ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Θεματικά Εστιασμένη Προσκομιδή Ιστοσελίδων από τον Κρυμμένο Παγκόσμιο Ιστό

Παναγιώτης Ν. Λιάκος

Επιβλέπων: **Αλέξιος Δελής**, Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2011

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Θεματικά Εστιασμένη Προσκομιδή Ιστοσελίδων από τον Κρυμμένο Παγκόσμιο Ιστό

Παναγιώτης Ν. Λιάκος

ΑΜ: M990

ΕΠΙΒΛΕΠΩΝ :

Αλέξιος Δελής, Καθηγητής ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Αλέξιος Δελής, Καθηγητής ΕΚΠΑ
Μανόλης Κουμπαράκης , Καθηγητής ΕΚΠΑ

Περίληψη

Ένα συνεχώς αυξανόμενο πλήθος υψηλής ποιότητας πληροφορίας είναι αποθηκευμένο σε σελίδες που έρχονται από τον Κρυμμένο Παγκόσμιο Ιστό (Hidden Web). Τέτοιες σελίδες είναι προσβάσιμες μόνο μέσω μιας διεπαφής επερωτήσεων που παρέχεται από τους Κρυμμένους Ιστότοπους και μπορεί να εκτείνονται σε μία ποικιλία θεμάτων.

Προκειμένου να παρασχεθεί κεντρικοποιημένη πρόσβαση στον Κρυμμένο Παγκόσμιο Ιστό, οι προηγούμενες προσπάθειες είχαν επικεντρωθεί στην ανάπτυξη τεχνικών παραγωγής επερωτήσεων που στοχεύουν στην λήψη ενός ολόκληρου Κρυμμένου ιστότοπου με το μικρότερο κόστος. Σε ορισμένες περιπτώσεις όμως, μας ενδιαφέρει η λήψη μόνο ενός συγκεκριμένου κομματιού ενός τέτοιου ιστότοπου. Για παράδειγμα, σε μία ειδησεογραφική βάση δεδομένων, έναν χρήστης μπορεί να ενδιαφέρεται για την ανάκτηση μόνο των αθλητικών άρθρων και όχι των πολιτικών. Σε αυτή την περίπτωση, πρέπει να κάνουμε την καλύτερη δυνατή χρήση των πόρων μας κατεβάζοντας μόνο το τμήμα του Κρυμμένου ιστότοπου που μας ενδιαφέρει.

Στην εργασία αυτή, ερευνούμε το πως μπορούμε να αναπτύξουμε μια θεματικά εστιασμένη εφαρμογή προσκομιδής κρυμμένων ιστοσελιδών (Hidden Web Crawler) που μπορεί αυτόνομα να εξάγει θεματικές σελίδες από τον Κρυμμένο Παγκόσμιο Ιστό, αναζητώντας μόνο στο υποσύνολο που είναι σχετικό με την αντίστοιχη κατηγορία. Για το σκοπό αυτό, παρουσιάζουμε τεχνικές παραγωγής επερωτήσων που λαμβάνουν υπόψη τους το θέμα το οποίο μας ενδιαφέρει. Προτείνουμε έναν πλήθος από διαφορετικές πολιτικές συγκομιδής ιστοσελίδων και τις αξιολογούμε πειραματικά με δεδομένα απο ένα δημοφιλή ιστότοπο.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανάκτηση Πληροφορίας

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Κρυμμένος Παγκόσμιος Ιστός, εφαρμογή προσκομιδής ιστοσελίδων, θεματικά εστιασμένη, επιλογή επερωτήσεων, αξιολόγηση

Abstract

A constantly growing amount of high-quality information is stored in pages coming from the Hidden Web. Such pages are accessible only through a query interface that a Hidden-Web site provides and may span a variety of topics.

In order to provide centralized access to the Hidden Web, previous works have focused on developing query generation techniques that aim at downloading all of a given Hidden Web site with the minimum cost. In certain settings however, we are interested in downloading only a specific part of such a site. For example, in a news database, a user may be interested in retrieving only sports articles but no politics. In this case, we need to make the best use of our resources in downloading only the portion of the Hidden Web site that we are interested in.

In this thesis, we study how we can build a topically-focused Hidden Web crawler that can autonomously extract topic-specific pages from the Hidden Web by searching only the subset that is related to the corresponding category. To this end, we present query generation techniques that take into account the topic that we are interested in. We propose a number of different crawling policies and we experimentally evaluate them with data from a popular site.

