Technical University of Crete

Department of Electronic and Computer Engineering

DESIGN AND EVALUATION OF TOPIC DRIVEN FOCUSED CRAWLERS FOR THE WORLD WIDE WEB

By

BATSAKIS SOTIRIOS

A Thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Computer Engineering

Chania, November 2007

Design and evaluation of topic driven focused crawlers for the World Wide Web

Batsakis Sotirios

Abstract

Focused crawlers are programs designed to browse the Web and download pages on a specific topic. They are used for answering user queries or for building digital libraries on a topic specified by the user. They are distinguished into classic, semantic and learning focused crawlers. Classic focused crawlers estimate the relevance of Web pages with the topic by computing the similarity of Web pages with a user provided list of keywords that describe the topic of interest. Semantic Crawlers are a variation of classic focused crawlers that use conceptual relations between terms (e.g. retrieved from an ontology) for estimating the relevance of the Web page with the topic. Learning crawlers employ a training process that guide the crawler towards pages related to the topic.

This work address issues related to the design and implementation of classic, semantic and learning focused crawlers. Several variants of classic focused crawlers relying up on web page content and link anchor text for estimating the relevance of web pages to a given topic are examined and implemented. A novelty of this work is the introduction of a new category of semantic crawlers making use of WordNet as the underlying ontology for obtaining terms conceptually related (but not necessarily lexicographically similar) with the topic. Learning crawlers based on Hidden Markov Model (HMM) for learning not

only the content of relevant pages but also paths leading to relevant pages following a certain number of routing hops are examined as well. An additional contribution of this work is the introduction of a new category of hybrid crawlers combining the strength of both classic and learning focused crawlers.

The crawlers referred to above are all implemented and a comparative analysis of their performance is presented. All crawlers achieve their maximum performance when a combination of web page and anchor text is used for assigning download priorities to web pages. Semantic similarity methods combined with a general purpose ontology source such as WordNet don't actually improve performance, except the implementation that restricts semantic similarity to synonym terms. Hybrid crawlers improved the performance of state of the art HMM crawlers yielding very promising results.

Contents

Chapter 1. Introduction

The World Wide Web is a huge information source with billions of web pages on every conceivable subject. General purpose search engines such as Google [5], Yahoo [7], MSN [8] and Ask [9] have appeared in order to assist users in finding information on the Web. These search engines are very complicated and sizable systems [1, 2], but they don't achieve a full coverage of the Web. Google achieves up to 76% and Yahoo up to 69% coverage, while other search engines index an even smaller percentage of the entire Web [3]. Information searches on the Web issued through Web search engines are not propagated over the Web in real time. Instead they index, analyze and categorize Web information accumulated locally in data repositories and this information is then used for answering user queries. The general purpose search engine approach effectively addresses the need of the end user to find specific information in real time.

Crawlers (also known as Robots or Spiders [20]) are tools for assembling locally information from the Web. Focused crawlers in particular, have been introduced for satisfying the need of individuals (e.g. domain experts) or organizations to create and maintain locally digital libraries on a subject or for answering complicated queries (for which a web search engine would yield limited or no satisfactory results). Typical requirements of such application users are the need for high quality up-to-date results, while minimizing the amount of resources dedicated to the search task. Focused crawlers download as many pages relevant to the subject as they can, while keeping the amount of irrelevant data downloaded to a minimum [30]. Besides the creation of specialized digital libraries, applications of

focused crawlers also include guiding intelligent agents on the Web for locating specialized information (e.g. flight schedules and ticket prices for a voyage planning agent). As the importance and the size of the Web grows so does the importance of Focused Crawlers.

1.1 Background

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

The Crawlers used by general purpose search engines retrieve Web pages massively regardless their topic. Methods for implementing such Crawlers include:

a) **Breadth First Crawlers:** The outgoing links from the given set of pages are extracted and inserted in a First In First Out (FIFO) queue, and their corresponding web pages are downloaded. The process continues similarly with the new pages.

b) **Page importance Crawlers:** They assign higher visit priority to web pages (i.e. to their corresponding URLs) linked to from more important pages. Page importance estimation criteria for assigning priorities to extracted URLs include *Backlink* count (i.e. number of web pages containing links to a given page) [22] and *PageRank* (the importance estimation method used in the Google search engine)[6].

Although simple, Breadth First Crawlers achieve good performance (measured as the average quality of downloaded pages using PageRank criterion)[19], and are effective for implementing non-focused Crawlers. The major disadvantage of Breadth First Crawlers (and of the other non topic driven Crawlers) is that they use only the link structure of the web and not web page content in assigning visit priorities to URLs; consequently they fail to focus on pages on a topic. Because pages on a specific topic are a minor fraction of the overall Web, crawling on that topic using non focused crawlers will result into downloading a large number of irrelevant pages, thus quickly exhausting the available resources. Therefore building a specialized digital library calls for focused crawlers.

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 relevant to the topic. They are distinguished into the following categories:

a) **Classic Focused Crawlers** [26] take as input a user query that describes the topic and a set of starting URLs (seeds). The crawling starts from the user provided seed URLs. The crawlers assign a priority value to visited pages according to their relevance to the topic. The web pages are ordered by relevance and the crawlers proceed by visiting the most relevant web pages first. The most common criterion for relevance estimation between a retrieved page and a user query is defined as the similarity between the text of the visited page with the query (topic). Typically this is computed using a text similarity model such as the Boolean or the Vector Space Model [12]. Focused

crawlers using Vector Space Model for relevance estimation (Best First Crawlers [25]) are the most effective classic focused crawling method so far [26]. Existing work on classic focused crawlers is presented in section 2.3. Our proposed variants and implementations of classic focused crawlers are discussed in section 3.2.

b) **Semantic Crawlers** are a variation of classic focused crawlers. Page visit priority is assigned to pages using their content and by applying semantic criteria for computing page-to-topic relevance. A page and the query can be relevant if they share conceptually similar (but not necessarily lexically similar) terms. Conceptual relations between terms are defined using an underlying topic specific or general purpose ontology. Thus, semantic crawlers differ with classic focused crawlers in the way content relevance is computed. To the best of our knowledge semantic crawlers haven't been compared with state-of-the-art classic focused crawlers such as those referred to above, nor have they been combined with modern semantic similarity methods (as those presented in [11]) so as to achieve their full potential. The present work addresses all these issues (section 3.3).

c) **Learning Crawlers** [33] 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 by machine learning or other processes) so that the crawler can distinguish between relevant and non relevant pages. Building upon this idea, a number

of approaches for learning relevant to the topic Web pages have appeared in the literature and include:

1. Approaches based on machine learning: The crawler is supplied with a training set consisting of relevant and non relevant Web pages which is used to train the learning Crawler [33,34]. During crawling higher visit priority is assigned to web pages classified as relevant to the topic.
2. Approaches that take not only the page content and the corresponding classification of web pages as relevant or non relevant to the topic into account, but also the link structure of the Web and the probability that a given page (which can be non relevant to the topic) will lead to a relevant page within the minimum number of steps (hops). Methods based in *Context Graphs* [31] and *Hidden Markov Models* (HMM) [16] are examples of this category of methods. Section 2.5 contain a detailed description of these methods and Section 3.4 the enhancements proposed in this work.
3. Hybrid methods that combine learning crawlers with ideas of classic focused crawlers [35]. Our work focuses on hybrid crawlers and proposes an approach that combines the strengths of classic focused crawlers (variations of Best First Crawlers) with Hidden Markov Models for learning not only how to distinguish between relevant and non relevant Web pages based on content, but also on learning how to guide the search for such relevant Web pages through a sequence of routing hops between Web pages (sometimes through non relevant pages). This method is described in section 3.4 and the experimental results obtained

(Section 4.6) indicate that it is a very effective crawling method.

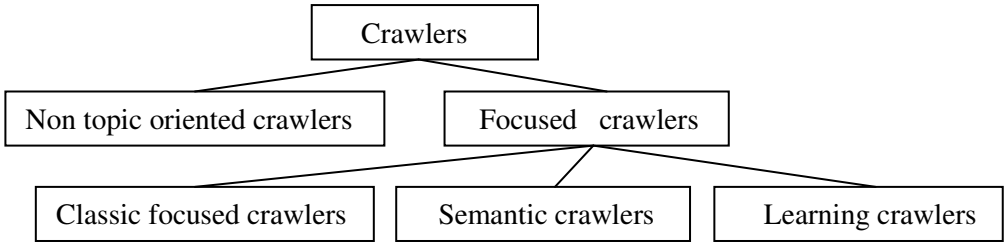


Fig. 1: Crawler Classification

1.2 Present work

This work deals with the design and evaluation of focused crawlers. State of the art approaches for building topic driven focused crawlers are considered including classic, semantic and learning crawlers. Several variants of these approaches are also proposed and evaluated. The emphasis of this work is on hybrid crawlers combining text and link information for reaching faster more promising pages on the topic of interest.

The first crawler implemented is the *Breadth First Crawler*. This is a classic non topic-oriented crawler which is used as a reference in all comparisons with focused crawlers. Several variants of the *Best First Crawler* [25] are also implemented and evaluated. Best First Crawler works by estimating the relevance of the retrieved page with the user query (both represented using term vectors) using Vector Space Model (VSM) [12]; then it visits the links extracted from the most relevant page. A URL can be represented either by the term vector of the Web page it was extracted from, or by the term vector of its corresponding anchor text (the text that appears on the link pointing to that URL). All solutions (using page content, anchor text or their

combination) are implemented and evaluated. These methods are described in section 3.2.

The second category of methods includes Semantic Crawlers that estimate the conceptual (semantic) relevance of a Web page with the query. The method by *Ehrig* et.al [13] combines focused crawlers and semantic relations from an ontology (in [13] topic specific ontologies were used), for assigning visit priorities to pages. In our implementation of semantic crawlers, term vectors are enhanced with synonyms and semantically similar terms from *WordNet* [4] (thus making our implementation the first general purpose semantic crawler implementation). Topic relevance can then be computed by VSM [12], the Semantic Similarity Retrieval Model (SSRM) [14] or by Mihalcea et.al. [15].

Our proposed approach to Learning Crawlers is influenced by work on *HMM Crawlers* [16, 18] for learning paths leading to relevant pages in addition to the content of the desired web pages. The user of an HMM Crawler provides a training set of pages (both relevant and non relevant to the topic of interest). These pages are clustered according to their content. Transition probabilities between the resulting clusters representing relevant or non relevant pages (leading to relevant ones) are computed and are used to estimate (given the cluster a Web page is assigned), the probability that it will lead to relevant pages. The higher this probability is the higher the visit priority given to the page's extracted links will be. K-Means [47] and X-Means [17] can be applied for the clustering of Web pages. K-means clustering is an algorithm to classify or to group objects based on attributes/features into K groups (K is positive integer predefined number). The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. X-Means is an extension of

K-means with dynamic estimation of the number of clusters dependent on the data. In this work focused crawlers based on both clustering approaches are implemented and evaluated.

Based on the HMM Crawler, *Hybrid Crawlers* that combine classic focused crawlers for assigning priorities to URLs based on topic relevance, and learning crawlers for learning access paths for reaching relevant pages (possibly through non relevant ones) are proposed. Two hybrid crawlers combining HMM with page or both page and anchor texts are implemented and evaluated. Our proposed approach to hybrid crawlers is presented in section 3.4.

The crawlers referred to above (and their variations) are all implemented and their performance is compared based on results obtained from the web using several different topics and seed (starting) pages. Section 4 presents a comparative study of the performance of all crawler variants by category along with a critical analysis of their performance.

