An Agent-Based Focused Crawling Framework for Topic- and Genre-Related Web Document Discovery

Nikolaos Pappas*
Idiap Research Institute
Rue Marconi 19
Martigny 1920, Switzerland
nikolaos.pappas@idiap.ch

Georgios Katsimpras
University of the Aegean
Department of Information and
Communication Systems Engineering
Karlovassi 83200, Greece
gkatsimpras@gmail.com

Efstathios Stamatatos
University of the Aegean
Department of Information and
Communication Systems Engineering
Karlovassi 83200, Greece
stamatatos@aegean.gr

***Abstract*—The discovery of web documents about certain topics is an important task for web-based applications including web document retrieval, opinion mining and knowledge extraction. In this paper, we propose an agent-based focused crawling framework able to retrieve topic- and genre-related web documents. Starting from a simple topic query, a set of focused crawler agents explore in parallel topic-specific web paths using dynamic seed URLs that belong to certain web genres and are collected from web search engines. The agents make use of an internal mechanism that weighs topic and genre relevance scores of unvisited web pages. They are able to adapt to the properties of a given topic by modifying their internal knowledge during search, handle ambiguous queries, ignore irrelevant pages with respect to the topic and retrieve collaboratively topic-relevant web pages. We performed an experimental study to evaluate the behavior of the agents for a variety of topic queries demonstrating the benefits and the capabilities of our framework.**

***Index Terms*—web document discovery, focused crawling, genre-aware crawling, utility-based agents, link analysis**

I. Introduction

The retrieval of relevant documents to a given topic is a problem that needs to be addressed nowadays taking into account the vast amount of web sources. Web search engines are able to retrieve a set of relevant documents based on a specific query. However, a number of top URLs returned as a result may contain irrelevant documents regarding the topic and some of the documents that a user is seeking are possibly hidden inside the domains of these URLs. A common thing one can do is to redefine their query in order to retrieve more specific results that they are interested in, or to follow the links starting from some of the top URLs returned.

Trying to solve the above problem automatically, there is a number of possible solutions. For example, a system that collects all the URLs from a general web search engine for a given topic and performs Breadth First Search (BFS) until it retrieves additional related documents. This method is costly because the search space increases exponentially in each of the level of the search depth, and cannot scale easily. Instead of performing a BFS a system could retrieve additional documents by using the APIs of the major web search engines and by searching on the already indexed documents per domain. However, using APIs involves some shortcomings such as the usage and query limitations, but most importantly they rely completely on third-party entities.

A more efficient solution can be obtained by using focused crawling techniques i.e. targeting the search from the seed URLs to pages containing a given topic. Since, a single thread performing this task might be inefficient, it is common to apply distributed methods to achieve more efficiency. However, the scoring between the threads concerns a single relevant path in the Web. Thus, the retrieved pages rely mostly in the initial seed URLs which will guide the crawler towards the most relevant sub space of the Web given that prior information. Exploiting different relevant paths from subsets of the initial seed URLs for a given topic has not been examined thoroughly in the literature. Moreover, to our knowledge there is no related study that adapts dynamically to the topic, handles topic ambiguities and balances between topic and genre scores.

In our effort, we tackle the problem of web document retrieval in an automated and real-time manner using a cost-effective approach. The proposed framework consists of utility-based agents that start from a set of initial seed URLs and perform focused crawling in order to retrieve relevant documents for a given topic. The agents are able to adapt to the topic by using an internal weighting mechanism which combines topic and genre relevance scores of unvisited web pages. Each agent has a priority mechanism that defines the next crawling step during search and ignores low scored web pages based on time tokens. In addition, the agents are able to handle ambiguous queries using a simple approach based on a specification of irrelevant context keywords. The proposed framework can be used for a variety of topics, web genres and applications such as web document retrieval, opinion mining and knowledge extraction.

The contribution of the present paper is three-fold. Firstly, we present an agent-based focused crawling framework that achieves great parallelism and can be applied to a variety of web-oriented applications. Secondly, we propose a weighting mechanism for handling together topic and genre relevance scores for the retrieval of web documents and thirdly we introduce an approach for dynamic adaptation to the topic that

*Research performed partly while at University of the Aegean and partly while at Idiap Research Institute.

is based on the Topic Specific Weight Table described in [5].

The remainder of this paper is organized as follows. In Section II we present the related work on the field of focused crawling and web document retrieval. In Section III we describe the architecture of the proposed framework and the mechanisms used by the utility-based agents. Finally, in Sections IV, V we provide our experimental study and in Section VI we conclude the paper by highlighting some future work directions.

II. Related Work

Numerous techniques for determining the importance of unvisited URLs with respect to a query exist in the literature. Many of them exploit textual information, link structure or even both. Fish Search [7] and Shark Search [11] were some of the earliest algorithms for crawling pages based on a query. The Fish Search algorithm is query driven; starting from a set of seed pages and it considers only those pages that have content matching a given query and their neighborhoods. Shark Search is a modification of Fish Search which differs in two ways: a child inherits a discounted value of the score of its parent, and this score is combined with a value based on the anchor text that occurs around the link in the web page.

