A Comparison over Focused Web Crawling Strategies

Ioannis Avraam
Department of Mathematics
Aristotle University of Thessaloniki
Thessaloniki, Greece
iavraam@auth.gr

Ioannis Anagnostopoulos
Dpt of Computer Science and Biomedical Informatics
University o of Central Greece
Lamia, Greece
janag@ucg.gr

Abstract—In this paper we review and compare focused crawling strategies, studied and published during the past decade. Despite giant leaps in communication, storage and computing power in recent years, crawlers have always struggled to keep up with Web content generation and modification. Focused crawlers attempt to i) accelerate the crawling process, ii) maximize the harvest of high quality pages, iii) assign appropriate credit to different documents along a crawling path, such that short-term gains are not pursued at the expense of less obvious paths that ultimately yield larger sets of valuable pages. Beyond the review and comparison of the focused crawling strategies, we additionally propose additions to the corresponding architectures for further research.

Keywords: Focused crawling, adaptive crawling, location-based Web search, context graphs.

I. INTRODUCTION

Due to the current size of the Web and its dynamic nature, building an efficient search mechanism is very challenging and it still remains an active area of research. Topical search is increasingly becoming a major focus point of research interest. The focused Web crawling techniques are commonly used to build search mechanisms able to auto-focus on one or few target topics. The main goal of a focused crawler is to selectively seek out pages that are relevant to a pre-defined set of topics while traversing the minimal portion of Web. The effective focused crawling is very hard to accomplish due to some characteristics of WWW such as its enormous size, its rapid rate of change and the dynamic nature of the user-generated content (UGC).

In this paper we present a detailed comparison of the most popular and sophisticated focused crawling strategies. Though this comparison, we are able to conclude to the most substantial elements of the crawling architectures. These elements are the ones that highly affect the overall performance of focused crawlers. Based on our findings, we are confident to propose specific additions to the focused crawling systems proposed in the past. It is our belief that our suggestions should be considered by the research community for further experimental research in the area of focused crawling. Furthermore, we present the graph of papers in the area of focused crawling, published in the WWW conferences of the past decade. The division of the papers of the graph into communities based on their topics of research, not only reveals that the different areas of research related to focused crawling are closely related but also highly interactive.

II. FOCUSED CRAWLING ARCHITECTURES

A. Focused Crawling with Reinforcement Learning

Most simple focused crawlers use one Baseline topic classifier called learner, trained offline for prioritizing unvisited pages of the crawl frontier. As shown in [1] a focused crawler with reinforcement learning has significantly lower loss rate. The proposed system uses 2 classifiers in sequence: A Naive Bayes Baseline learner, trained offline and a Naive Bayes apprentice, trained online. The Baseline learner acts as a trainer for the apprentice, which is responsible for assigning priorities to the pages of the crawl frontier.

B. Context Focused Crawling

Most typical focused crawlers provide credit to paths that lead to highly relevant pages, while ignoring less promising paths. As shown in [2], a focused crawler equipped with the ability to model the context within which relevant documents are found, has significantly higher harvest rate. The Context Focused Crawler uses a technique that recognizes less promising
paths that ultimately yield high quality pages. The proposed crawler performs a crawl session in 2 distinct phases: i) the initialization phase, ii) the focused crawl phase.

In the first stage of the initialization, the crawling algorithm builds a general and compact context called a Context Graph, using link hierarchies. The system is initialized with a set of seed pages. Starting with each of the seed pages, the crawler performs back-crawling using Google. Thus, a set of pages linking to seed pages is retrieved. These pages compose layer 1 of the context graph.

In the second phase of the crawling session, the system performs focused crawling using the aforementioned classifiers. All the retrieved pages are classified into a sequence of queues each one associated with one layer of the graph. Documents that cannot be classified to any layer are assigned to a queue named “other”. In each iteration of the crawling algorithm, the crawler pops the page with the highest priority from the first non-empty queue, thus retrieving documents closely related to the target topic earlier in the crawling session. The crawler then downloads and classifies the children of the current page.

C. An Intelligent Web-Forum Crawler - iRobot

The design of an intelligent forum crawler, called **iRobot**, that deals effectively with the above difficulties is presented in [3].