1.3 Contribution of the current thesis

The contributions of this work are summarized below:

a) This thesis presents a critical evaluation of state of the art approaches to Web Crawling, including Classic, Semantic and Learning Focused Crawlers. To our knowledge a similar evaluation hasn't appeared in the literature before.
b) Proposes several variants to existing crawling methodologies based on recent semantic relevance estimation methods and compare their performance with classic focused crawling methods.
c) Proposes a novel hybrid approach to learning crawling combining classic focused crawlers for assigning priorities to URLs with ideas from learning crawlers

for learning paths for reaching web pages relevant to the topic.

1.4 Thesis outline

The work in this thesis is organized as follows: Related work on focused crawling is presented in Section 2. It is organized in six subsections; the first is the introduction, the second subsection (2.2) presents non topic driven crawlers, the third subsection (2.3) classic implementations of focused crawlers, the fourth subsection (2.4) the preliminary related work on semantic crawlers , the fifth subsection (2.5) presents previous work on Learning Crawlers and the sixth is a summarization of the above.

Issues related to the design and implementation of Web crawlers is presented in section 3. Subsection 3.1 is an introduction to the topic, subsection 3.2 provides a detailed description of classic crawlers implemented in this work and subsection 3.3 deals with issues related to the design of semantic crawlers. In subsection 3.4 particular emphasis is given to learning crawlers and to the subsequent design of hybrid crawlers.

Section 4 provides a description of the experimental results. Subsection 4.1 present the purpose of the experiments, in the second part (subsection 4.2) the performance measures used to evaluate the crawlers are described. The experimental setup is discussed in subsection 4.3. Experimental results on Classic Crawlers are presented in subsection 4.4 followed by results obtained by semantic and learning crawlers in subsections 4.5 and 4.6 respectively. Subsection 4.7 presents a critical analysis of the performance of various crawlers methods considered in this work. Finally conclusion and issues for further research are discussed in Section 5.

Chapter 2. Related Work

2.1 Introduction

Related work on crawlers includes contributions regarding both classic (non-topic oriented) and focused (topic-oriented) crawlers. Existing work on the design and implementation of non focused crawlers and of focused (classic, semantic and learning) crawlers proposed in the literature is presented in this chapter.

Classic non focused Crawlers (e.g. crawlers used by web search engines for assembling web pages to local repositories) download Web pages massively regardless of content in order to create vast page repositories. Focused crawlers on the other hand are more selective, downloading only pages related to a known (user provided) topic. Issues related to the design and implementation of classic as well as of focused crawlers are discussed in the following and include:

a) **Search strategy**: The crawler can browse the web in a breadth first order or select links to follow using importance estimation criteria. Focused crawlers assign visiting priorities to pages according to the relevance of the page with a topic specified by a user.
b) **Refreshing policy**: Due to the dynamic nature of the Web, pages must be revisited in order to keep page repositories up-to-date. The optimal page refresh policy that achieves keeping page repositories up-to-date without unnecessary downloading of non out-dated pages is a very important issue in crawler design [21]. Also, satisfying the conflicting demands for high downloading rate without putting excessive load to the

visited Web sites is a major concern when designing a Crawler for a search engine.

c) **Synchronization**: Crawlers used by commercial search engines use multiple parallel processes that massively retrieve Web pages, regardless of their topic. These processes must be synchronized in order to avoid duplicated page downloading [20].

2.2 Non Focused Crawlers

Typically non focused crawlers are used by general purpose search engines for assembling locally Web information. Methods for implementing such Crawlers include:

a) **Breadth First Crawler:** After downloading the initial pages (called seed pages) the outgoing links extracted from these pages are put in a FIFO queue. The links extracted first point to pages that are given the highest priority for downloading and further crawling. Breadth first crawling is one of the most commonly used crawling approaches for assembling locally Web content for use by Web search engines. GoogleBot [5], Slurp [7], MSNBot [8] and Teoma [9] are examples of crawler implementations used by commercial search engines. Page refresh policy, synchronization, and optimal downloading rate are important issues here [20, 21]. Technical issues such as the supported file formats, file size limitations and the visiting policy are also of great importance. Breadth first crawlers are capable of crawling a large part of the Web [3]. Then the downloaded pages are analyzed (e.g. by content, type) indexed and subsequently stored in data repositories composed of thousands of computers and Terabytes of data [1, 2]. This approach requires huge

resources which are available only to large companies or organizations such as Google or Yahoo.

Crawlers such as Mercator [45] and Larbin [10] are examples of Breadth First Crawlers which are freely available to programmers for testing and system development. When limited resources are available they can crawl a small part of the indexed web and retrieve web content for further processing. Breadth first crawlers yield high quality pages [19] but aren't topic oriented.

b) **Page importance Crawlers:** They assign higher visit priority to URLs retrieved from more important pages. Typically, page importance for assigning priorities to extracted URLs is computed by *Backlink* count (where higher priority is given to pages pointed to by many other Web pages) and *PageRank* [6]. Other criteria such as the position of the page within the Web site hierarchy (e.g. low depth, as indicated by fewer -or none - slashes into the page URL, lead to higher priority), or the number of outgoing links of that page (*Outlink* count) can be used as well. Cho et.al [22] provides a survey on this type of Crawlers. Page importance criteria have been shown to improve the quality of downloaded pages [22].

2.3 Classic Focused Crawlers

Crawlers used by search engines (such as those referred to in section 2.2) are designed to maximize the total number and probably the quality of downloaded web pages. Topic oriented or Focused Crawlers take as input a user query (Classic Focused crawlers), or example pages provided by

the user as a training set (Learning Crawlers) and focus the crawling process on pages relevant to the topic. Focused crawlers keep the overall number of downloaded Web pages to a minimum while maximizing the percentage of relevant pages.

The performance of a focused crawler depends on the selection of good starting pages (seed pages). Good seed pages can be either web pages relevant to query topic or pages from which relevant pages can be accessed through a small number or routing hops. For example, if the topic is on scientific publications, a good seed page can be the publications page of an author, lab or department or alternatively the web page of the author, lab or department respectively (although the last may contain no publications at all, it is known to lead to pages containing publications). Seed pages should also be important as well (where importance is defined using link analysis methods such as HITS [46] and Page Rank [6]). The rationale behind this requirement is that important Web pages (when used as starting pages –seeds – for crawling) may guide crawling process to other important Web pages fastest, thus improving the quality of the results. The seed pages are often selected by submitting the query that describes the topic of interest to a search engine and by using the top search engine results.

Early approaches on Focused Crawling include among others the Fish-Search algorithm [23]. The basic idea of the algorithm is that when several pages are candidates for link browsing and downloading, priority is given to pages relevant to the topic (a page is labeled as relevant if it contains the query text). Every candidate page is assigned a Boolean value derived by a simple lexicographic rule (and it is downloaded by a separate application thread). Threads corresponding to relevant pages create new threads for their

outgoing links, while threads corresponding to irrelevant pages are stopped. This work examined the separate use of the anchor text in assigning priorities to URLs.

The main disadvantage of the Fish-Search algorithm is that priorities take Boolean values; therefore all relevant pages are assigned the same priority. The Shark-Search algorithm [24] is a direct successor of Fish-Search, where VSM [12] is used for assigning non Boolean priority values to candidate pages. This improved the results of crawling [24]. The Vector Space Model became the basis of classic focused crawlers ever since.

According to VSM, documents are represented as term vectors and the weight W_{ij} of a term j in document i is computed as:

$$W_{ij} = tf_{ij} * idf_j$$
$$tf_{ij} = \frac{f_{ij}}{maxf_i}, \quad idf_j = log\frac{N}{N_j} \qquad (1)$$

Where tf_{ij} is the term frequency of term j in document i, idf_j is the inverse document frequency of term j, f_{ij} is the frequency of appearance of term j into document i, $maxf_i$ is the maximum frequency of all terms into document i, N is the total number of documents and N_j is the number of documents containing term j.

Recent approaches to focused crawling include InfoSpiders and Best-First Crawler [25]. InfoSpiders use Neural Networks, while Best First Crawlers use text similarity by VSM for assigning priority values to candidate pages.

Given a query and a Web page, the priority of the Web page is computed by Best First Crawlers as the cosine

similarity between their document vectors where W_{qj}, W_{ij} are term weights of the query and the web page respectively:

$$similarity(link_i, query_q) = \frac{\sum_{j=0}^{T} w_{ij} * w_{qj}}{\sqrt{\sum_{j=0}^{j=T} w_{ij}{}^2} \sqrt{\sum_{j=0}^{j=T} w_{qj}{}^2}} \quad (2)$$

Where T is the total number of terms into query and page content.

In fact, the Best First Crawler is a simplified version of the Shark-Search crawler: It doesn't combine link anchor text and previous visited pages scores into the page priority function, as Shark-Search does. Also, 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) in the case of focused crawling is problematic since this might require recalculation of all term vectors at every crawling step. In addition, idf values are highly inaccurate at the early stages of crawling because of the small number of retrieved documents. Best First Crawlers have been shown to outperform InfoSpiders, and Shark-Search and also other non-focused crawling approaches such as Breadth First, and PageRank [26]. Best first crawling is considered to be the most established approach to focused crawling due to its simplicity and efficiency. The N-Best First Crawler is a generalized version of Best First Crawler: at each step, instead of choosing one Web page for link extraction and downloading of pages pointed to by these links, N pages with highest priority are chosen [27].

Along the same lines, an approach referred to as "intelligent crawling" [28] suggests combining page content, URL string and statistics about relevant/irrelevant pages and sibling pages for assigning priorities to candidate URLs. These statistics are updated and combined during crawling

for guiding the selection of the next links to follow yielding a highly effective crawling algorithm that learns to crawl without direct user training.

2.4 Semantic Crawlers

Semantic Crawlers are implemented by combining an ontology with semantic similarity measures [14] for detecting topic relevance between retrieved Web pages and user queries. Semantic similarity plays an important role here: it can be used to detect topic relevance by associating terms in a query and the Web page using the ontology, and by assigning a degree of relevance to each such term association.

Ehrig et.al [13] proposes use of topic oriented ontologies for finding pages relevant on the topic of interest. Every term in a Web page is examined and contributes positively to assigning a priority score if it is a query term or if it is semantically related to the user query terms. The following variations for evaluating semantic relations of page terms with query terms were used:

a) If a term is directly connected (distance 1) to a query term, then it is considered relevant (distance is defined as the length of the shortest path connecting two terms represented as vertices into the ontology graph where edges represent relation of adjacent terms).
b) If a term is close to a query term (distance 2 or less) using only IS-A relations then it is relevant to the query term.
c) Every page term appearing into the ontology graph is assigned a relevance value depending on its distance with query terms. The greater the distance the lower the relevance value will be. Specifically, using a topic

specific underlying ontology the semantic similarity between terms is computed as:

$$sim_{df}(t_1, t_2) = df^{sp(t_1,t_2)} \quad (3)$$

Where df is a decreasing factor (0.5 in this work) and $sp(t_1, t_2)$ is the length of shortest path connecting terms t_1 and t_2 into the ontology graph (0 if the terms belong to the same synonym set). The longer the distance of the terms into the graph the smaller their similarity is. This method is a variation of the shortest path semantic similarity method.

The last approach has the best performance for computing the conceptual similarity between terms and was also used in our work for comparison with other semantic relation methods and state-of-art classic focused crawling approaches. Another state of the art term similarity method used in present work is the Li et.al method [42]:

The semantic similarity between two terms t_1 and t_2 is computed as a function of the length of the path connecting the terms in the underlying ontology graph and the depth of terms into the taxonomy:

$$sim_{li}(t_1, t_2) = e^{-aL}\frac{e^{bH} - e^{-bH}}{e^{bH} + e^{-bH}} \quad (4)$$

Where L is the shortest path length between t_1 and t_2, H is the depth of the most specific common concept of t_1, t_2 into the taxonomy and a, b are constants ($a = 0{,}2$ and $b = 0{,}6$ in our implementation).

