?hl=en&q=optical+nerve
5.Java programming	http://dir.yahoo.com/Computers_and_Internet/Programming_and_Development/Languages/Java/ http://www.ask.com/web?q=java%20programming http://www.dmoz.org/Computers/Programming/Languages/Java/
6.First Aid	http://dir.yahoo.com/Health/First_Aid/ http://www.ask.com/web?q=first+aid http://www.dmoz.org/Health/Public_Health_and_Safety/First_Aid/
7.Graph Algorithms	http://www.ask.com/web?q=graph+algorithms http://www.google.com/search?hl=en&q=graph+algorithms http://en.wikipedia.org/wiki/Category:Graph_algorithms
8.Dengue Fever	http://dir.yahoo.com/Health/Diseases_and_Conditions/Dengue_Fever/ http://www.ask.com/web?q=dengue+fever http://www.dmoz.org/Health/Conditions_and_Diseases/Infectious_Diseases/Viral/Hemorrhagic_Fevers/Dengue_Fever/
9.Champions league	http://dir.yahoo.com/Recreation/Sports/Soccer/Tournaments/Club/UEFA_Champions_League/ http://www.google.com/search?hl=en&q=champions+league http://www.ask.com/web?q=champions+league
10.Olympic games	http://dir.yahoo.com/Recreation/Sports/Events/International_Games/Olympic_Games/ http://www.dmoz.org/Sports/Events/Olympics/ http://www.ask.com/web?q=olympic+games

Table 2 : Topics and seed pages for crawling initialization.