A subsequent approach was introduced in [3]. They presented a naive Bayes classifier to calculate a topic-relevance score for a fetched page and then used this as a ranking score of the unfetched child pages. In [14], the authors proposed a naive Best-First crawler which calculates the cosine similarity of a crawled page to a topic or a query in order to evaluate the gain of following the links discovered on that page. In [1] an intelligent crawler able to learn its way to the topic was proposed. This method combines page content, URL string information, sibling pages and statistics about relevant pages in order to assign a priority value to candidate pages. An alternative method was presented in [10]. The authors used a complex knowledge representation method to semantically associate the documents. All terms that are conceptually similar to the terms of the topic are retrieved from an ontology. A similar ontology-based focused crawler but serving in the domain of transport services is discussed in [9].

Diligenti et al. [8] introduced the concept of context graph which is a learning-based approach that uses a mixture of both content measures and backward crawling. Another learning-based approach, which uses a genetic algorithm to compute the ranking score for a page, taking into account text and link characteristics of the fetched pages, is presented in Johnson et al. [13]. A focused crawler approach that uses reinforcement learning where the crawler is trained off-line using a collection of pre-fetched documents and hyperlinks, is proposed in Rennie et al. [17].

Regarding to link analysis, some of the recent examples are [15], [4] which exploit information extracted from link analysis based on URL score, anchor score and relevance score. Other approaches are based on page rank value. Examples of this approach are presented in [20], based on To-page rank value, and in [19], based on T page rank. Others are based on meta search and content block partition, as in [18]. Jamali et al. [12] used a combination of link structure of the fetched pages and the content similarity of a document to a certain domain. This mixture of link structure and content is used to compute a ranking score for the candidate unfetched pages. Other approaches include a rule based focused crawler [2].

Another interesting approach that benefits from the best answers returned by a Web search engine is discussed in [16]. Specific information from a set of documents is used to query a Web search engine. The top n results returned from the search engine are used to train a classifier that is used to guide the crawlers. A more recent approach by Hati et al. [5] (2010) calculates the url score based on its anchor text relevancy, its description from Google relevancy, the cohesive text similarity with topic keywords and the relevancy of its parent pages. The relevancy score is calculated on the vector space model. In [2] the authors proposed an architecture using multiple agents for cooperative information gathering. Assis et al. (2009) [6] proposed a genre-based crawling approach. It exploits not only content-related information but also genre information present in pages to guide the crawling process.

In our effort, we integrated link (similar to [5]) and genre (similar to [6]) analysis in a unified agent framework. Our contribution to the link analysis method described in [5] is the topic disambiguation, the online TSWT calculation, the dynamic expansion of TSWT during search, independency from Google's description and expiration of low prioritized URLs based on time tokens (see Section III).

III. The Proposed Framework

The architecture of the proposed framework is depicted in Figure 1. The initial coordination of the agents is performed by the Manager, which is based on a set of starting parameters (number of agents, number of seed URLs, number of concurrent requests, categories of web pages i.e. genres, query), it collects the seed URLs from the Seeding Module and distributes them to a set of focused crawler agents. The seed URLs are collected dynamically from the major web search engines (Google, Bing, Yahoo) and belong to specific genres (Web, News, Discussions, Blogs)

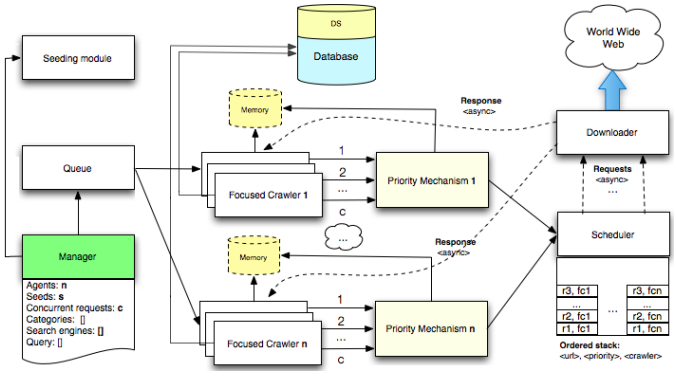

Fig. 1. The architecture of the proposed framework.

A Focused Crawler Agent (FCA) in our framework could

be defined as a utility-based agent i.e. an autonomous entity which observes the environment (the visited pages and the unknown URLs it discovers) and directs its activity (prioritizes URLs to increase the possibility to find and be directed towards relevant URLs), in order to achieve a specific goal (retrieve only relevant URLs for a specific topic during its activity period[1]). It also uses existing knowledge (TF-IDF in vector space, relationship between pages and other metrics) and previous experience in order to achieve its goal (see Section 3-A1, 3-A2). The internal memory of the agent can be accessed by an internal Priority Mechanism (see Fig 1) that is responsible for deciding on the next action of the agent.

The FCAs upon initialization are registered to a Directory Service (DS) in the database where they store the relevant URLs they collect during their search. In addition, using their Priority Mechanism they send requests to the Scheduler which uses an ordered stack for each agent and it is responsible for filtering already visited pages and forwarding asynchronous download requests to the Downloader. The Downloader requests the pages from the Web and returns the responses asynchronously to the target agent's page processing function.