According to results reported in [14] this method have been proven to be fast and accurate (achieving accuracy up to 82% compared to results obtained by humans).

General purpose taxonomies such as WordNet can also be applied for focused crawling. WordNet is an online lexical

reference system developed at Princeton University. WordNet attempts to model the lexical knowledge of a native speaker of English. WordNet can also be seen as ontology for natural language terms. It contains around 100,000 terms, organized into taxonomic hierarchies. Nouns, verbs, adjectives and adverbs are grouped into synonym sets (synsets). The synsets are also organized into senses (i.e. corresponding to different meanings of the same term or concept). The synsets (or concepts) are related to other synsets higher or lower in the hierarchy defined by different types of relationships. The most common relationships are the Hyponym/Hypernym (i.e., Is-A relationships), and the Meronym/Holonym (i.e., Part-of relationships). There are nine noun and several verb Is-A hierarchies (adjectives and adverbs are not organized into Is-A hierarchies). Figure 2 illustrates a fragment of the WordNet Is-A hierarchy.

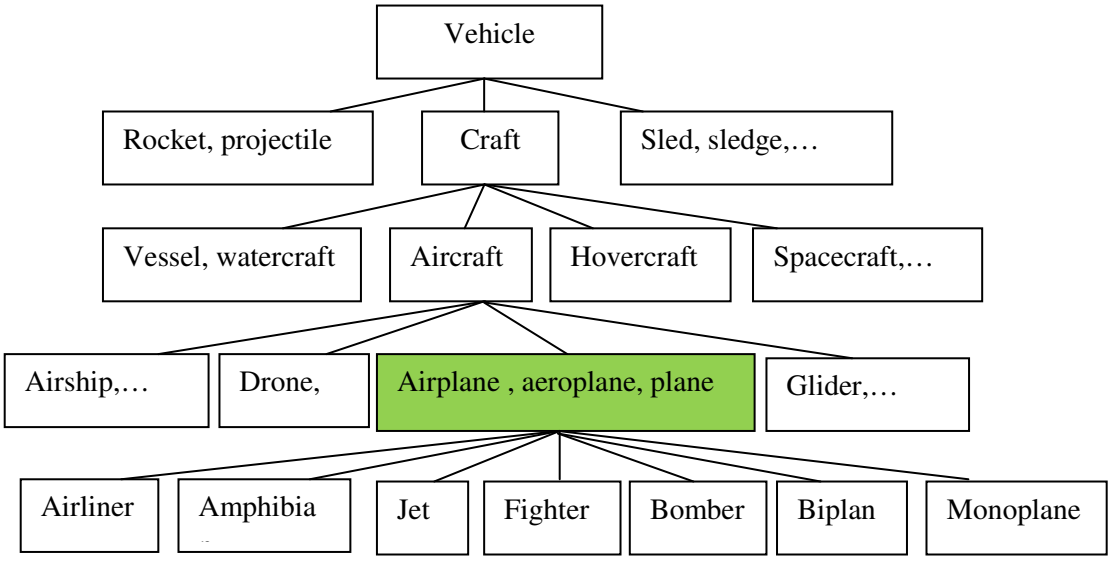


Fig. 2 WordNet Hypernym/hyponyms synsets relations example

To the best of our knowledge a comparative study between semantic and other focused crawling approaches hasn't been reported in the literature before. The implementations in [13] are compared only with a basic focused crawler (assigning each page a simple binary priority value depended on the presence of query terms) rather than with the widely used Best First Crawlers making use of VSM for estimating topic relevance [29]. The proposed work deals with exactly this issue and presents a comparative study between classic and several variants of semantic crawling approaches (including Ehrig et.al [13]).

2.5 Learning Crawlers

Early approaches to developing learning crawlers applied a learning classifier (that relied on web taxonomies such as Yahoo [7]) and used for distinguishing between relevant and non relevant pages [30]. Every page containing links candidate for downloading is classified as relevant or not relevant and assigned a priority value according to that classification (higher priority was assigned to relevant pages). This work is considered to be one of the first contributions in the field of Learning Crawlers. Resent approaches involving machine learning methods for focused crawling include decision trees [34], Neural Networks and Support Vector Machines [33].

Building upon similar ideas the crawler in [31] introduced the concept of Context Graphs: For every relevant page a search engine's backlink service is applied to retrieve its predecessor pages. Then, a classifier is build according to the distance of pages (Level) to the relevant pages set. Download priorities are estimated using this classifier: The closer a candidate page to a relevant one is, the greater the priority of that page will be.

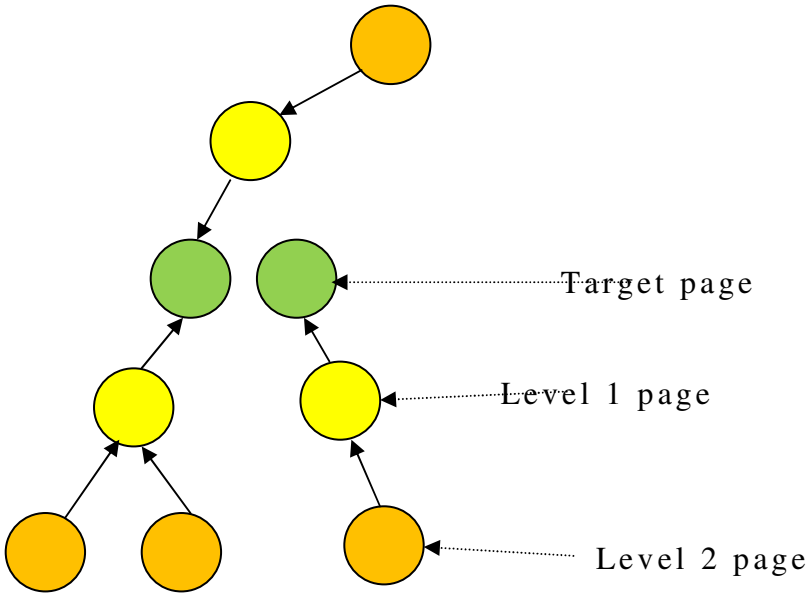


Fig. 3 Context graph: Pages are classified according to their distance (Level) from target pages.

An extension to the Context Graph method was the Hidden Markov Model (HMM) crawler [16]. The user browses the Web and indicates if a downloaded page is relevant to the topic or not. The visiting sequence is also recorded and is used for training the algorithm to identify paths leading to relevant pages. The downloaded pages are clustered and a Hidden Markov Model [44] is created: Every 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 relevant page (0 if the page is a target/relevant page). During crawling every page is assigned a value equal to the probability that given the cluster the page belongs to, crawling will lead to a target page, this probability is computed using the Hidden Markov Model.

Specifically All pages are represented by their term vectors according to VSM and they are clustered. Thus every page into the training set in characterized by the cluster it belongs to and by its distance (level) from a target page (Fig. 4).

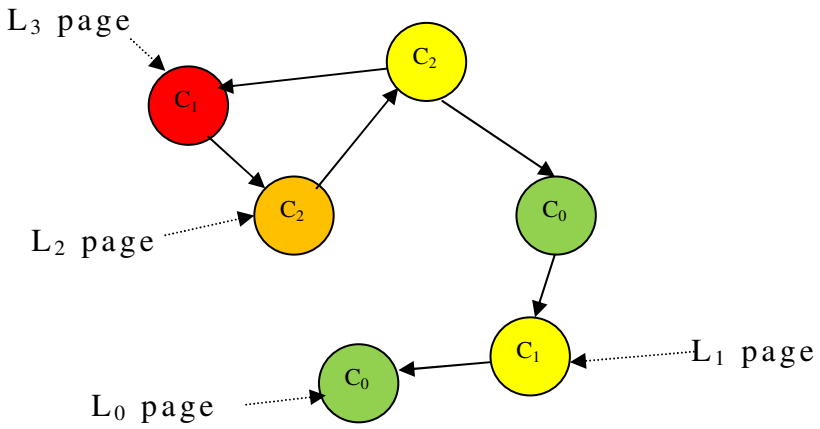


Fig. 4 Representation of the HMM training set using distance from target pages (Level) and clusters of pages in the training set.

In figure 4 green pages indicate target or level 0 pages, yellow pages are level 1 pages (1 link distance from target pages), orange pages are level 2 (2 links away from target pages), and red pages are 3 or more links away from target pages. Labels on pages represent the cluster the page belongs to (e.g. C_0, C_1 and C_2 labels corresponding to Cluster 0, Cluster 1 and Cluster 2 respectively). Notice that pages within the same Cluster can belong to different levels, and that pages in the same level can belong to a different cluster. Every page is characterized by its level or hidden state L_i where *i* is the level, and by the cluster C_j it belongs (or visible state). That set of pages with hidden and visible

states form a Hidden Markov Model [44]. The following summarizes the parameters and notation used by HMM crawler:

I. Initial probability matrix :

$\pi = \{ P(L_0), \dots, P(L_{states-1})\}$

Where $states$ denotes the number of hidden states and$P(L_i)$ represents the probability of being at hidden state i at time 1. This probability is computed by assigning to each page a value equal to the percentage of pages with the same hidden state into the training set.

II. Transition Probabilities Matrix A:

$$A = [\alpha_{ij}]_{0 \le i < states, 0 \le j < states}$$

Where α_{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 estimated by counting the corresponding transitions from state L_i to L_j on the user training set, and by normalizing by the overall number of transitions from state L_i.

III. Emission Probabilities Matrix B :

$$B = [b_{ij}]_{0 \le i < states, 0 \le j < clusters}$$

Where b_{ij} represents the probability of being at cluster C_j given state L_i and $clusters$ is the number of clusters of pages. Probabilities are computed by counting the number of pages into cluster C_j with hidden state L_i and normalizing by the overall number of pages with hidden state L_i.

During crawling page content is processed and the HMM crawler assigns the page to a cluster using *K-Nearest Neighbors* algorithm [43]. Given the page cluster and the Hidden Markov Model parameters (π, A and B matrixes) the probability that the next page visited will be a target page is

computed using *Viterbi* algorithm [40]. That probability represents also visit priority of the link. The *Viterbi* algorithm computes a prediction of the state in the next time step given the sequence of web pages observed thus far. 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 system is in hidden state L_j at time t, based on observations made thus far. Given values $a(L_j,t-1)$ of parent pages, values $a(L_j,t)$ are computed using the following recursion:

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

Where a_{ij} is the transition probability of state L_i to L_j from matrix A and b_{jc_t} is the emission probability of cluster c_t from hidden state L_j from 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 the system will be in state L_j at the next time step is computed as follows:

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

The probability of been at state L_0 (relevant page) in the next step is the priority assigned to pages.

Chakrabarti et.al [32] proposed a two classifier approach. The open directory (DMOZ) [39] Web taxonomy is used to classify downloaded pages as relevant or not, and feed a second classifier which is trained using these pages. The second classifier is used to evaluate 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 to assign visit priority values to pages is presented in [33]. Classifiers based on Support Vector Machines [38] (SVM) seem to outperform Bayes Classifiers and classifiers based on Neural Networks on that task.

Resent contributions to the field of learning crawling include Hybrid crawlers [35] combining ideas from learning and classic focused crawlers. In [35] a Hybrid Crawler is proposed: 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 breadth first crawler. In our work, we apply a hybrid method for improving the performance of learning crawlers. However, instead of alternating crawlers between two modes of operation (Learning or Breadth first crawler) we combine the page priority functions computed by a Hidden Markov Model Crawler and that of the Best First Crawler in order to evaluate the overall priority value of a Web page.