The system consists of two main parts: i) the offline part, used for training and initialization and ii) the online part, used for crawling.

The goal of the offline part is to reconstruct the sitemap and to select optimal traversal paths. It uses two modules to create clusters of pages with identical repetitive regions and identical URL formats using algorithms presented in [4] and [5]. After filtering pages with similar structure and URL, the offline subsystem estimates the informativeness of each cluster using an intelligent module that reclaims structural elements and the semantic diversity of the pages in a cluster. The last module used for the offline training, creates lookup tables that describe optimal traversal paths between the clusters of the reconstructed sitemap, using an algorithm that discovers the paths with the minimal cost in each iteration.

D. Focused Crawling in Web Databases

Given the dynamic nature of the Web and the growing volume of hidden information in online databases, it is important to develop focused crawlers able to discover searchable forms that serve as entry points to Web databases. The Form-Focused Crawler (FFC) was a prior attempt to address this issue. FFC uses 3 classifiers trained offline to discover searchable forms: i) a page classifier that classifies a page as being on-topic or off-topic using a topic taxonomy from DMOZ (Open-Directory Project), ii) a link classifier that classifies the links extracted from an on-topic page using the Context Graphs technique discussed in subsection B in order to discover paths with “delayed benefit” and iii) a Searchable Form Classifier (SFC) that filters out non-searchable forms using the pages' structural features.

The Adaptive Crawler for Hidden-Web Entries (ACHE) [6] is an extension of FFC. It is another example of crawling architecture with reinforcement learning. The ACHE embeds the Hierarchical Form Identification (HFI) [7] strategy that combines a SFC and a Domain Specific Form Classifier (DSFC) and composes them in ensemble. The DSFC extracts textual features of forms to identify which ones belong to a specific database domain. Thus, the form filtering process retrieves only searchable forms that belong to the target database domain. In addition, ACHE uses an adaptive link learner. This module is invoked periodically to update the link classifier using textual elements extracted from links that lead to searchable forms.

E. Crawlers Focused on Geographic Locations

WWW is an immense source of data related to geographic locations. Local search exhibits special characteristics, as pages related to a certain location are not as strongly coherent as pages of classic topics. This is why designing an efficient geo-aware focused crawler is very challenging.

The architecture of a geospatial Web search engine, designed for harvesting pages related to urban areas is presented in [8]. The main parts of the search engine are the focused crawler, the indexer and the search module provided to the user as search interface. The crawler uses a content classifier, called geoparser to identify pages related to geographic areas. The geoparser extracts geographic information from the text of the Web pages and provides a constant feedback about the geographic background of every retrieved page to the focused crawler. A separate module called geocoder maps every identified location-related page with a specific geographic coordinate and provides that information as additional data to the indexer. Furthermore, the crawler uses a technique for dynamically expanding the search radius around each relevant page, in order to overcome difficulties aroused from the lack of strong cohesion between pages related to a specific location.

A collaborative geographic focused crawler is developed and studied in [9]. The distributed crawling architectures include a set of crawling nodes each one responsible for a specific portion of the Web. In the proposed system, each node is associated with a specific geographic location. Assigning all URLs, extracted from a retrieved page P, to the crawling node that has the highest probability to be associated with P, proved to be the most efficient crawling strategy. In addition, a variety of collaboration policies used for the exchange of URLs between the nodes was studied.
Some policies are based on textual features such as the occurrences of the name of a location in the i) URL, ii) Extended Anchor Text and iii) full content of a page, while others use non-textual based techniques such as Hash functions, Naive Bayes classifiers and IP address of the Web servers for identifying the location related to a retrieved page. Experiments showed that URL and Extended Anchor Text based policies exhibit the best overall performance.

F. A Self-Learning Focused Crawler with no Offline Training.

The design of an intelligent focused crawler that does not use any prior knowledge or pre-trained classifiers is presented in [10]. The proposed system uses an algorithm that adapts the focus of the crawling process dynamically and estimates the probability of a candidate page to satisfy a specific predicate without downloading it. The crawler is trained online using data collected during the crawl. A variety of elements is used to update the crawler's knowledge: i) the text of pages linking to the candidate pages, ii) terms extracted from the candidate URLs, iii) the number of pages linking to candidate pages and satisfy the predicate and iv) the number of candidate pages' siblings that satisfy the predicate. Experimental results verified that different predicates are best “served” by different learning factors or different combinations of learning factors.