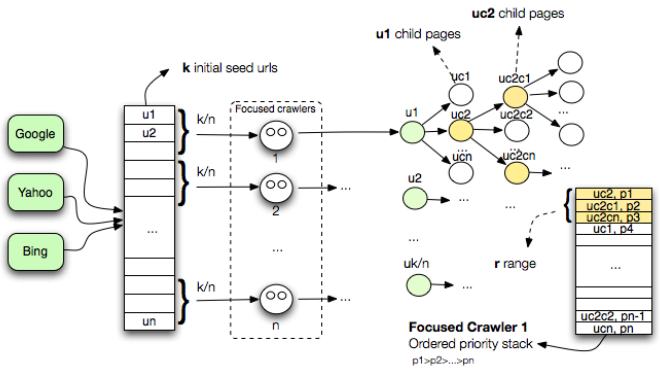

Fig. 2. The FCAs following different crawling paths based on the initial seed URLS. A usage example of internal priority stack during search is also displayed for the FCA 1.

Figure 2 demonstrates the different paths that the agents follow and their internal priority stack. In the beginning, the FCA requests to download the initial seed URLs that are received upon initialization. For each URL response, it extracts the child URLs inside the page and uses its Internal Weight Mechanism in order to weigh the unvisited URLs and assign them a priority. When the priority is assigned, the URL is inserted in its internal ordered priority stack. The next step is selected by the top r priority URLs using the Priority Mechanism. Therefore, each FCA can handle r concurrent requests accelerating the discovery of web pages. The top r URLs are formed as a request packet and are sent to the Scheduler. The responses come asynchronously (see Fig. 1) and the agent continues prioritizing and picking the next top r URLs to follow.

[1]The activity period ends when the FCA reaches the maximum number of relevant URLs (predefined goal) or when it has no further move to make.

At the processing stage of a web page, a simple topic disambiguation is applied for queries which have ambiguous meaning; all the URL responses that contain irrelevant context keywords regarding the topic are ignored as well as their child URLs. The disambiguation is based on a predefined specification of irrelevant contexts. Despite the simplicity of this method, the irrelevant pages are reduced significantly and the system functionality is satisfied.

A. Internal Weight Mechanism

The Internal Weight Mechanism combines topic and genre relevance scores of unvisited web pages which are calculated by using TF-IDF metrics, relations between URLs and information about each of the crawled pages from the previous experience of a FCA. Below, we describe the metrics used and the final calculation of the link score.

1) TF-IDF Measure in Vector Space Model: Starting with a set of d documents and a set of t terms; the vector-space model treats each document as a vector v in the t-dimensional space R_t. Let the term frequency be the number of occurrences of term t_i in the document d_j i.e $TF_{i,j}$. The weighted term frequency $TFW(d_j, t_i)$ measures the association of a term t_i with respect to the given document d_j:

$$TFW(d_j, t_i) = \left\{ \begin{array}{ll} 0 & if \;\; TF_{i,j} = 0 \\ 1 + \log\left(1 + \log\left(TF_{i,j}\right)\right) & otherwise \end{array} \right\} \quad (1)$$

Finally, TFW from Formula 1 and inverse document frequency (IDF) are combined together, which forms the TF-IDF measure of term t_i in a specific document d_j.

$$\textit{TF-IDF}(d_j, t_i) = TFW_{d_j,t_i} * IDF_i \quad (2)$$

The overall weight w_{t_i} of a term t_i in the collection of documents d is calculated by the formula:

$$w_{t_i} = \sum_{i=1}^{j} (\textit{TF-IDF}(d_j, t_i)) = \sum_{i=1}^{j} (TFW_{d_j,t_i} * IDF_i) \quad (3)$$

and the normalized weight W_{t_i} is calculated by the formula:

$$W_{t_i} = \frac{w_{t_i}}{w_{max}} \quad (4)$$

where w_{max} is the max weight of a term in the set of terms t. The FCA keeps in its internal knowledge the weights W_{t_i} for each term t_i that it finds during its search. The collection of documents d along with the set of terms t are dynamically increasing so that the weights for each term are updated when a term t_i appears in a new document d_j. A similar update is performed for the IDF_i when a term appears, with respect to the collection of documents.

2) Topic Specific Weight Table (TSWT) Construction: The internal knowledge of the FCA that represents its experience provides enough information for the creation of a Topic Specific Weight Table (TSWT) for a number of seed URLs. The agent takes as input the initial seed URLs, visits them and stores the TF-IDF metrics in the vector-space model. When the number of the initial seed URLs is reached then the creation of the TSWT is performed using the Internal Weight Mechanism.

All the terms that have been addressed during the search are sorted based on the normalized weight W_{t_i} and the top k words represent the TSWT. An example of a TSWT with size 10 from an execution with query *Audi* is listed in the following table. The 10 top results of the Web category of Google was used as seed URLs and contributed to the forming of the TSWT table. We can observe that the FCA adapted to the context of the query *Audi* (Table I).

Keyword	Normalized Weight
audi	1.0
hwi	0.94
engine	0.70
car	0.68
volkswagen	0.53
autoblog	0.53
race	0.45
quattro	0.30
union	0.28
german	0.22

TABLE I
AN EXAMPLE OF TSWT FOR THE QUERY *Audi*.