2.6 Summary

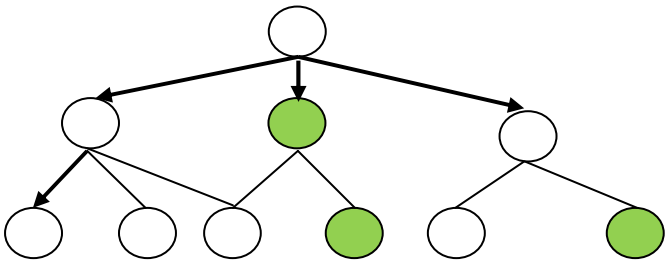
Related work on focused crawlers includes classic, semantic and learning approaches. The Best First Crawler and variations of this method (e.g. N-Best First Crawler) form a common and effective approach for focused crawling [26]. Semantic crawlers presented in [13] are not well studied and a comparison with state of the art classic focused crawlers such as Best-First hasn't appeared in the literature before. Learning crawlers form a distinctive category of focused crawlers based on a training set provided by the user for topic description. Learning crawlers based on SVM classifiers for assigning page visiting priorities achieve good performance [33], while methods that learn paths leading to relevant to the topic pages such as Context Graph method [31] and Hidden Markov Model Crawlers [16,18] are of great

importance. Also the newly proposed hybrid methods [35] are very promising approach to focused crawling.

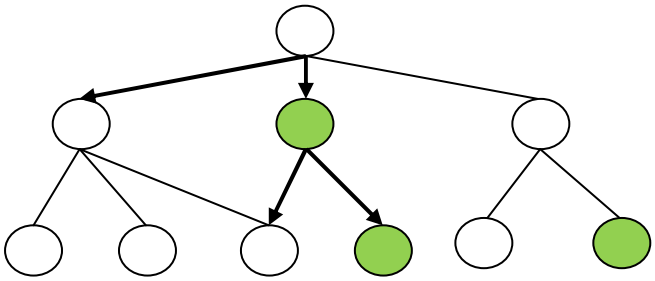
Chapter 3. Crawler Design

3.1 Introduction

Issues related to design and implementations of focused crawlers are discussed next. Given an application (general purpose web search engine or topic specific digital library) the appropriate type of web crawler has to be determined first. For the first application type, a breadth first crawler is a reasonable solution. Focused crawlers (classic, semantic or learning crawlers) are best suited for the later application type.



Breadth First Crawler



Focused Crawlers

Green circles denote relevant to the topic pages and arcs links between Web pages. Arrows denote visit sequence using different crawlers. Focused Crawlers assign higher visit priorities to links contained in relevant to the topic pages.

Fig. 5 Crawler Operation

Fig. 5 demonstrates the search stages of a crawler. Web pages are denoted by circles (green circles correspond to pages relevant to the topic at hand) while links denote outgoing links from a page. The crawler retrieves pages from the web starting with a seed page shown at the root of the tree. As discussed in the introduction, the outgoing links (URLs) of each visited page are placed in a queue from which the web page to visit next is selected in some order. The crawler gets the URL, download the page and places URLs extracted from the downloaded page in the queue. This process is repeated until the crawler decides to stop (e.g. disk space exhausted, time lapsed or the user is satisfied with the results). Focused crawlers introduce a number of criteria (e.g. page importance, relevance to topic) for assigning priorities to web pages in the queue and for selecting which page to visit next. Fig. 6 illustrates the operation stages of a crawler:

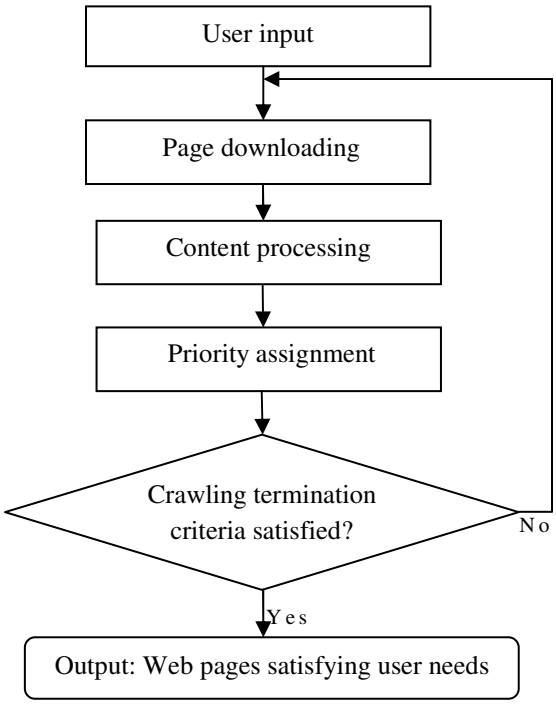


Fig. 6 Overview of Crawler operation

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**: Pages from queue are downloaded in some order. Focused crawlers may decide to exclude pages not satisfying the topic criteria from further investigation. Pages are stored locally at a page repository for further processing.

c) **Content processing**: The page content is lexically analyzed and reduced into term vectors (all terms are reduced to their morphological roots by applying Porter's stemming algorithm [48] and stop words are removed). Each term in a vector is represented by its term frequency-inverse frequency vector (*tf-idf*) according to VSM. The outgoing links of the page are also extracted and placed in the priority queue.

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.

All Crawlers follow the above design. Breadth First Crawler requires only seed pages as input. Best-First and Semantic

Crawlers take the seed pages and a user query as input while Learning Crawlers accept a training set of URLs of pages instead of a query. Crawlers also differ in the way priorities are assigned to extracted URLs. This is the most crucial part in the implementation of focused crawlers.

All Crawlers in this work are implemented in Java [36] using Eclipse [37]. The downloaded pages must be of text/html format and their content size must not exceed 100KB. Restrictions are also imposed on connection timeout and downloading times for performance reasons. Those restrictions apply to all implemented crawlers. The crawling process is repeated until the predefined number of pages is retrieved (Fig. 6). In or experiments this number is set equal to 1000 web pages.

3.2 Classic Crawlers

The ***Breadth First Crawler*** forms the baseline for implementing Best First, Semantic and Learning Crawlers. It is a simple program that gets one or more seed pages as input and follows the links in a breadth first way until the desired number of Web pages is downloaded. Fig. 7 illustrates the interface of the Breadth First crawler implemented. It accepts one or more seed pages as input. Downloaded pages are shown below.

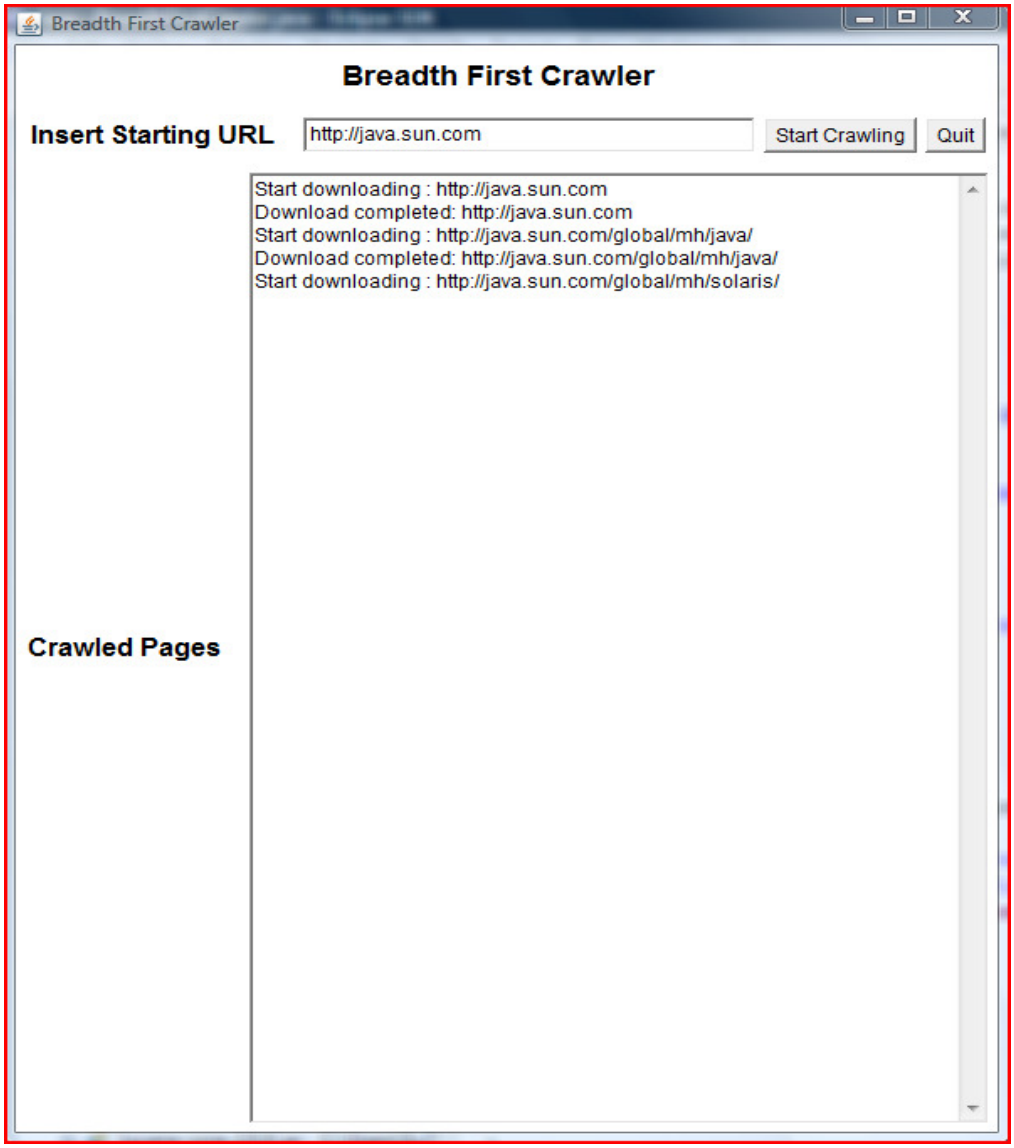


Fig. 7 Screenshot of Breadth First Crawler

3.2.1 Best First Crawler with page content criteria

The second classic Crawler (and the first focused) implemented is the ***Best First Crawler using page content*** for prioritizing candidate URLs. When a Web page is downloaded its content is lexically analyzed and represented by term vectors. Each term in such vector is represented by its *tf-idf* weight according to VSM [12]. Priority assigned to a link equals the cosine similarity (Eq. 2) of the page containing the link and the user query.

Notice that using inverse document frequency (*idf*) weights can be problematic because *idf* weights need to be updated at every crawling step, for this reason *idf* weights can be inaccurate at the initial steps of crawling when the number of retrieved pages is small [25]. Most Best First Crawler implementations use only term frequency (*tf*) weights. In this work *idf* weights are provided by the IntelliSearch web search engine [41] holding *idf* statistics for English terms. At the next step the link with the highest priority is selected for downloading.

3.2.2 Best First Crawler with anchor text similarity

The second variation of Best First Crawler is the ***Best First Crawler using anchor text similarity***. The anchor text of a URL is the clickable text that appears on the link inside a Web page pointing to that URL. In this work we implemented a variant of the above Best First Crawler which instead of page content uses URLs anchor text as the representation of page content and for assigning download priorities. Notice that links from the same page may be assigned different priority values, as opposed to the first implementation, using page text content for assigning priorities, where all links into the same page are given the same priority. As will be shown in the results, selection of anchor text for assigning priority values improved the general performance of the crawler, using both harvest ratio and average similarity criteria (section 4.3).