SUBJECT AREA: Information Retrieval

KEYWORDS: Hidden Web, crawler, topic-sensitive, query selection, evaluation

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου Αλέξη Δελή για την αδιάκοπη στήριξη και τη σταθερή καθοδήγηση που μου παρείχε καθόλη τη διάρκεια εκπόνησης της διπλωματικής εργασίας μου.

Επίσης, θα ήθελα να ευχαριστήσω τον κ. Αλέξανδρο Ντούλα για τη συνεχή επικοινωνία και βοήθεια τόσο κατά την αναζήτηση του θέματος όσο και κατά την υλοποίησή της παρούσας εργασίας.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An ever-increasing amount of high-quality information on the Web today is accessible through Web pages pulling information from data sources such as databases or content management systems. Such pages are collectively called the Hidden Web because they are hidden from the common users but are typically only accessible after issuing one or more queries to a search interface.

In most cases, the information existent in the Hidden Web is of high quality as it has been carefully reviewed, edited or annotated before being stored in a database or a content management system and may span a multitude of topics ranging from sports and politics to different medical treatments of a particular disease.

In order to facilitate the discovery of information on the Web, search engines and content-aggregation systems could greatly benefit from approaches that would allow them to collect and download the Hidden Web pages. Having information from the Hidden Web all in one place can be of great benefit to both users (as they can have a one-stop shop for their information needs) and for the search engines (as they can serve their users better).

Since a Hidden Web site cannot be accessed by traditional search engine crawlers but only through a search interface, previous work has investigated ways of collecting the information from the Hidden Web sites. In most cases [13, 14, 4, 5, 9, 16, 17] previous work has focused on generating queries that are able to download *all* of (or as much as possible) a given Hidden Web site, with the minimum amount of resources spent (e.g. queries issued). For example, the technique in [13] iteratively issues queries to Hidden Web sites and can download about 90% of some sites using about 100 queries.

Although such approaches can work well for the cases where we are interested in doing a comprehensive crawl of a Hidden Web site, there are cases where we may be interested only in a specific portion of a Hidden Web site. For example, a search engine that specilizes in traveling may be interested in picking only news articles that pertain to traveling from a general-purpose news database. A portal talking about politics may want to identify the politics-related articles from a blogs database and leave out the sports-related ones. Or, a mobile application focusing on night-life in San Francisco may want to pull only the related

articles from all the postings on events in the wider Northern California.

In this thesis, we study the problem of building a topic-sensitive Hidden Web crawler, that can automatically retrieve pages relevant to a particular topic from a given Hidden Web site. One way to achieve this goal would be to employ previously-developed techniques (e.g. [13, 14, 4]) to retrieve the majority of the Hidden Web site and then only keep the content that we are interested in. Since this approach may lead to downloading a number of pages that we are ultimately not interested in, it may also lead to a great waste in resources, measured in time, money, bandwidth or even battery life in the case of a mobile setting.

To this end, the goal of our crawler is to retrieve from a Hidden Web site as many pages related to a given topic as possible with the minimum amount of resources. Our main idea is to proceed in an iterative fashion issuing queries to the Hidden Web site that are very relevant to the topic that we are interested in.

In summary, this thesis makes the following contributions:

- We formalize the problem of tpically-sensitive Hidden Web crawling, i.e. downloading the pages from a Hidden Web site that pertain only to a given topic.
- We present an algorithm for performing topically-sensitive Hidden Web crawling. Our idea is to identify candidate keywords from the crawled documents that are relevant to the topic of interest.
- We propose a number of different policies that can be used to decide which of the candidate queries we can issue next. As will show later in our experimental section some polices are much better in producing good keywords than others.
- We evaluate our algorithm using the different policies on a real *Hidden Web* site and we showcase the merits of each of the polcies.

Chapter 2

Topic-Sensitive Hidden Web Crawling

In this section, we initially describe the structure of the Hidden Web together with its idiocracies and the challenges that arise while trying to crawl it. Then, we present a Topic-Specific Hidden Web crawling algorithm that effectively addresses the issue of downloading documents from a collection, by using queries relevant to the topic in search, and thus, avoiding the waste of resources. Later, we give the details of the structure our algorithm uses, as a pool of words and discuss how we can extract appropriate terms that are used as queries. Finally, we present the different policies proposed in this work, for the evaluation of the returned documents.