III. COMPARISON OF FOCUSED CRAWLING STRATEGIES

A. Prioritizing Techniques

As a focused crawler always downloads just a fraction of Web pages, it is highly desirable that the downloaded fraction contains the most relevant pages. This requires a metric of importance for prioritizing Web pages’ links. The crawling architectures we studied, mainly use 2 prioritizing techniques. First, some systems use a static configuration, usually created during the initialization phase of the crawling session. On the other hand, most systems use an algorithm that estimates a score of importance for every extracted URL in order to evaluate the relevance of the page, pointed by the URL, to the target topic. The estimated score defines the priority of the links in the crawl frontier. In each iteration of the algorithm, the crawler downloads the page pointed by the URL with the highest score. Typical examples of such architectures are the FFC-ACHE systems, the Baseline-Apprentice crawler and the self-learning focused crawler presented in [10]. The Context Focused Crawler creates a model of the context within which topically relevant pages are found. It uses a predefined understanding of the topical structure, called Context Graphs, which enables the crawler to predict the distance of a retrieved page from a target document and not to exclude irrelevant pages that lead to relevant ones from the crawl. The Context Graphs technique is also used by ACHE’s link classifier. Another strategy that gives credit to paths with delayed benefit, is the one used by the geospatial search engine that performs urban web crawling. In contrast with the Context Graphs technique, which uses a static representation of the crawling environment, the geo-aware focused crawler implements an algorithm that dynamically expands the radius of permitted hops around relevant pages and narrows the crawling scope around the irrelevant ones.

B. Learning Strategies

In order to keep the crawling process going, a focused crawler uses a set of knowledge data, acquired with specific learning techniques. Having reviewed several crawling architectures, we concluded that there are two basic learning techniques adopted by focused crawlers: i) The offline learning and ii) the reinforcement learning.

Focused crawlers that use the offline learning technique, build the modules needed to direct the online focused crawling during a separate initialization phase. Some systems train their learners using prior knowledge acquired from topic taxonomies of online catalogues (e.g., DMOZ), while others create a background of knowledge, collecting training data for their learners from various sources (subsection III.D).

C. Evaluation – Classification Tools

The most important modules of a focused crawling architecture are those acting as classifiers because they are the key elements that affect the quality of the crawl. Classifiers estimate the degree of relativeness of every retrieved Web page or extracted link to the topic of interest.

A common probabilistic model used for building classifiers is the Naïve Bayes (NB) model. NB classifiers are preferred because of their reasonable performance, high speed and rapid configuration. They also obtain better results with a much lower number of features than linear methods such as Support Vector Machines (SVMs) and have low rate errors in general. Systems that use Naive Bayes classifiers are shown in Table I.
### TABLE I. CLASSIFICATION TECHNIQUES

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Naive Bayes</th>
<th>Decision Tree</th>
<th>SVM</th>
<th>Custom Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline-Apprentice</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self-Learning Focused Crawler</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>FFC-ACHE</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Context Focused Crawler</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geospatial Search Engine</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Collaborative Geographic Crawler</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

Except for architectures that use typical learning techniques such as NB, Decision Trees, Multilayer Perceptrons and SVMs, there are also others that build their learners using custom probabilistic models such as the self-learning focused crawler presented in [10]. This system uses a combination of the interest ratios of its learning factors, which is a metric that signifies whether a given set of events make it more likely for a candidate page to be classified as relevant to the user-defined predicate.

#### D. Training Features

There is a wide variety of features that can be used to train the focused crawlers’ learning mechanism. Most such features are extracted from the Web pages’ text but there are also several non-textual elements that provide valuable data to the crawlers’ learners. Tables II and III show the non-textual and textual features used as a training data set by the systems we reviewed in section II.