In our effort, the TSWT calculation is performed online i.e. during the search procedure. Moreover, we discriminate two categories of TSWT, namely static and dynamic.

The static category refers to a TSWT with a fixed size (i.e. consisting of a fixed number of keywords). On one hand, if one uses a very small size of TSWT, the representation of the topic will be very poor and on the other hand, if one uses a very big size of TSWT the representation of the topic will be more abstract (since there will be many keywords contributing and it will probably overlap with other topics). Our experiments on the variation of TSWT, showed that the value of 10 (used in [5]) is not optimal.

The dynamic category refers to a TSWT that is dynamically expanded during the search. The dynamic expansion of TSWT is achieved by the addition to the table of a portion of top weighted keywords of the most relevant pages (based on a threshold ranging from 0 to 1) met during the search. Intuitevely, the highly relevant pages to a topic that are found during search contribute incrementally to the representation of the topic, thus an even better adaptation to the topic can be achieved comparing to a static TSWT (see Section V).

3) Relevance Calculation: Let A be the document vector (containing the weights of each term) and B the vector which represents the weights in the TSWT or Genre. The relevance calculation is performed using the cosine similarity measure. In the case of Genre the vectors are set by default to the value of 1. Given two vectors of attributes, A and B, we denote:

$$Similarity(A, B) = cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} \quad (5)$$

The cosine similarity of vectors A and B will range from 0 to 1, since the term frequencies cannot be negative.

4) Link Score Based on TSWT: When a FCA processes a document it extracts all the URLs contained in it. The Link Score assigns scores to the unvisited URLs, using existing internal knowledge and the metadata of hyperlinks. The unvisited URLs are modeled as an abstract unvisited page along with the anchor metadata. For each of the child pages p the Link Score based on TSWT is calculated as follows:

$$\begin{aligned} LS_T(p) = LinkScore_T(p) = & \; AnchorRelevanceScore(p) \\ & + \; UrlRelevanceScore(p) \\ & + \; CohesiveTextRelevanceScore(p) \\ & + \; \sum_{i=1}^{n}(ParentRelevanceScore(P_i)) \end{aligned} \quad (6)$$

where p is a child page and P_i from 0 to n are the parent pages of p. The $AnchorRelevanceScore$ expresses the similarity between keywords in the TSWT (topic keywords) and anchor text. It is calculated by the following formula:

$$AnchorRelevanceScore(p) = \frac{\sum_{i \in (A \cap T)} (W_i)}{s} \quad (7)$$

where A is the set of keywords in the anchor metadata, W_i the weight of the keyword $i \in (A \cap T)$, T is the topic keywords and s the size of the TSWT. This measure is a normalized percentage of the words appearing in the anchor metadata with the keywords in the TSWT. The $UrlRelevanceScore$ expresses the similarity between keywords in the TSWT (topic keywords) and URL keywords, and is calculated as follows:

$$UrlRelevanceScore(p) = \frac{\sum_{i \in (U \cap T)} (W_i)}{s} \quad (8)$$

Same as before, U is the set of keywords in the URL, W_i the weight of the keyword $i \in (U \cap T)$, the T is the topic keywords and s the size of the TSWT. The $CohesiveTextRelevanceScore$ is the score of an unvisited page with respect to topics in the sentence its URL appears. For the cohesive-text, one sentence or group of meaningful sentences around the anchor link is extracted. The $CohesiveTextRelevanceScore$ expresses the similarity between the keywords in the cohesive-text and the TSWT (topic keywords) and is calculated as before:

$$CohesiveTextRelevanceScore(p) = \frac{\sum_{i \in (C \cap T)} (W_i)}{s} \quad (9)$$

where C is the set of keywords of the cohesive-text, W_i the weight of the keyword $i \in (C \cap T)$, the T is the topic keywords and s the size of the TSWT.

Finally the $ParentRelevanceScore$ is the relevance of a parent P_i of the page p with the TSWT, calculated with cosine similarity of the vector A of the parent page with the vector B of the TSWT.

$$ParentRelevanceScore(P_i) = Similarity(A_{P_i}, B_{TSWT}) \quad (10)$$

The vectors containing the weights W_i in the R_t space of the parent page and TSWT accordingly. This score expresses the relevance of the parent page with the TSWT.

5) Link Score Based on Genre: It is calculated similarly to the Link Score based on TSWT, but in this case the vector B consists of a predefined set of genre keyword specifications of News, Discussions and Blogs. Since no weights have been calculated for the genre keywords the values for the vector is set to the default value of 1. The formula for the calculation of this score is similar to Equation 9: $LS_G(p) = LinkScore_G(p)$. When there are multiple genres defined then the above link score is calculated for each of them and the maximum score from all these genres is chosen to participate to the final link score.

6) Final Link Score Based on Combined Weights: The final score is composed of the combination of the above mentioned scores with simple summing and with weighted mean summing. The simple summing is calculated by the following formula if we set each of the weights to 1. The weighted mean summing uses the weights w_T and w_G as open parameters that sum up to 1. The formula of the Final Link Score ($FinalLS$) for a given page p is provided below:

$$FinalLS(p) = w_T * LS_T(p) + w_G * LSG(p) \quad (11)$$

where w_T is the weight of the Link Score based on TSWT and the w_G the weight of the Link Score based on Genre. Weight variations for Final Link Score create different Internal Weight Mechanisms for the FCAs.