2.1 Hidden Web Structure

A typical topic-sensitive (or focused) Web crawler that aims at retrieving pages from the surface Web, has to evaluate the contents of each downloaded page to decide whether it is relevant to a particular topic. The goal of the crawler is to examine as small of a subset of the total search space as possible in order to avoid wasting resources by downloading pages that are irrelevant to the topic.

Figure 2.1 illustrates this process. Pages that are relevant with our search are represented with black boxes while the irrelevant ones are represented with white ones. The goal of the crawler is to follow only links with relevant pages, in order to use its resources efficiently.

In order to determine whether a given page is relevant to the topic, the crawler typically employes a classifier that can assess how close the given page is to the topic at hand. This process overall, aims at estimating the likelihood that a given link may lead to a promising Web page (i.e. of the desired topic) before actually following the link.

Although this has been shown to work well in practice for the surface web [7], it is not directly applicable in the case of a Hidden Web site, as these pages cannot be retrieved by following links. A query interface is usually the unique “entry point” Hidden Web sites

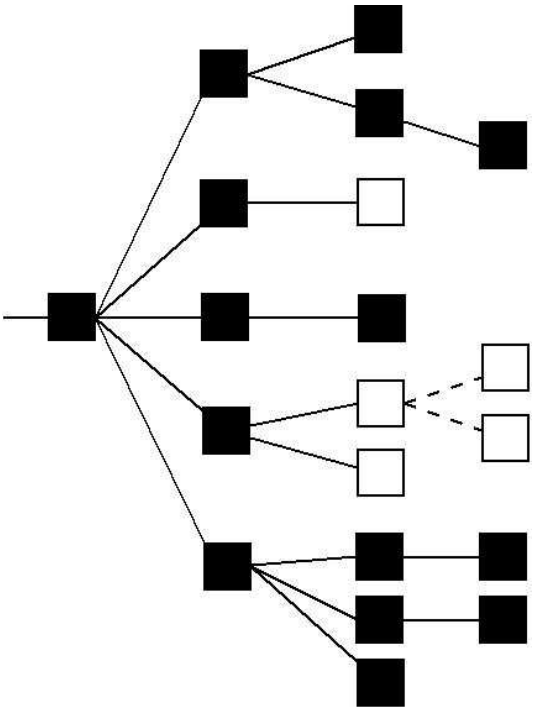

Figure 2.1: Topic-Sensitive Crawling in the publicly indexable Web

provide and thus, access to their contents is made possible only by issuing queries through this interface. Figure 2.1 illustrates an example of such an interface. This example is from the Open Directory Project Web site (hereafter also referred to as *dmoz*).

Figure 2.2: An example of a Hidden Web Site's search interface

At a high level, in order to access the pages from a Hidden Web site we typically need to apply the following steps:

1. First of all the users submit a query through the interface provided by the Web Site, that characterizes what they want to locate.

   An example is presented in Figure 2.1, where a user is ready to search for documents relevant with the term 'Godfather'.

2. Then the users receive a result index page, than contains links and possibly additional information of the pages that matched the submitted query.

   Figure 2.1 shows an example result index page. We can see that the pages are usually ordered according to how relevant they are to the submitted query.

3. Finally, the users, having identified an interesting page in the results, can click on its link and eventually visit the actual Web page, such as the one illustrated in Figure 2.1.

Search: **Godfather**

Open Directory Sites (1-20 of 94)

1. The Godfather - A fan features a review, pictures, and ratings. Other films also presented.
   -- *http://zeetosee.tripod.com/volumeII.html#godfather* *Arts: Movies: Titles: G: Godfather Trilogy: Godfather, The: Reviews (7)*

2. FilmSite: The Godfather - Detailed synopsis by Tim Dirks.
   -- *http://www.filmsite.org/godf.html* *Arts: Movies: Titles: G: Godfather Trilogy: Godfather, The: Reviews (7)*

3. Allwatchers Review: The Godfather - Review and viewer comments.
   -- *http://www.allwatchers.com/Topics/Info_4367.asp* *Arts: Movies: Titles: G: Godfather Trilogy: Godfather, The: Reviews (7)*