3.2.3 Best First Crawler with page content and anchor text.

The third variation of Best First Crawler combines the previous two implementations using page content and link

anchor text respectively. Each URL is assigned a priority value defined as:

$$Priority(i) = \frac{\text{similarity}(p_i, q) + similarity\ (a_i, q)}{2} \quad (7)$$

Where $priority(i)$ is the priority value assigned to link i, $\text{similarity}(p_i, q)$ is the similarity of query q and p_i (the content of the page where the link i is located) and $\text{similarity}(a_i, q)$ is the similarity of anchor text a_i of link i and query q.

The idea behind the Best First Crawler with page content only is that a page relevant to the topic is more likely to point to a relevant page than to a non relevant one. Thus, the higher the relevance of the page containing the link is, the higher the probability that the link will point to a relevant page is.

The second implementation (Best First Crawler using anchor text similarity) tries to overcome a disadvantage of the Best First Crawler with page content only: all links within a page have the same priority regardless of anchor text. Anchor text may be regarded as a summary of the content of the page that the link points to. Therefore it is reasonable to use this descriptor for assigning priorities to pages. However anchor text isn't always descriptive of page contents and by ignoring the page content useful information may not be used. So the third Best First Crawler implementation uses both page anchor text and page content.

3.3 Semantic Crawlers

Best First crawlers estimate the relevance between the page content or anchor text and a user query. There may exist conceptually related terms in both the query and the page (or anchor text), indicating a relevance to the topic. However if these terms aren't lexicographically similar their relevance

will be ignored because VSM computes text similarity as a function of similarities between identical terms found in the vectors which are compared. This can be resolved using ontologies or term taxonomies. In ontologies conceptually similar terms are related by virtue of IS-A links. All terms conceptually similar to user query terms are retrieved from the ontology and used for enhancing the description of the topic (e.g. by adding synonym terms to the topic keywords) and for computing the similarity between query and candidate pages. For this, various methods have been proposed including among others Semantic Similarity Retrieval Model (SSRM) [14] and Mihalcea et.al [15]. The most important representatives of this category of methods are implemented within Best First crawlers forming the so called hereafter Semantic Crawlers.

In this work, WordNet [4] term taxonomy is used as an ontology for retrieving conceptually similar terms. WordNet was selected because it provides a vast coverage of the English vocabulary so it can be used for focused crawling on almost every topic making our implementation the first general purpose Semantic Crawler. The general design remains similar to that of Classic Focused Crawlers (Fig. 6) but the priorities assigned to links are evaluated using methods such as SSRM [14] and Ehrig et.al [13]. Other parts of the system such as downloading, link and anchor text extraction, preprocessing and representing texts using Vector Space Model term vectors, remain the same.

In the following, candidate links for downloading are represented by their anchor texts. Each candidate link is assigned a priority value which is computed as the semantic similarity between their anchor text and the topic [14, 15]. In turn, semantic text similarity is computed as a function of

the semantic (conceptual) similarities between the terms they contain. This can be defined in many different ways [11] leading to the implementation of three semantic crawlers.

3.3.1 Ehrig Crawler

In this implementation Web pages are represented by the anchor text of the links pointing to them (instead of page content as in [13]. Anchor texts and the user query are represented by term vectors using *tf* weights [13]. Page priorities are computed as:

$$Priority_{ehrig}(i) = \sum_{j=0}^{j=T} \sum_{k=0}^{k=T} sim_{df}(t_j, t_k) * w_{ij} * w_{qk} \quad (8)$$

Where T is the total number of terms into anchor text and query, and sim_{df} is term semantic similarity computed using equation 3. Note that only *tf* weights are used without normalizing by vector length (as it is recommended for short topic and page descriptions), and that WordNet is used instead of topic specific ontologies as in [13].

3.3.2 SSRM Crawler

SSRM [14] is used for assigning visit priorities to web pages. Specifically the priority of a URL (represented by its anchor text) is defined as follows:

$$Priority_{SSRM}(i) = \frac{\sum_{k=0}^{k=T} \sum_{j=0}^{T} sim_{li}(t_k, t_j) w_{ik} * w_{qj}}{\sqrt{\sum_{j=0}^{j=T} {w_{ij}}^2} \sqrt{\sum_{j=0}^{j=T} {w_{qj}}^2}} \quad (9)$$

Where T is the total number of terms into the anchor text and the query. Li et.al.[42] is the term similarity method

used in our implementation. The URL with highest priority value is downloaded first.

3.2.3 Semantic Crawler with synonym set expansion

An obvious improvement is to expand text vectors with synonym sets in WordNet and use both anchor text and page content for computing text similarity and assigning priorities:

$$Priority_{synset\ expand}(i) = \frac{\text{similarity}(p_{i,}\, \dot{q}) + similarity\ (\dot{a}_{i,}\, \dot{q})}{2} \quad (10)$$

Where $Priority(i)$ is the priority value assigned to link i, $similarity(p_{i,}\, \dot{q})$ is the cosine similarity of expanded query $\dot{q}$ (using WordNet synonym sets) and p_i (the content of page where the link i is located) and $similarity\ (\dot{a}_{i,}\, \dot{q})$is the cosine similarity of expanded anchor text $\dot{a}_i$ of link i and expanded query $\dot{q}$.

3.4 Learning Crawlers

The main idea behind Learning Crawlers is that the crawler learns user preferences on the topic from a set of example pages (the training set). Training may involve learning the path leading to the desired content. In most cases the training set consists of relevant and irrelevant pages. Every downloaded page is classified (based on the results of learning) as relevant or irrelevant and is assigned a priority. The Context Graph method [31] works not only by classifying the crawled pages as relevant of not relevant, but also by learning the distance (in number of routing hops) that may lead from an irrelevant page to a relevant one (Fig 3). The non relevant pages in the training set were downloaded

using recursively Google's backlink service, starting from relevant pages, in order to compute their distance (level) from the relevant or target pages. During crawling, pages similar to those closer to target pages are given higher priority.

The Hidden Markov Model Crawler [16, 18] extends the previous idea by categorizing pages not only by their distance from a target page but also by using their content, thus estimating a relation between page content and the path leading to relevant pages. Initially, a user browses a sequence of pages labeling them as relevant or not. As pages are downloaded, the visiting sequence is recorded and a context graph is created without the need of a backlink service as in [31].

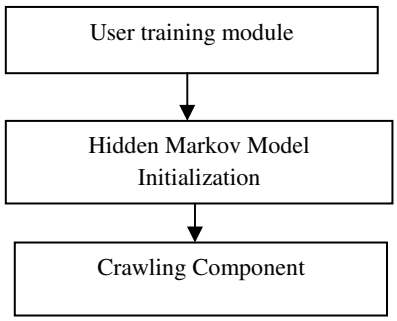


Fig. 8 Outline of learning focused crawling

Figure 8 illustrates the functional components of the HMM crawler implemented:

I. **Training component**: The first component records the URL's visited by the user and the page view sequence. Then it downloads pages and computes the *tf-idf* vectors representing their content. Finally pages are clustered using a clustering algorithm. In our implementation K-Means and X-Means [17] were used for clustering.

II. **HMM initialization**: The second component takes the HMM representation of user training set (as in fig. 4) as

input and computes the Hidden Markov Model Parameters (i.e. π, A and B matrixes). This component is applied during the initialization phase before crawling.

III. **Crawling component**: it downloads selected pages, extracts content and links, process page content and assigns the page to a cluster using *K-Nearest Neighbors* algorithm [43]. Given the page cluster and the Hidden Markov Model parameters (π, A and B matrixes) the probability that the next page visited will be a target page is computed using *Viterbi* algorithm [40]. That probability represents also visit priority of the link. If two clusters yield almost identical probabilities (i.e. the difference of 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 the *Viterbi* algorithm).

Three Learning crawlers have been implemented: the first is the Hidden Markov Crawler (variants proposed in [16] and [18]). The next two variants (Hybrid Crawlers) are proposed in this thesis. They combine the page priority functions computed by the Hidden Markov Model Crawler and that of the Best First Crawler in order to evaluate the overall priority value of a Web page.

3.4.1 Hidden Markov Model Crawler

Two variants of this crawler have been implemented:

a) The first hidden Markov Model implementation uses K-Means algorithm for clustering as described in [16]. In this work the dimensionality reduction step is omitted. K was set to 5, and the last fifth cluster

holds the relevant pages. Page priorities ($priority_{hmm}$) are computed using *Viterbi* [40] algorithm (Fig. 9).

b) The second variant is almost identical to the previous one but instead of K-Means, X-Means [17] is used. Other minor modifications are (a) *idf* weights are not precomputed (as in the previous variant), but are computed using the training set and (b) the relevant pages don't form a separate cluster but they may belong to the same cluster with non relevant pages. As will be shown in the experiments the two variants demonstrated identical performance. The first variant was used for comparisons with the Hybrid Crawlers proposed in this work.

Figure 9 summarizes HMM Crawlers priority assignment procedure:

Input: Training set, candidate page (*p*).

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

1. Cluster training set using K-Means or X-Means algorithm
2. Compute π, A, B matrixes.
3. Classify candidate page *p* to a cluster c_t using K-Nearest Neighbor algorithm
4. Compute hidden state probabilities for current step using Viterbi formula:
$$a(L_j, t) = b_{jc_t} \sum_{i=0}^{states} (a(L_i, t-1) * a_{ij})$$
5. Compute hidden state probabilities estimation for next step using formula :
$$a(L_j, t+1) = \sum_{i=0}^{states} (a(L_i, t) * a_{ij})$$
6. Assign priority $priority_{hmm}(p) = a(L_0, t+1)$ to page p.

Fig. 9 HMM Crawler priority estimation algorithm

3.4.2 Hybrid Crawlers

Two variants of hybrid crawlers are implemented:

a) **Hybrid Markov Model Crawler**: The Hidden Markov Model Crawler suffers from at least two drawbacks: (a) it doesn't assign different priorities to pages belonging to the same cluster and (b) it is very difficult to represent the set of Web pages not relevant to the topic by clusters (it is a very heterogeneous set).

A hybrid approach combining the text similarity of a page with the centroid of the cluster containing the positive example pages (using VSM) is proposed here for dealing with these two problems. The centroid is computed as the average vector of the pages belonging to the cluster. Text similarity between candidate pages with the centroid may differ even if pages belong to the same cluster thus dealing with the first problem mentioned. Similarity with the centroid of relevant pages is not affected by the way non relevant pages are represented thus dealing with the second problem as well.

The Hybrid Markov Model Crawler differs from the HMM Crawler in the way priorities are assigned to candidate pages. It computes a priority score for a page using the Hidden Markov Model ($priority_{hmm}$) and also computes the text similarity of page content with the centroid of the cluster containing the relevant pages from the user training set using equation 2. Finally, the priority of page p_i is computed as follows:

$$priority_{hybrid}(p_i) = \frac{similarity(p_i, c_r) + priority_{hmm}(p_i)}{2} \quad (11)$$

Where c_r is the centroid of relevant pages in training set, similarity(p_i, c_r) is the cosine similarity of page content p_i with centroid c_r of relevant pages, $priority_{hmm}(p_i)$ is the priority assigned to page link i into page p_i using Hidden Markov Model and $priority_{hybrid}(p_i)$ is the priority assigned to links in page p_i by the Hybrid Crawler.

b) **Hybrid HMM Crawler with page content and anchor text**: An obvious extension to method (a) is to use both anchor and page text in the computation of page priorities. This lead to the following equation:

$$priority_{hybrid\ anchor}\ (a_i) = \frac{priority_{hmm}(p_i) + \frac{\text{similarity}(p_i, c_r) + \text{similarity}(a_i, c_r)}{2}}{2} \quad (12)$$

Where c_r is the centroid of relevant pages in training set, $similarity(p_i, c_r)$ is the cosine similarity of page content p_i with the centroid c_r of relevant pages, $similarity(a_i, c_r)$ is the cosine similarity of link anchor text a_i with the centroid c_r of relevant pages, $priority_{hmm}(p_i)$ is the priority value assigned to links into page p_i using Hidden Markov Model and $priority_{hybrid\ anchor}\ (a_i)$ is the priority assigned to the link with anchor text a_i by the Hybrid HMM Crawler with page content and anchor text.