B. Priority Mechanism

The Priority Mechanism is activated each time the FCA crawls a web page. Initially, it orders the priority stack of the FCA and then, using a predefined range r, picks the top r prioritized URLs, forms a request and yields them to the Scheduler with the defined priority accordingly. Before the rooting to the Scheduler the URL is loaded to the locked priority Stack (internal structure of the FCA) where every URL that hasn't been crawled yet is stored. This stack is used because there is a possibility that the agent meets the same URL before it has been crawled (due to the concurrency of the system). Therefore, before the agent adds or updates a URL in its priority stack, it checks first whether this link is locked or not. The procedure described above can be summarized in the following algorithm.

PRIORITIZE ALGORITHM:

```
RemoveExpired(PriorityStack)
OrderedPriorityStack = sort(PriorityStack)
TopRUrls = OrderedPriorityStack[:r]
for Url, Priority in TopRUrls do
   Lock(Url)
   del PriorityStack[Url]
   yield Request(Url, Priority)
end for
```

When a URL is entered to the priority stack by the FCA apart from the priority, a timestamp is assigned to this URL. This timestamp defines a limited time for URL to remain in the priority stack. The Priority Mechanism is responsible for the deletion of the entries of the priority stack that are very old and still remaining to the stack (usually low prioritized URLs). With this method the space of the memory is managed more efficiently, the FCA obtains the ability to *forget*, it avoids visiting low priority URLs in the future and the removed URLs do not contribute to the weighting scheme.

DELEXPIRED ALGORITHM:

```
for Url, TimeStamp in PriorityStack do
   if HasExpired(TimeStamp) then
      del PriorityStack[Url]
   end if
   yield Request(url, priority)
end for
```

IV. EXPERIMENTAL SETUP

Taking into account that our framework is intended to be applied for web document retrieval on the Web, we need an online evaluation methodology that will be able to satisfy this purpose. Moreover, it relies on web search engines to collect the seed URLs for the focused crawling process, thus it was not feasible to evaluate it offline using a fixed dataset. In this section, we describe our evaluation methodology and the evaluation metrics that were used in our experimental study.

A. Evaluation Methodology

In order to evaluate the proposed framework, we performed experiments that produced web page collections for specific queries. A crucial parameter is the number of the collected pages. This is an application-dependent decision since for the same query different users may demand the discovery of a few or a lot web pages. In this study, we used a predefined threshold of pages.

Formally, given a query q, a set of general web search engines S and a threshold t of crawled web pages, we initialize our framework and the agents follow their topic-relevant web paths using the initial seed URLs collected from S. When the number of crawled web pages becomes equal to t the focused crawling procedure is then terminated.

B. Evaluation Metrics

Regarding the evaluation metrics, we selected the well-known precision metric which is commonly used in the information retrieval domain. The precision estimates the fraction of the retrieved web documents that are relevant and is calculated by the following formula:

$$Precision(q) = \frac{|Relevant_q \cap Retrieved_q|}{|Retrieved_q|} \quad (12)$$

where $Relevant_q$ are the documents that contain the query q and $Retrieved_q$ are the documents that were retrieved for the query q. The above metric is used to evaluate the performance of the system for a given execution.

Concerning the measurement of the execution time we simply calculate the time passed from the starting moment of an execution until the agents reach the predefined threshold t of crawled pages.

V. Experimental Results

In this section we evaluate the various aspects of our agent-based focused crawling framework in a realistic web setting. Starting from the most general aspects we demonstrate the efficiency of the system in execution time and the additional exploration of the system compared to a general web search engine. We continue with more specific aspects such as disambiguating the topic, the agent's behavior on topic adaptation based on TSWT and finally we evaluate the weighting mechanism for genre-based crawling.

In the following experiments, we used the evaluation methodology (see Section IV) for ten subsequent executions and we report the average values of the performance metrics.

A. Concurrent Processing of Requests

Each agent in our framework uses concurrent processing of the requests chosen as next step for the focused crawling procedure. We demonstrate the efficiency of the system regarding the concurrency aspect using a single FCA. For this experiment we used 50 seed URLs from Google for two topic queries (*Audi*, *Jaguar*) and a predefined threshold t of 150 crawled pages. We varied the number of the concurrent requests from 1 to 12. The averages in precision and execution time are displayed in Fig. 3.

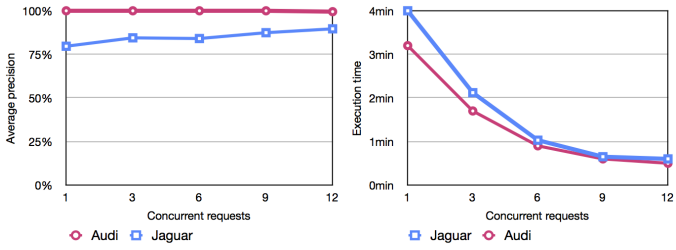

Fig. 3. On the left, concurrency versus precision is displayed and on the right, the average execution time for the queries *Audi* and *Jaguar*. The seed URLs were collected from all genres.