4. GoneMovie.com: The Godfather III - Synopsis, trivia, and multimedia.
   -- *http://www.gonemovies.com/WWW/TopFilms/Godfather/EnglischGodfather3.asp* *Arts: Movies: Titles: G: Godfather Trilogy: Godfather, Part 3, The (4)*

5. GoneMovie: The Godfather - Features pictures, plot summary, sound clips, and facts about the film.
   -- *http://www.gonemovies.com/WWW/TopFilms/Godfather/EnglischGodfather.asp* *Arts: Movies: Titles: G: Godfather Trilogy: Godfather, The (3)*

Figure 2.3: List of matching pages for query 'Godfather'

Filmsite Movie Review

The Godfather (1972)

TOP ★ 100
View all

Pages: (1) (2) (3)

Background

● The superb, three-part gangster saga was inaugurated with this film from Italian-American director Francis Ford Coppola, **The Godfather** (1972). The first two parts of the lush and grand saga are among the most celebrated, landmark films of all time. Many film reviewers consider the second part equal or superior to the original, although the first part was a tremendous critical and commercial success - and the highest grossing film of its time. This mythic, tragic film contributed to a resurgence in the American film industry, after a decade of competition from cinema abroad.

One of the original "Movie Brats" who had not had a hit after seven films, director Coppola collaborated on the epic film's screenplay with Mario Puzo who had written a best-selling novel of the same name about a Mafia dynasty (the Corleones). *The Godfather* catapulted Francis Ford Coppola to directorial superstardom, and popularized the following euphemistic phrase (of brutal coercion): "I'm gonna make him an offer he can't refuse."

The almost three hour, R-rated saga film (for violence and graphic language) won three Oscars: Best Picture, Best Actor (Marlon Brando refused to accept the award) and Best Adapted Screenplay (Mario Puzo and Francis Ford Coppola). The other seven nominations included three for Best Supporting Actor (James Caan, Robert Duvall, and Al Pacino), Best Director, Best Sound, Best Film Editing, and Best Costume Design. One of *The Godfather's* original eleven nominations was removed, Best Music (Original Dramatic Score), when it was determined that Nino Rota's score had been used for a previous film.

Figure 2.4: The second matching page for query 'Godfather'

2.2 A Topic-Sensitive Hidden Web Crawling Approach

As we discussed in the previous section, access to the contents of a Hidden Web site is achieved by issuing queries through a search interface and obtaining a result index page. Therefore, a Hidden Web crawler should automatically issue queries to the site and then retrieve the pages included in the results.

One of the biggest challenges in implementing such a crawler is for the crawler to pick the most appropriate terms that will retrieve the pages pertaining to the desired topic in the most effective way. If the crawler can pick terms that are very well-suited to the topic we are interested in, the crawler will be able to also retrieve pages that are of the given topic. On the other hand, if the crawler issues terms that are completely irrelevant, it will simply waste its resources by downloading pages that are of no interest to the users (and potentially also degrading the quality of the search engine that employs the crawler).

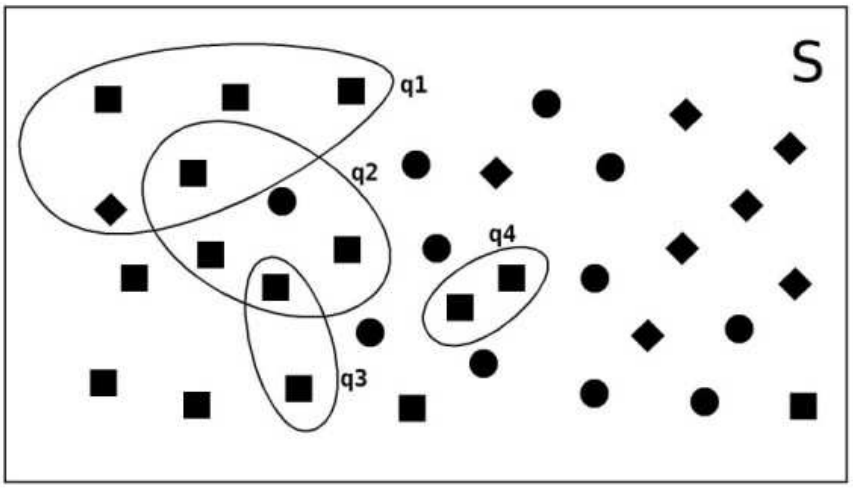

Figure 2.5: Morphing a Hidden Web site as a set

To illustrate, we assume that the crawler downloads pages from a Hidden Web site that contains a set of pages S (the rectangle in Figure 5). These pages might cover a vast area of topics, most of which may potentially be irrelevant with our search. Every possible submitted query q_i can be considered as a subset of S that contains the pages that the site would return as a result.