The priority function of equation 12 improves the performance of the hybrid Crawler. As will be shown in the experiments when anchor text is used the crawler is even more focused to the topic. Figure 9 illustrates the operation of hybrid crawlers:

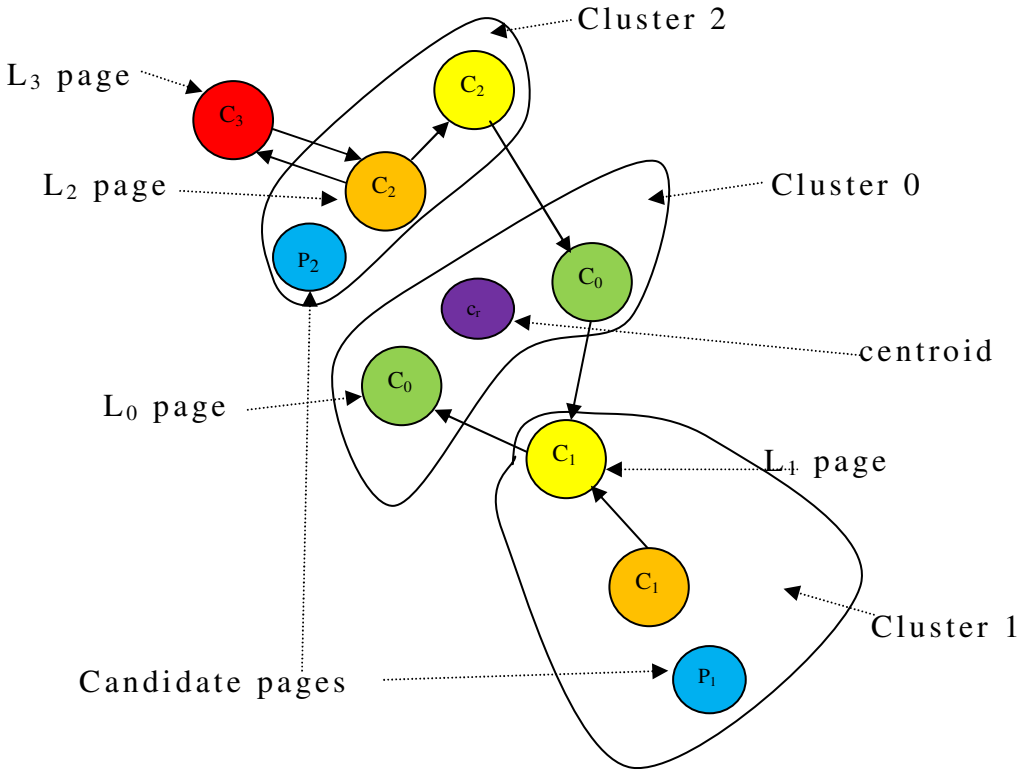


Fig. 10 Hybrid crawlers operation.

In figure 10 two pages (blue circles) are candidate for downloading. The HMM Crawler will assign higher priority to candidate page p_1 belonging to cluster 1 since this cluster leads with higher probability to target pages (cluster 0) in two link steps (since the probability of leading to cluster 0 in one step is identical for clusters 1 and 2). Instead, a Hybrid crawler will select for expansion the page p_2 belonging to cluster 2 because of its proximity (similarity) with the centroid of cluster 0 (the cluster containing the relevant pages from the training set).

3.5 Summary

Classic crawlers including the well known Breadth-First crawler and variations of the Best-First Crawler presented in this chapter have been implemented in the current thesis.

Semantic crawlers including a variation of the Ehrig crawler using WordNet, and the novel SSRM and Synonym set expansion crawlers have been implemented and compared with state of the art Best First Crawlers. Finally a set of state of the art HMM crawlers including [16,18] and the here proposed hybrid crawlers are also implemented and their performance is evaluated.

Chapter 4. Experimental Results

4.1 Introduction

The following set of experiments is designed to:

a) Provide a critical evaluation of the various types of crawlers examined in this work including classic (Breadth-First), topic driven (Best-First and its variants including Semantic crawlers), Learning and Hybrid crawlers.

b) Demonstrate the superiority of the new Hybrid crawler proposed in this work over state of the art HMM learning crawlers such as [16, 18].

Six different topics were used ("linux", "asthma", "robotics", "dengue fever", "java programming" and "first aid") and the ability of the crawlers to download pages on the above topics was measured. Their performance was computed using two well established measures referred to as harvest ratio and average similarity. Each crawler downloaded 1000 pages and its average performance (over all topics) was computed using both criteria. Relevant judged pages were provided by the user who manually inspected results obtained by the Google search engine on each topic. These results were used as ground truth and compared with results obtained by the crawlers. The more similar (to ground truth) the results of a crawler are, the most successful the crawlers is (the higher the probability that the crawler retrieves results similar to the topic). Page to topic relevance is computed by VSM in all cases.

4.2 Performance measures

Two different evaluation criteria were used:

a) **Harvest ratio:** For every page its cosine similarity with all pages judged as relevant by the user is computed and the maximum of these cosine similarities is taken. If the maximum similarity is greater than a predefined threshold (0.75 in this work) then the page is marked as relevant (otherwise the page is marked as irrelevant). The harvest ratio is defined as the percentage of downloaded pages with similarity greater than the threshold (in this thesis the number of relevant pages was used instead of the fraction of them among the total number of downloaded pages).
b) **Average similarity**. The maximum similarity of each downloaded page with all pages marked as relevant is computed. The average similarity is defined as the average value of these similarities for all downloaded pages.

The first criterion is more selective than the second. Harvest ratio can be adjusted (by using higher threshold) to measure the ability of the crawler to download pages highly relevant to the topic. An application called “evaluator” was developed for automating the evaluation process. It receives as input the positive pages set (50 relevant pages on every topic in our experiment) and the 1000 evaluated pages downloaded by the crawler, and computes the performance of the crawler at hand with both criteria.

4.3 Experiment setup

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 & anchor text
3) Semantic Crawlers:
   e) Semantic Crawler using Ehrig et.al. [13] method for text similarity estimation.
   f) Semantic Crawler using SSRM [14] method for text similarity estimation.
   g) Semantic Crawler with Synset Expansion.
4) Learning Crawlers:
   h) Hidden Markov Model Crawler
   i) Hybrid Hidden Markov Model Crawler
   j) Hybrid Hidden Markov Model Crawler with page content & anchor text.

All Crawlers were evaluated using the following topics and seed pages:

query	seed
Linux	http://dir.yahoo.com/Computers_and_Internet/Software/Operating_Systems/UNIX/Linux
Asthma	http://dir.yahoo.com/Health/Diseases_and_Conditions/Asthma/
Robotics	http://dir.yahoo.com/Science/Computer_Science/
Dengue Fever	http://health.yahoo.com/
Java programming	http://dir.yahoo.com/Computers_and_Internet/
First Aid	http://dir.yahoo.com/Health/

Fig. 11 Experiment setup

1000 pages were downloaded for each crawler and for each topic. Notice that in four out of the six topics the seed page doesn't directly link to target pages.

The experiments in this section are organized by crawler type showing a comparison between various implementations of the crawler of the same type. Specifically the experiments are organized as follows:

a) Classic Focused Crawlers Experiment
Crawlers (a)-(d) were evaluated using the six topics of Fig. 11.

b) Semantic Crawlers Experiments
Crawlers (e)-(f), and (c)-(d) for comparison, were evaluated using the 6 topics of Fig. 11.

c) Learning Crawlers Experiment
Crawlers (h)-(j) were evaluated using four topics ("Robotics", "Dengue Fever", "Java Programming" and "First Aid").

In the experiments below each method is represented by a plot showing number of relevant pages in the Y axis as a function of total number of pages retrieved. Each point in a plot corresponds to harvest ratio or average similarity measured respectively.

Notice that Learning Crawlers have different input (the training set) than the Classic and Semantic focused Crawlers (that have the user query as input) so direct comparisons between the performance of learning and other categories of crawlers in not really plausible.

4.4 Classic Focused Crawlers

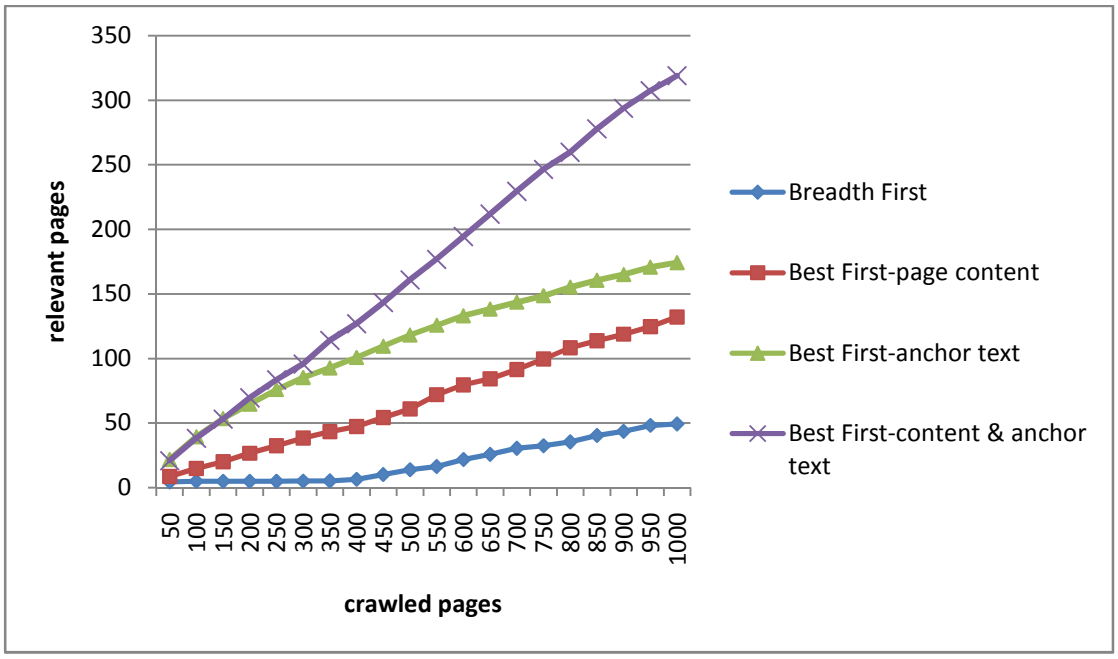


Fig. 12 Harvest ratio for classic crawlers

The comparison in Fig. 12 indicates the poor performance of Breadth First Crawler, as expected for a non focused crawler. The fact that the Best First Crawler using anchor text only outperforms the crawler using only page content indicates 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 is not computed by page or anchor text alone. Instead, the combination of page content and anchor text forms a more reliable page description.