On the left, the average precision remains steady and even increases for query *Jaguar* as the concurrency is increased. On the right, the execution time of the system, as it was expected, is greatly improved when using concurrent requests greater than one. The execution time decreases dramatically as the number of concurrent requests increases. As shown in Fig. 3, the average execution time for 1 request is about 3-4 minutes and it decreases to almost 30 seconds when the concurrent requests become 12.

B. Exploring Beyond the Web Search Engine's Results

Usually when searching for a topic in a general web search engine, irrelevant pages may appear in the results. In this case, one could manually search inside the results to find more interesting pages or redefine the keywords for their query. The purpose of focused crawling addresses the above problems. Intuitively, the seed URLs will be explored in depth and information that does not appear in the results will be discovered by the FCAs.

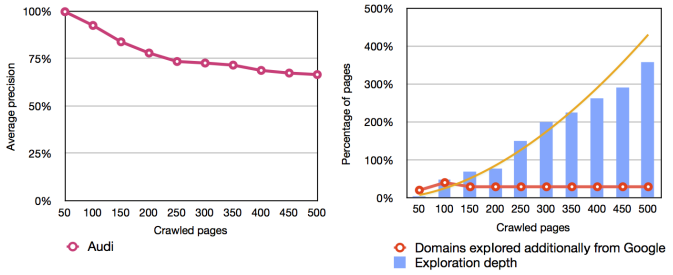

Fig. 4. On the left, the average precision is displayed and on the right, the additional exploration on common domains with Google for the query *Audi*. The seed URLs were collected from News genre.

In order to confirm this assumption we executed the following experiment. For 10 steps during a run of a predefined threshold t of 500 crawled web pages, we compared the pages discovered by the FCAs with the top 500 pages from Google results. In the Fig. 4 we can observe the additional exploration made by the agent for the query *Jaguar* taking seed URLs from Google News genre. The left diagram displays the average precision of the agent and the right one displays the additional exploration made compared to Google. About 30% percentage of the common domains with Google results, are further explored in depth by the FCAs, so there is a chance of discovering topic-relevant pages, not displayed for the top c Google results.

C. Topic Disambiguation

When searching for pages that match an ambiguous topic, it is very likely some seed URLs or some pages crawled so far to be irrelevant with the topic. In this case, the focused crawling procedure may be guided to a irrelevant subspace of the Web regarding the topic and the performance results may considerably decrease. To overcome this problem, FCAs disambiguate the meaning of the topic. Despite the simplicity of the method, the irrelevant pages decrease dramatically when disambiguation is applied. In Fig. 5 on the left, we observe the effect on precision and on the right the increase of irrelevant pages with and without using disambiguation.

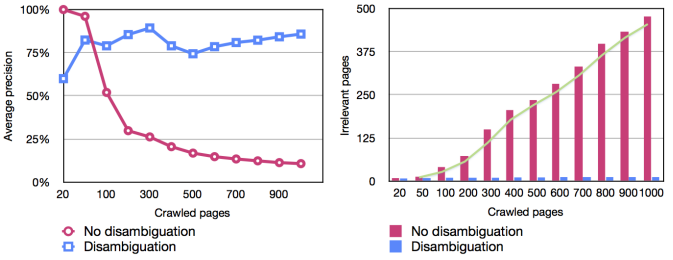

Fig. 5. On the left, the average precision is displayed and on the right, the number of irrelevant pages for both using or not disambiguation of the topic. The seed URLS were collected from all genres for the query *Jaguar*.

As an ambiguous query we used *Jaguar* (car) in a frame of 1000 crawled pages with irrelevant context keywords {cat, animal, jungle, cheetah, tiger, leopard, football, hat, watches, soccer}. Without using disambiguation many of the pages

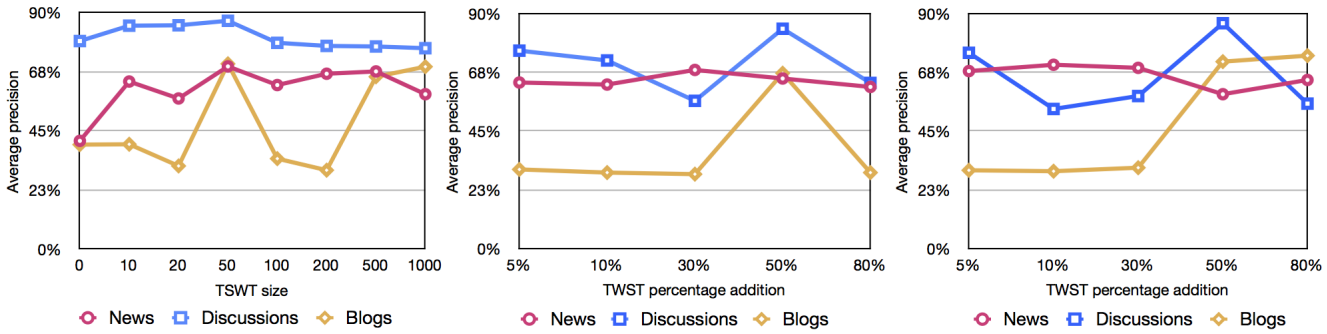

Fig. 6. On the left, various values for static TSWT are displayed against the system performance, on the middle, the dynamic addition of keywords to TSWT with 0.6 similarity threshold and on the right, with 0.8 threshold for three different genres of web pages (News, Discussions and Blogs) and the query *Jaguar*.

crawled belonged to other contexts (cheetahs, soccer team, etc.). When using disambiguation we achieved crawling pages that mostly belong to our desired context (car).