In practice, we are interested only in pages relevant to a specific topic (i.e. the squares in Figure 2.2). To this end, in order to minimize the total cost of crawling, we need to discover

queries that focus on the topical area of our interest, and in that way avoid the extra cost of issuing queries that return irrelevant results.

In order to select the best keywords, at each step of the algorithm we examine the candidate keywords that we have retrieved from the collection so far and we pick one to be issued as our next query. We describe both the keyword extraction and selection processes in sections 2.3 and 2.4 respectively. In order to bootstrap our algorithm, we consider an initial description of the topic as a set of terms. This initial description of the topic can either be one or more terms that are relevant to the topic, or a document relevant to the topic. After the first step, and after we have downloaded each batch of results based on the term that we queried, we augment our list of candidate terms to include terms that were newly found. This however, implies that we need to be able to determine whether the downloaded page pertains to the desired topic or not. For this step, we also employ a classifier as we will discuss in more detail in Section 2.5.

```
(1) while (available resources) do
      /* Extract the terms */
(2)   if(cnt++ mod N == 0)
        T(di) = ExtractTerms(di);
      /* select a term to send
         to the site */
(3)   qi = SelectTerm(T(di));
      /* send query and acquire
         result index page */
(4)   R(qi) = QueryWebSite(qi);
      /* download and evaluate
         the pages of interest */
(5)   DownloadAndEvaluate(R(qi), pi);
    done
```

Figure 2.6: Topic-Sensitive Crawling of a Hidden Web Site

Figure 2.6 shows the algorithm for a Topic-Sensitive Hidden Web Crawler.

Step (2) of the algorithm is only issued once every N times –to contain the number of requisite computations– and it extracts the best terms of an input document. This is done by calculating the *TF/IDF* weight for every term of the document. The process is explained in detail is Section 2.3.

Step (3) picks the best of the terms that Step (2) extracted from the document, that has not been used so far, while step (4) uses this term to issue a query and retrieves the result index page. Step (4) is further explained in Section 2.4.

Finally, Step (5) downloads the Hidden Web pages that where included in the results and had not been downloaded previously and evaluates the contents of the results using one of the policies presented in this work. The evaluation process of this step is responsible for the maintenance of the collection of words used for keyword extraction. This process depends heavily on the policy that is to be followed. The different policies are explained in Section 2.5.

2.3 Word Collection

In this section we provide the details of the Word Collection that serves as a pool of possible queries for our algorithm.

Document *di* used in Step (2) of the Algorithm 2.1 is the *Word Collection* that defines the topic in search. It consists of text that is related to that topic and is initialized with an exemplary document that describes the topic as input. So, if for instance we wanted to crawl for sports articles from a news site, we could provide the algorithm with an input document (or snippet, or a small set of keywords) that would consist of a few sport-related articles. However, the Word Collection cannot remain static during the execution of the algorithm for a variety of reasons.

First, the input document given to the algorithm during the initialization of the Word Collection, may not be enough for the extraction of all (or enough) terms that are needed for the retrieval of a sufficient amount of Web Pages. No matter how good that initial document may be in capturing the essence of the topic in search, it may manage to provide only a limited number of terms.

Second, the initial Word Collection may be too specific, in a way that the terms extracted would not be general enough to capture the whole topical area of the document. For instance, if those sport-related articles mentioned earlier, were taken from a WNBA fan-site, the terms extracted from the Word Collection would retrieve results concerning women's sports and basketball. We are interested in matching the input document with a broad topic, in order to retrieve all the related Web Pages. Therefore, it is necessary to broaden our Word Collection during the execution of our algorithm.

Finally, to successfully retrieve the maximum amount of Web Pages from a Hidden Web site, it is essential that we adapt to its terminology. For instance, we cannot retrieve but a subset of Web Site that indexes multilingual content if all the words in our Web Collection are English.