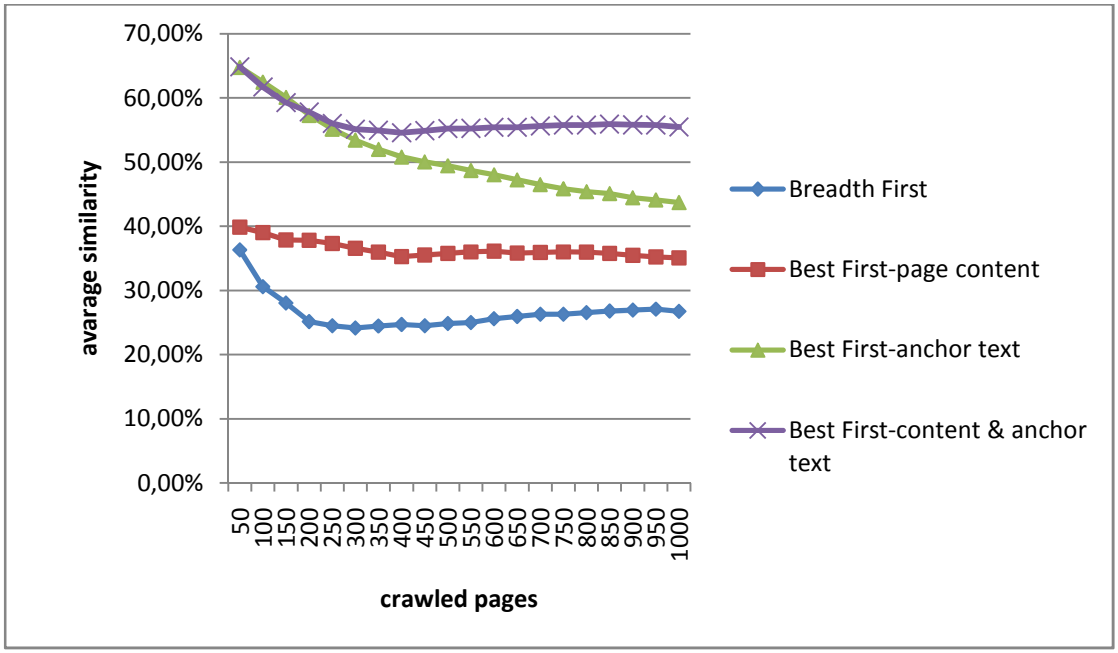


Fig. 13 Average similarity for classic focused crawlers

Fig. 13 confirms the results of the previous comparison. Overall a best first crawler combining page and anchor text achieves superior performance over all its competitors with both criteria.

4.5 Semantic Crawlers

The second experiment measures the performance of semantic crawlers using the six topics of Fig. 11 (as in the previous experiment).

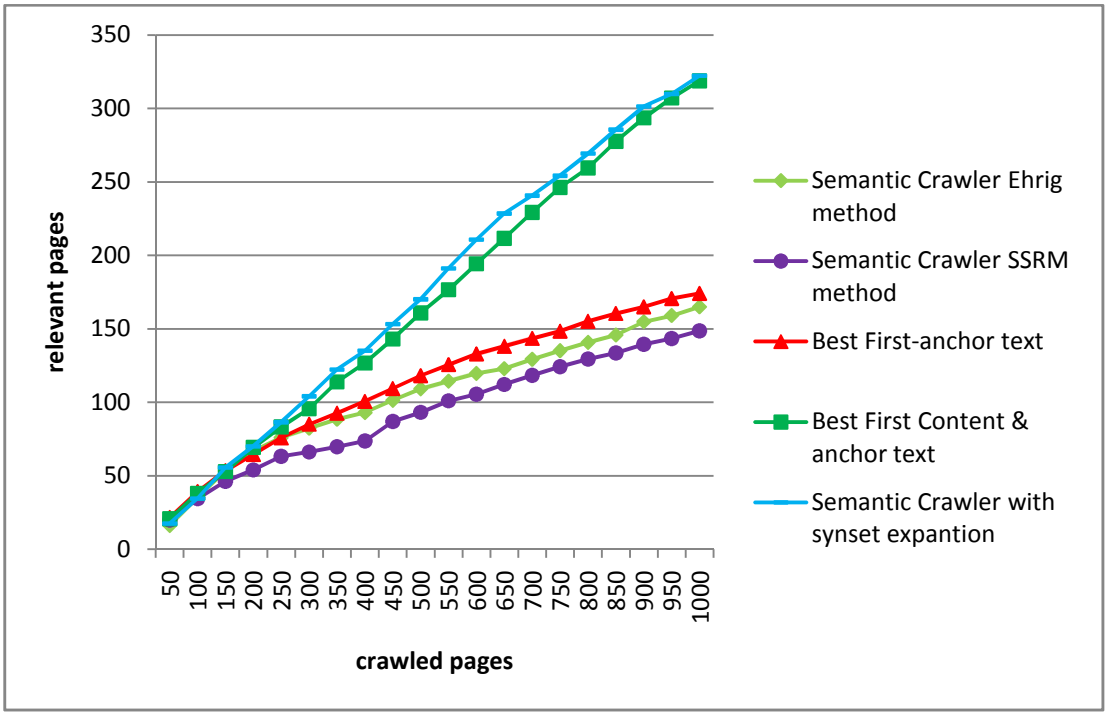


Fig. 14: Harvest Ratio for Semantic Crawlers.

Fig. 14 illustrates only marginal performance improvements of semantic crawlers over best first crawlers. It is conjectured that the poor performance of semantic crawlers 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 for English terms and not all linked terms are actually very similar, implying that the results can be improved by using topic specific ontologies. Such topic specific ontologies on several diverse topics were not available to us for these experiments.

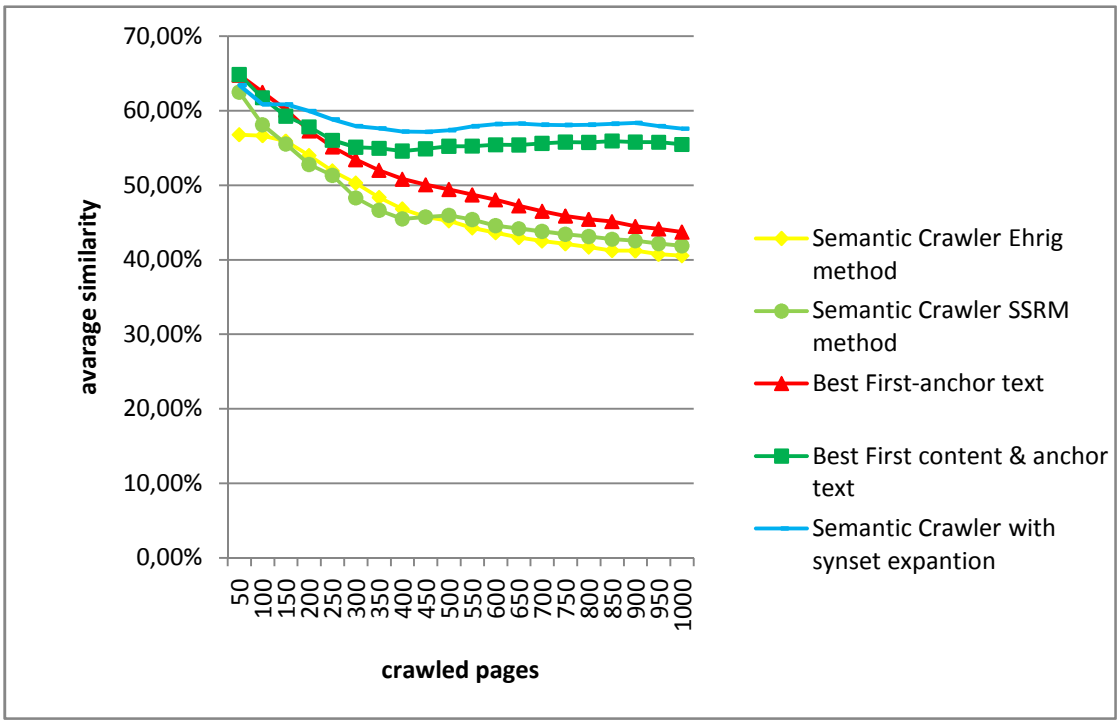


Fig 15: Average Similarity for Semantic Crawlers

Results with average similarity actually confirmed the results of Fig. 14. Here semantic crawlers improved again the results of best first crawlers but only marginally, indicating that average similarity (as less strict criterion) is more tolerant to relaxed interpretations of conceptual similarity as provided by WordNet and term similarity measures (such as Li et.al [42]).

4.6 Learning Crawlers

The results below are taken on four topics ("robotics", "dengue fever", "java programming" and "first aid") and measured on the first 1000 web pages returned by each crawler on each topic. Only Learning crawlers were evaluated in this experiment: Two variants of HMM Crawlers were tested corresponding to different implementation of the clustering component (with K-Means and X-Means

respectively). The results indicate that K-Means (using K=5 as suggested at [16]) and X-Means Hidden Markov Model Crawlers have identical performance. Both crawlers demonstrated poor performance (Figs. 16-17) and this can be attributed to several reasons: both variants don't assign different priorities to pages into the same cluster, and between links into the same page. Both variants must be provided with a training set very similar in content and link structure to the part of the Web that will be crawled (something not always achievable). Because the two HMM Crawlers (using X-Means and K-Means) have identical performance the first variant (HMM Crawler using K-Means) was chosen for comparison with the other Learning Crawlers.

In Fig. 16 the performance of the HMM crawler is compared with the performance of the new Hybrid crawlers (using combination of page content and anchor text) proposed in this work. The first (Hybrid HMM using page content) prioritizes links using equation 11 (similarity of the page containing the links with the centroid of the relevant pages in the training set). In addition to that the second implementation (Hybrid HMM Crawler with anchor text) also combines the similarity of the centroid with the anchor text of links pointing to candidate pages for priority assignment as suggested by equation 12.