D. Using Multiple Focused Crawler Agents

Apart from performing focused crawling with a single agent, many agents can be used in order to explore together a set of seed URLs . This means that an additional exploration in each segment of the seed URLs will be made. Doing so, we reduce the search space and each agent focuses on a topic-relevant path which initiates from a segment of seed URLs.

In Fig. 7 on the left, the performance of three different FCAs is displayed (FC1, FC2, FC3) for the query *Audi*. The seed URLs were collected from three different categories: Web, News and Discussions and were distributed equally to the agents in order to perform a search. On the right, we display the average precision of the three agents compared to the average precision achieved by a single FCA taking as seed URLs the sum of the seed URLs of the other three agents.

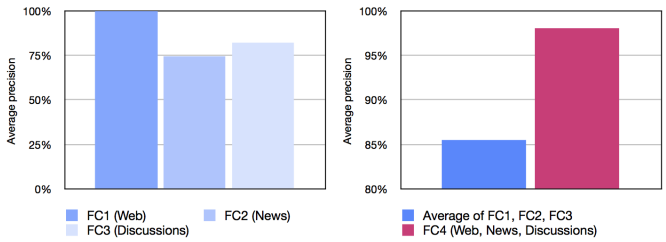

Fig. 7. On the left, the average precision is displayed for three agents (FC1, FC2, FC3) using seed URLs from Web, News and Discussions genres and on the right, the average performance of those three agents versus a fourth one (FC4) that uses the sum of the seed URLs of the former for the query *Audi*.

The precision of the FC4 agent is higher than the other three agents, mostly because FC4 had larger prior information (seed URLs) and it was guided toward the path of the most relevant URLs. On the other hand the diversity of the explored URLs by FC1, FC2, and FC3 was greater than FC4. In addition, the system achieves great parallelism and it is more computationally efficient by using more than one agents (similarly to the experiment described in Section V-A).

E. Topic Specific Weight Table (TSWT)

The FCAs at the their initial stage try to learn an amount of relevant keywords represented by the TSWT. In the Internal Weight Mechanism the dynamic increase to the TSWT during the search of an agent is supported. The most relevant pages that were visited by the agent contribute their most highly weighted keywords to the TSWT based on a minimum relevance threshold T_r.

We compared the performance of static versus dynamic TSWT in various web genres of seed URLs for the query *Jaguar* (Figure 6). For the static TSWT case, we varied the size of the TSWT from 0 to 1000 using a predefined threshold t of 400 crawled pages. For the dynamic TSWT case, in each experiment, the size of TSWT was set to 100 terms while we varied the relevance threshold T in range $\{0.6, 0.8\}$ and the percentage of terms added to the TSWT from 0% to 80%.

In Fig. 6, we can observe that there is a clear improvement in the precision when using a static TSWT of 50 size for all the web genres. For the Blogs categories though, the precision is affected strongly when varying the size of the TSWT. This sensitivity that appears to the variation of TSWT size is possibly caused by the noise (diverse content) that exists in this specific genre. For the dynamic TSWT with an initial size of 100 terms, an improvement to the precision can be observed for 50% TSWT percentage addition to the 0.6 and 0.8 relevance threshold.

F. Genre-based Crawling

Apart from the weighting of URLs based on TSWT, we examined the weighting of URLs based on the Genre of the web page and also based on the combination of them. The goal is to obtain topic-relevant documents that belong to one or more genres. The genre of a web page can be Blogs, Discussions or News and is defined by a keyword specification representative of each genre. In the following experiment, we study the effect of genre crawling to the performance of the FCAs on the Web (Figure 8). For the experiments a predefined threshold t of 500 crawled pages was used.

In Fig. 8, we display the average precision of the guided search based on individual genres (News, Discussions, Blogs), multiple genres using combined weights with TSWT ($0.5 *$

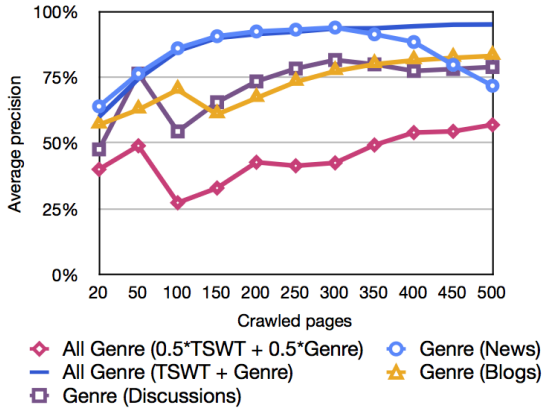

Fig. 8. Performance comparison of weighting strategies for genre crawling on Web. Individual and multiple genres are considered for the query *Jaguar*.