Therefore, it is clear that for effective Topic-Sensitive Hidden Web Crawling, the pool of

words that is used for term extraction must be enriched continuously as well as adapt to the site in search. To face this issue, we exploited the contents of the results as potential parts of the Word Collection. Each result page is evaluated using one of the policies described in Section 2.5 and the contents of the ones relevant to the topic in search are added to the Word Collection.

In that way, the Word Collection is able to provide a sufficient amount of terms to be issued as queries. Furthermore, since the Word Collection is updated with content directly from the site in search, it can provide the algorithm with terms that not only are relevant to a specific topic, but have a higher significance for that site.

2.4 Extracting Keywords

In order to effectively retrieve results, the queries picked must be focusing on the topic that we are interested in.

The queries can be as simple as a single term or more complex, e.g. a phrase. Both methods have been examined in the past, and the latter, although is able to capture the essence of a topic finer, has failed to furnish the expected improvements.

Issuing a well-chosen term phrase instead of a single term through a search interface, will likely return extremely relevant results. However, a great number of also relevant results, will be cut-off and additional queries will be required to discover them. Therefore, we chose to use single terms for our queries.

One approach to choosing appropriate single terms, would be the use of "keyword resources". Predefined terms, characteristic for different topical areas. However, this method lacks significantly in adaptability. Another approach, is measuring the *term frequency (tf)* of every word in the result pages and picking the most common one. However, this is not effective when trying to retrieve topic-specific pages. Choosing an initial word carefully will not help either, because it is extremely likely that irrelevant pages will be included in the results nonetheless and will affect the query selection process.

Therefore, a filtering of the results is necessary in order to maintain a collection of words that characterize the topic in search appropriately. The different policies that we propose for this purpose are further discussed in Section 2.5. However, even with the filtered results and by additionally applying a stop-word exclusion mechanism, the queries picked using term frequency factors will still be too general.

To successfully address this matter, a collection depended factor such as the *inverse*

document frequency (idf) must be used; the *idf* factor "varies inversely with the number of documents n to which a term is assigned in a collection of N documents" [15]. Therefore, *idf* measures accurately the general importance of a term. Thus, by applying the composite *TF/IDF* term weighting system we can successfully retrieve the most important words of a document. In our case this document is the Word Collection that we discussed in the previous section.

However, *TF/IDF* measurements are computationally intensive.In order to limit calculations we decided to apply the term extraction process once every N submitted queries. The initial document is rich enough to provide us with an adequate number of terms. Additionally, the collection of words that is used for most of the process is not greatly affected with the results of every submitted query. The terms extracted from it using *TF/IDF* are more or less the same for consecutive queries. Therefore, it would be meaningless to delay the process with unnecessary calculations.

2.5 Result evaluation policies

In this section we provide the details of the various evaluation policies that are presented in this work. This policies are used in order to decide if each page contained in the results is relative to the topic in search, and therefore will be helpful later. The contents of the pages that are considered in-topic are added to the Word Collection, and consequently take part in the keyword selection process.

The policies examined here are:

- ***perfect:*** We use the categorization information directly from the site in search. Of course such information is not available in most cases. In our work, we will use this policy as a benchmark to determine how well the rest of the policies can do relative to this one that has perfect information regarding the topic that every result document belongs to. Hidden Web sites usually categorize their contents using a broad classification taxonomy. For example, all the *Movie* related web sites listed in the *dmoz* site are under the topic: "/Top/Arts/Movies". In this policy, the Word Collection will be supplied only with in-topic content.

- ***do-nothing:*** We accept all of the returned pages as in-topic.

  This policy updates the Word Collection with every page the crawler manages to discover. Since the first few terms are extracted from the input document it is expected

that the first queries that will be submitted will be meaningful and so the corresponding result pages will have a high chance of being in-topic. However, since the results are never filtered it is expected that a lot of out-of-topic pages will find their way to the Word Collection and affect the term selection process significantly.

- ***NaiveBayes:*** We use a Naive Bayes classifier that decides if the result is or is not relative to the topic in search.

  This policy examines the effects the presence of classifier has to the crawling process. The classifier needs to go through a training phase before it can be used. Therefore, it is clear that this method can only be used for topics that the classifier is already trained for. Obviously, the better the classifier is trained the less erroneous pages will be added to the Word Collection.