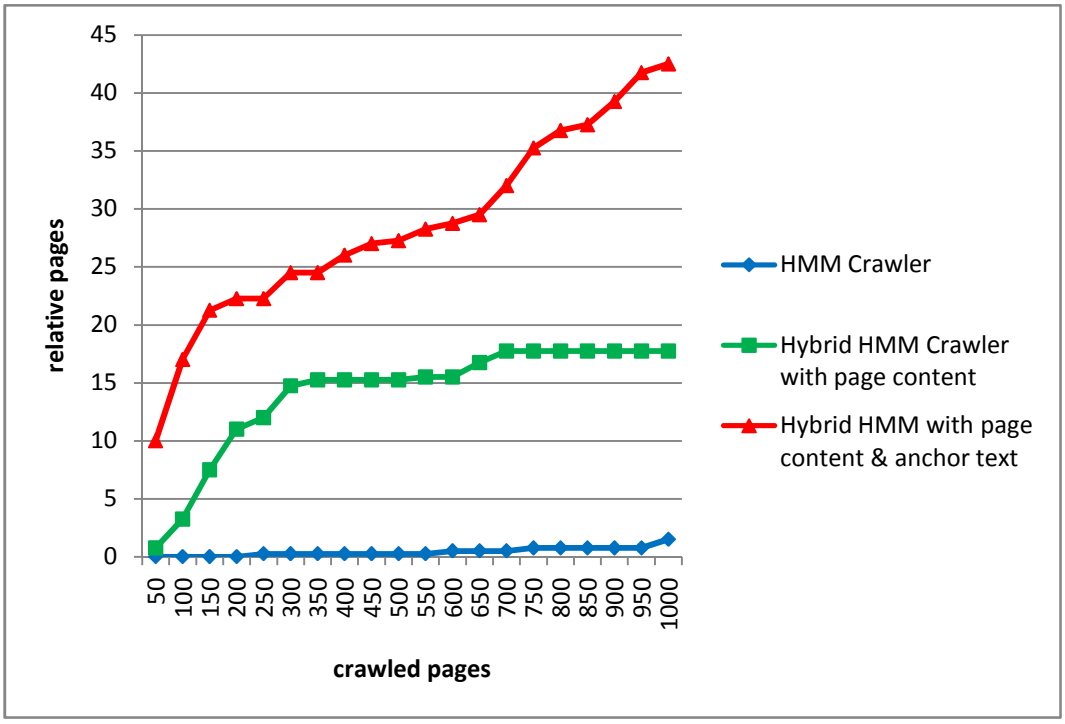


Fig. 16: Harvest Ratio for HMM & Hybrid Crawlers

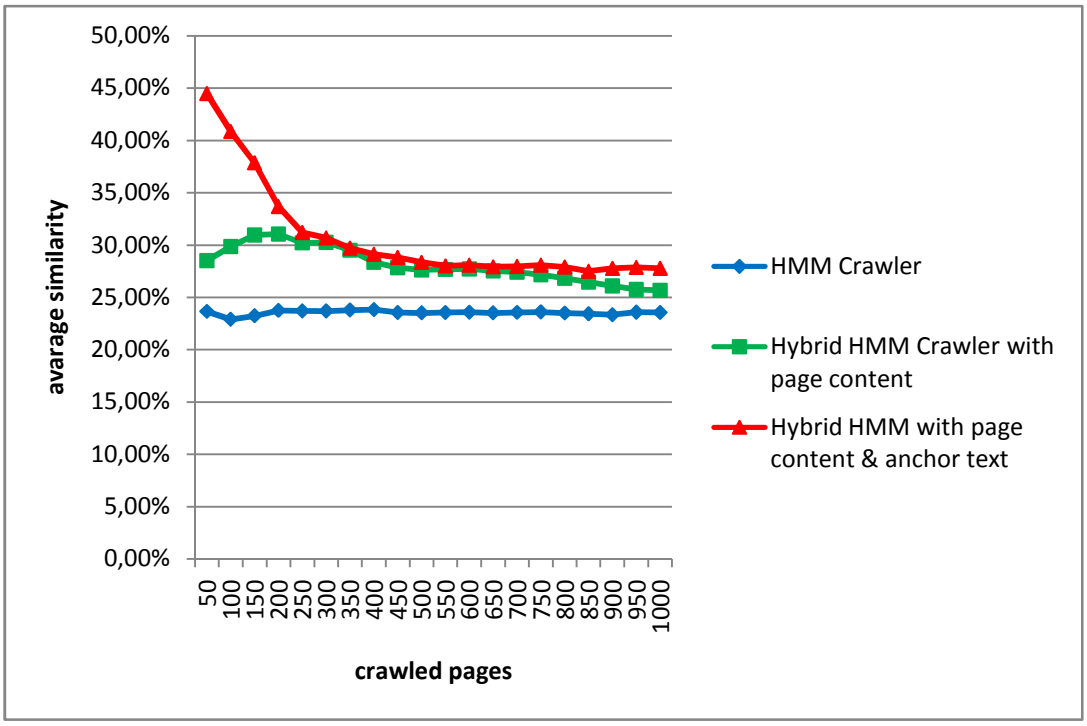


Fig. 17: Average Cosine Similarity for HMM & Hybrid Crawlers

The Hybrid crawlers outperform the Hidden Markov Model using both criteria. The use of positive examples centroid as

a query clearly increases performance because it overcomes the problems of HMM crawlers. As Fig. 16 and Fig. 17 indicate, the results obtained by Hybrid crawlers are promising and may lead to further research on this direction.

4.7 Discussion

Classic Focused Crawler results show that combining page content and anchor text (Best First Crawler -page content and anchor text) yields the best results. Both page content and anchor text form a representative content descriptor for web pages. Semantic Crawlers, when combined with a general purpose ontology, performed poorly compared to Best First crawlers. By restricting semantic relations to synonym sets (Semantic Crawler - Synset expand method) the performance was improved marginally. Synonyms, although not lexically similar succeed in identifying pages with content similar to the topic, indicating that it is possibly to expect further performance improvements by using topic specific ontologies rich in terms very similar to the terms of the topic. At this point, ontologies of this type are not available to us. Both Hybrid Crawlers achieve better performance than the Hidden Markov Model Crawler. The results obtained indicate that positive examples are more important than the negative ones during training in an environment such as the World Wide Web. Using only positive examples the performance of learning crawlers is expected to improve.

Chapter 5. Conclusions and future work

In the present thesis, several variants of focused crawlers were implemented and evaluated using common evaluation criteria. First the Breadth First Crawler and variants of the Best First Crawler using page content, anchor text or both were compared. Then semantic relations were used in the implementation of three Semantic Crawlers that were compared with classic focused crawlers (variations of best first crawler). Finally, based on the Hidden Markov Model learning crawler, two novel hybrid crawlers combining elements from learning and classic focused crawlers were implemented and evaluated.

The experimental results indicate that the implementation of focused crawlers is a process where minor changes in the crawler design have great effect in performance. The combination of anchor text and page content yields great performance improvement in the case of classic, semantic and learning focused crawlers. The addition of semantic relations didn't improve performance with the exception of expansion with synonyms where semantic relations are restricted to synonym terms. Performance is expected to improve 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 not described by a simple query. It is not only that Learning crawlers receive different input than that of other focused crawlers but also they are intended to perform a very difficult task: they attempt to learn web crawling patterns

leading to relevant pages possibly through other non relevant pages thus increasing the probability of failure (since web structures cannot always be modeled by such link patterns). However the idea looks promising overall 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.

Another direction for future work would be to do more elaborate tests with semantic crawlers, making use of topic specific ontologies (e.g. medical ontologies for applications related to healthcare). The positive results obtained by hybrid crawlers indicate that the relevance of a candidate page with the set of positive examples only, is an effective way for assigning priorities to candidate pages. Using only positive examples (instead of positive and negative) might improve the performance of learning crawlers in terms of speed and accuracy.

References:

[1] “Web Search for a Planet: The Google Cluster Architecture” LA Barroso, J Dean, U Holzle - Micro, IEEE, 2003.

[2] “Very Large Scale Retrieval and Web Search” D Hawking, N Craswell, In E. Voorhees and D. Harman, editors, TREC: Experiment and Evaluation in Information Retrieval. MIT Press, 2005.

[3] “The Indexable Web is More than 11.5 Billion Pages” A Gulli, A Signorini - International World Wide Web Conference, 2005.

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

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

[6] “The Anatomy of a Large-Scale Hypertextual Web Search Engine” S Brin, L Page WWW7 / Computer Networks, 1998.

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

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

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

[10] http://larbin.sourceforge.net/index-eng.html

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

[12] “A Vector Space Model for Automatic Indexing” G Salton, A Wong, CS Yang –Communications of the ACM, 1975.

[13] “Ontology-Focused Crawling of Documents and Relational Metadata” Alexander Maedche, Marc Ehrig, Siegfried Handschuh, Raphael Volz, and Ljiljana Stojanovic. Proceedings of the Eleventh International World Wide Web Conference WWW-2002.

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

[15] “Measuring the Semantic Similarity of Texts.” Corley,C., Mihalcea, R.:,Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment. Ann Arbor, June 2005.

[16] “Focused Crawling by Learning HMM from user’s Topic-Specific Browsing.” H. Liu, E. Milios, and J. Janssen. In Proceedings of 2004 IEEE/WIC/ACM International Conference on Web Intelligence, pages 732–735, Beijing, China, September 20-24, 2004.

[17] “X-means: Extending K-means with Efficient Estimation of the Number of Clusters.” D. Pelleg and A. Moore. In Proceedings of the 17th International Conf. on Machine Learning, pages 727–734. Morgan Kaufmann, San Francisco, CA, 2000.

[18] “Using HMM to Learn User Browsing Patterns for Focused Web Crawling” H Liu, J Janssen, E Milios - Data & Knowledge Engineering, 2006.

[19] “Breadth-First Search Crawling Yields High-Quality Pages.” M. Najork and J. L. Wiener. InProc. 10th International World Wide WebConference, 2001.

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

[21] “Searching the Web.” Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan. Transactions on Internet Technology, 2001.

[22] “Efficient Crawling Through URL Ordering.” Junghoo Cho, Hector Garcia- Molina, Lawrence Page. Seventh International Web Conference (WWW 98). Brisbane, Australia, April 14-18, 1998.

[23] “Information Retrieval in Distributed Hypertexts” P. De Bra, G.-J. Houben, Y. Kornatzky, and R. Post, in: Proceedings of RIAO'94, Intelligent Multimedia, Information Retrieval Systems and Management, New York, NY, 1994.

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

[25] “Evaluating Topic-Driven Web Crawlers” F. Menczer, G. Pant, M. Ruiz, P. Srinivasan, , Proc. 24th Annual Intl. ACM SIGIRConf. on Research and Development in Information Retrieval,ACM Press, New York, NY, 2001

[26]“Topical Web Crawlers: Evaluating Adaptive Algorithms” F Menczer, G Pant, P Srinivasan – ACM Transactions on Internet Technology (TOIT), 2004.

[27] “A General Evaluation Framework for Topical Crawlers” P Srinivasan, F Menczer, G Pant – Information Retrieval, 2005 – Springer.

[28] “Intelligent Crawling on the World Wide Web with Arbitrary Predicates.” C. Aggarwal, F. Al-Garawi, and P. Yu. In Proc. 10th Intl. World Wide Web Conference, pages 96–105,2001.

[29] “A Survey of Focused Web Crawling Algorithms.” Novak, B. Proceedings of the 7th International multi-conference Information Society IS-2004, Ljubljana: Institut “Jožef Stefan”, 2004.

[30] “Focused Crawling: A New Approach for Topic Specific Resource Discovery” S Chakrabarti, M van den Berg, B Dom - WWW Conference, 1999.

[31] “Focused Crawling Using Context Graphs.” M. Diligenti, F. Coetzee, S. Lawrence,C. L. Giles, and M. Gori. In Proc. 26thInternational Conference on Very LargeDatabases (VLDB 2000), pages 527–534,Cairo, Egypt, 2000.

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

[33] “Learning to Crawl: Comparing Classification Schemes” G Pant, P Srinivasan – ACM Transactions on Information Systems (TOIS), 2005.

[34] “Focused Crawling by Exploiting Anchor Text Using Decision Tree” Li Jun, Furuse K,Yamaguchi K. C , Proceedings of the 14th International World Wide Web Conference.2005:1190-1191.

[35] “A Novel Hybrid Focused Crawling Algorithm to Build Domain-Specific Collections” Y Chen, PhD thesis – 2007.

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

[37] http://www.eclipse.org/

[38] “A Tutorial on Support Vector Machines for Pattern Recognition” CJC Burges - Data Mining and Knowledge Discovery, 1998.

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

[40] “The Viterbi Algorithm” GD Forney - Proceedings of the IEEE, 1973.

[41] “IntelliSearch: Intelligent Search for Images and Text on the Web” E Voutsakis, EGM Petrakis, E Milios. 3rd Intern. Conference on Image Analysis and Recognition (ICIAR 2006), pp. 697-708, Sept. 18-20, 2006, Povoa de Varzim, Portugal.

[42] “An Approach for Measuring Semantic Similarity between words using Multiple Information Sources” Y Li, Z Bandar - IEEE Transactions on Knowledge and Data Engineering, 2003.

[43] “Nearest Neighbor Pattern Classification” T Cover, P Hart - Information Theory, IEEE Transactions on, 1967.

[44] “An Introduction to Hidden Markov Models” L Rabiner, B Juang - ASSP Magazine 1986.

[45] “Mercator: A Scalable, Extensible Web Crawler” A Heydon, M Najork – World Wide Web, 1999 – Springer.

[46] “Mining the Link Structure of the World Wide Web” Soumen Chakrabarti, Byron E. Dom, David Gibson, Jon Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. IEEE Computer, 32(8):60-67, 1999.

[47] “Data Clustering: a Review” AK Jain, MN Murty, PJ Flynn - ACM Computing Surveys (CSUR), 1999.

[48] “An Algorithm for Suffix Stripping” Porter, M.F. (1980) Program, 14(3): 130- 137.