$TSWT + 0.5 * Genre$) and multiple genres with simple score summing (TSWT + Genre). The seed URLs for the individual genres were collected accordingly from the genres provided by the general web search engine (News, Discussions, Blogs) and for the other were collected from all genres (Web).

We observe that the best performance in precision is achieved by using simple summing (TSWT + Genre) and that the performance is decreased when searching for individual genres. The multiple genre search with combined weights $(0.5 * TSWT + 0.5 * Genre)$ appears to have the lowest precision. This does not imply that the results retrieved by the FCA are not satisfying. The loss in precision has to do with the search path of the agent and the selection of its next step i.e. the agent takes into account topic and genre information to prioritize URLs for his next move. Thus, the results have less topic-relevant documents, which nevertheless belong to the genres that the agent is searching for.

VI. Conclusions and Future Work

We presented a focused crawling framework composed of utility-based agents that achieves great parallelism and can be applied to a variety of web-oriented applications such as web document retrieval, opinion mining and knowledge extraction. We proposed a weighting mechanism for handling together topic and genre relevance scores of unvisited web pages for the retrieval of web documents. Using genre-based crawling is particularly useful when there is a requirement for discovering user-generated content (user comments, reviews and discussions). News, Discussions and Blogs are page genres that usually contain such content which can be exploited for further processing. Finally, we introduced an approach for dynamic adaptation to the topic that is based on the Topic Specific Weight Table (TSWT) and is able to enhance the performance of the system.

As future work, we are planning to use this framework for discovering opinions about various topics in the task of opinion mining. In addition, we would like to experiment with more sophisticated methods for topic disambiguation and the exchange of information between the agents.

VII. Acknowledgements

Our research work was supported by the European Union through the inEvent project FP7-ICT n. 287872 (see http://www.inevent-project.eu). The authors thank Andrei Popescu-Belis and Thomas Meyer for their helpful remarks.

References

[1] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu. Intelligent crawling on the world wide web with arbitrary predicates. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, New York, NY, USA, 2001. ACM.

[2] J. Akilandeswari and N. Gopalan. Design of an enhanced rule based focused crawler. *First International Conference on Emerging Trends in Engineering and Technology, 2008. ICETET '08.*, 2008.

[3] S. Chakrabarti. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16), 1999.

[4] X. Chen and X. Zhang. Hawk: A focused crawler with content and link analysis. *e-Business Engineering, 2008. ICEBE '08*, 2008.

[5] A. K. D Hati, B Sahoo. Adaptive focused crawling based on link analysis. *Education Technology and Computer ICETC 2010 2nd International Conference on (2010)*, 4, 2010.

[6] G. T. de Assis, A. H. F. Laender, M. A. Gonçalves, and A. S. da Silva. A Genre-Aware Approach to Focused Crawling. *World Wide Web*, 2009.

[7] P. M. E. De Bra and R. D. J. Post. Information retrieval in the world-wide web: making client-based searching feasible. *Comput. Netw. ISDN Syst.*, 27(2), 1994.

[8] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[9] H. Dong, F. K. Hussain, and E. Chang. A transport service ontology-based focused crawler. *Proceedings of the 2008 Fourth International Conference on Semantics, Knowledge and Grid*, 2008.

[10] M. Ehrig and A. Maedche. Ontology-focused crawling of web documents. In *Proceedings of the 2003 ACM symposium on Applied computing*, SAC '03, New York, NY, USA, 2003. ACM.

[11] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The shark-search algorithm. an application: tailored web site mapping. *Computer Networks and ISDN Systems*, 30(17), 1998.

[12] M. Jamali, H. Sayyadi, B. B. Hariri, and H. Abolhassani. A method for focused crawling using combination of link structure and content similarity. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '06, pages 753–756, Washington, DC, USA, 2006. IEEE Computer Society.

[13] J. Johnson, K. Tsioutsiouliklis, and C. L. Giles. Evolving strategies for focused web crawling. In T. Fawcett and N. Mishra, editors, *ICML*. AAAI Press, 2003.

[14] F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz. Evaluating topic-driven web crawlers. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, New York, NY, USA, 2001. ACM.

[15] A. Pal, D. S. Tomar, and S. C. Shrivastava. Effective Focused Crawling Based on Content and Link Structure Analysis. *Journal of Computer Science*, 2(1), 2009.

[16] G. Pant, K. Tsioutsiouliklis, J. Johnson, and C. L. Giles. Panorama: extending digital libraries with topical crawlers. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '04, New York, NY, USA, 2004. ACM.

[17] J. Rennie and A. McCallum. Using reinforcement learning to spider the web efficiently. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[18] Y. Sun, P. Jin, and L. Yue. A framework of a hybrid focused web crawler. *Future Generation Communication and Networking Symposia, 2008. FGCNS '08. Second International Conference on*, 2, 2008.

[19] F. Yuan, C. Yin, and J. Liu. Improvement of pagerank for focused crawler. *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. SNPD 2007.*, 2, 2007.

[20] Y. Zhang, C. Yin, and F. Yuan. An application of improved pagerank in focused crawler. *Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, 2, 2007.