### TABLE II. NON-TEXTUAL FEATURES

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>iRobot</td>
<td>Structural elements, number of duplicate pages</td>
</tr>
<tr>
<td>Baseline-Apprentice</td>
<td>Class probabilities calculated by the Baseline NB classifier</td>
</tr>
<tr>
<td>FFC-ACHE</td>
<td>Structural elements</td>
</tr>
<tr>
<td>Collaborative Geographic Crawler</td>
<td>IP addresses, Hash functions</td>
</tr>
<tr>
<td>Self-Learning Focused Crawler</td>
<td>Linkage locality, Sibling locality</td>
</tr>
</tbody>
</table>

IV. DISCUSSION

In this section we assess the effectiveness of the systems we studied and propose extensions to specific focused crawling architectures.

As shown in [1] and [6], the crawl under the guidance of online trained learners exhibits significant reduction in loss rate. Furthermore, the systems that adopt reinforcement learning can be configured to perform the crawl using only the learners trained online. Experiments verified that even in that case, the harvest rate is comparable to the harvest rate of typical crawlers trained exclusively offline. What could be proposed as an expansion of the architectures that use only learners, which are trained offline, such as the Context Focused Crawlers and iRobot, is the inclusion of modules that periodically update those learners (e.g., content and link classifiers) with features acquired from successful crawling paths. Such an amendment could help a focused crawler become more focused to the target topic. The experimental evaluation of the collaborative geographic crawler discussed in section II, showed that a parallelization technique increases the download rate and reduces the overall duration of the crawling session significantly. Based on the above observation, we could propose the implementation of a parallelization policy for focused crawlers with distinct crawling tasks. For example, an addition that could speed up the crawling procedure, carried out by the Context Graphs architectures, is the assignment of the pages of each layer of the graph to separate crawling nodes operating in parallel and coordinated by a master node.
A practical addition that ameliorates the quality of a focused crawl is a module that keeps track of the downloaded pages and prevents the download of duplicates. A focused crawler could store the URLs of the retrieved pages in a local repository or could perform some type of URL normalization in order to avoid crawling the same resource more than once. The use of the above technique is imperative for distributed focused crawlers, such as the proposed collaborative geographic crawler (see subsection E), to prevent the retrieval of the same Web pages from more than one crawling node.

Missing from the proposed systems we studied is a re-visit policy that updates the knowledge of the focused crawlers’ learners by re-crawling specific pages with high topical coverage. A technique that creates patterns of pages with large coverage and little redundancy is investigated in [11] and is best suited for the initialization phase of focused crawlers. Discovering the appropriate pages for selective re-crawling is an essential point. This is due to the fact that in one hand the rate of change on the web pages stands in high levels. Broder et al. in [12] estimated that rate of change of the Web is 7% per week, while nearly one year later, Ntoulas et al. when revisiting the same issue found out that every week, 8% of downloaded pages were new [13]. On the other hand, another critical point for the selective re-crawling is the degree of change on the Web. In a four year study described in [14], Koehler addressed the problems of defining the lifecycle of the web objects and their changes, finding out that the average half-life of a web page is approximately two years. Five years later, in a similar study that lasted for a significantly shorter time (100 days), Kim and Lee monitored approximately two to three million URLs in two-day time intervals, in order to analyze changes in the web pages [15]. They ended up with the results that that the average half-life of a web page gets shorter and that on a weekly basis, about 5% of URLs detected were new.

Finally, empirical results showed that the selection of the initial pages of the crawl affects the quality of the crawling process considerably. In most of the systems studied in section II, the seed pages are randomly selected by sampling a topic taxonomy or by performing breadth-first crawling from a set of seeds retrieved from a topic taxonomy. A promising extension would be the integration of an algorithm that optimizes the selection of seed pages by retrieving a pre-defined number of seeds with the highest PageRank or by selecting the appropriate seeds that maximize the potential and the value of relatively of the portion of the Web graph, covered by them, to the target topic as in [16].

After thoroughly examining the proceedings of the WWW conferences of the past decade, we present the graph of related papers in the area of focused crawling published in these conferences in Figure 1. Each node represents a paper connected with a link if it is cross-referenced by another node of the network. Vertices with high in-degree (i.e. papers with a large number of citations) are larger in magnitude than others with low in-degree. The community “Other” includes papers with topics of research beyond the scope of this paper such as focused crawling metrics and algorithms designed for optimizing the effectiveness of searching mechanisms. The isolated nodes of the graph mostly represent posters of the most recent years and this is why they do not have any citations.
REFERENCES