- ***CosineSimilarity:*** We examine the cosine similarity of every result page with the initial document that defines the topic and accept only a small percentage of the closest ones. This policy is clearly superior to the NaiveBayes policy in terms of adaptability, since it requires no training. However, since this method depends heavily on the input document, it is important that, the effect the quality of the latter has in its decision making, be examined as well.

The first two policies were used to provide us with comparison points and serve as benchmarks for the rest of the policies. The *perfect* policy makes absolutely no mistakes when determining the topic of a document and thus its word collection is always in-topic. On the other hand the *do-nothing* policy adds every single returned page to the Word Collection and thus may consider terms that are not necessarily related to the topic of interest. The *perfect* policy depends on information that cannot be considered available, while, the *NaiveBayes* policy requires the presence of training data and therefore lacks in adaptability.

Chapter 3

Experimental Evaluation

In this section, we initially detail the dataset used in our experiments. Then, we show results of the algorithm using the policies described in section 2.5 for a variety of topics. Firstly, we present the results the four policies had over the Topic *Sports* of *dmoz*. Then, we evaluate the performance the three most interesting policies had over five Topics of *dmoz*. Finally, we test the effect the size of the input document has on the *CosineSimilarity* policy and compare our results with that of a simple *Hidden Web* crawler.

Variable N of the algorithm in Figure 2.6 was set to seven for all of our tests; we have experimentally established for the dataset we worked with that this number (7) offers a good compromise between computations and the extraction of new terms. As far as the *CosineSimilarity* policy is concerned, we kept 1% of the returned documents after the submission of every query for future use in the term-extraction process. We opted for this specific 1% in order to hold approximately the same number of returned documents by competing policies.

3.1 Dataset

The experiments we conducted focused on downloading the Open Directory Project [1] contents. *Dmoz* indexes 3.8 million links that cover a vast area of topics. Each link is accompanied with the site's title, a brief summary and its categorization. The links are searchable through a keyword-search interface and *dmoz* enforces an upper limit on the number of returned results (10,000 links).

We considered the titles and summaries of each indexed link of the *dmoz* website as a document, and provided them to our evaluation policies for possible use in the keyword extraction process.

For every topic we conducted an experiment on, we needed an initial exemplary document that defines it. To build those documents we used the titles and summaries of a small number of random links of the *dmoz* site which belong to the corresponding category.

Furthermore, the presence of a text corpus was necessary, in order to perform the *TF/IDF* measurements and extract the best keywords for each topic. For this purpose we used the

titles and summaries of about 200,000 links in *dmoz*.

3.2 Evaluation of the different policies over the same topic

In order to evaluate the performance of the policies proposed in 2.5 we attempted to retrieve all pages of *dmoz* relevant with *Sports*. There was a total of 90,693 pages categorized under this topic.

The results of our experiments are illustrated in Figure 3.1.

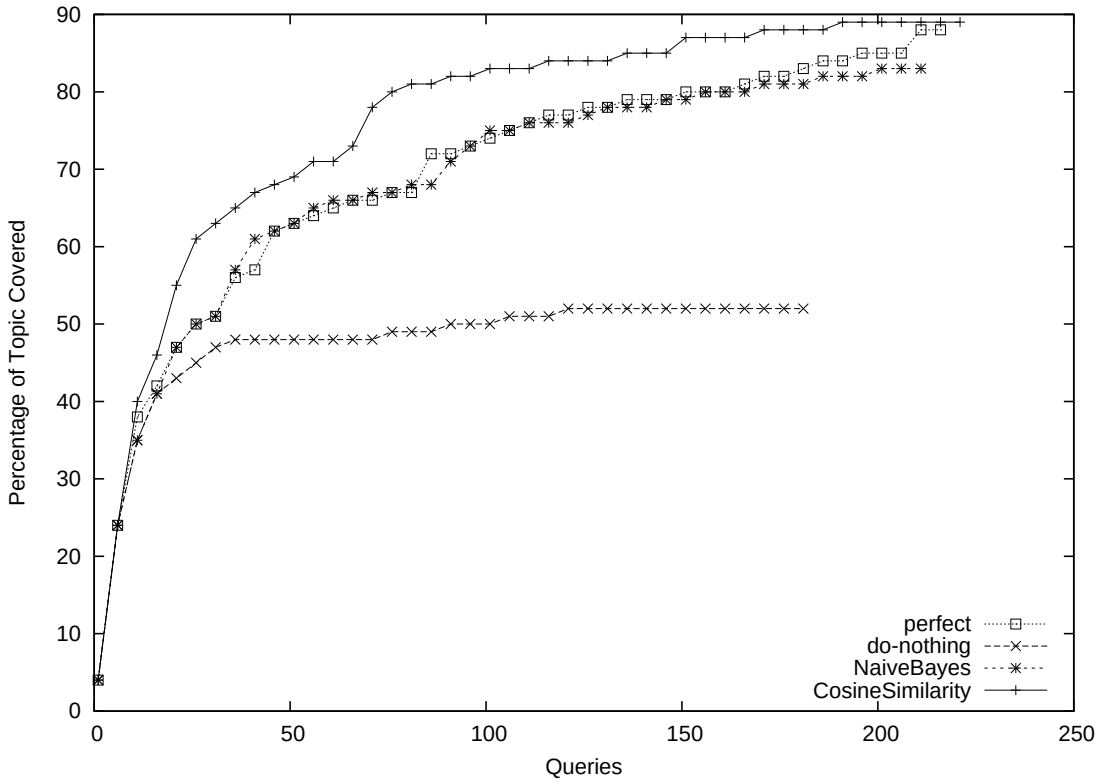

Figure 3.1: Results for topic Sports

We can see that all four policies behaved identically for the first seven queries, where the initial document is being used for keyword extraction. *Do-nothing*'s performance was enormously affected after those first few queries, when the Word Collection was updated with all the returned results, since as the name of the policy implies, no action was taken to filter them. As a consequence, *do-nothing* performed badly, as expected.

Policies *perfect* and *NaiveBayes* behaved similarly with the first one managing to retrieve a slightly bigger percentage of *Sports* related documents. This can be explained from the fact

that the two policies had a high percentage of common queries issued (28%).

After 210 queries, the *perfect* policy retrieved 86.43% of the total relevant documents, while the *NaiveBayes* policy managed to collect 83.42% of them.

The *CosineSimilarity* policy managed to outscore all three of them by retrieving 89.66% of the documents categorized under the Topic *Sports*. 19% of the terms it used for query submission where common with the *perfect* policy and 20% of them where common with the *NaiveBayes* policy.

Tables 3.1 to 3.4 present some sample queries that were issued by each policy. One noticeable remark is that there is a lot of overlapping in the results of each term. Although every policy manages to discover meaningful terms that return a great number of results, a large portion of them has been already discovered by previous queries.

Furthermore, it is safe to assume, that without the use of an upper limit in the number of returned results for every submitted query, most terms would successfully retrieve a significantly larger amount of links in-topic.

Another interesting fact, is that the *do-nothing* policy is the most successful one, in finding "popular" terms. Most of the ones illustrated in Table 2 returned the maximum of 10,000 results, while the average after 210 queries was 8,650. This is due to the fact that the Word Collection of this policy was eventually enriched with terms from every possible topic of the Open Directory Project. The *NaiveBayes* policy was second with 7,927, the *perfect* policy third with 7,530.31 and the *CosineSimilarity* policy last with 7,426 results per query.

3.3 Classification Errors

Using the *dmoz* categorization information for every downloaded document, we were able to measure the error rates of the different evaluation policies.

The *perfect* policy was obviously the most successful one on this matter, since it uses this information and avoid making any misclassifications.

The *do-nothing* policy chooses to accept every document it downloads as relevant to the topic in search, so its errors are equal to the total number of links retrieved minus the links that were actually in-topic. Thus, its error percentage was 95.8%.

The classifier used in the *NaiveBayes* policy on the other hand, proved to be very effective and misclassified only 6.6% of the returned documents.

However, the impact of all errors is not the same for our algorithm. An in-topic document that is classified as irrelevant is not added to the Word Collection and does not affect the term

No	term	results	new links	new links in-topic	total links in-topic
1	results	10,000	9,490	4,065	4,065
2	statistics	10,000	8,054	3,512	7,577
3	roster	10,000	7,712	5,503	13,080
10	men	10,000	7,578	2,066	31,812
15	scores	5,556	1,811	505	37,706
20	players	10,000	5,147	1,576	42,776
25	hockey	6,943	2,693	1,127	45,243
30	tennis	7,353	4,400	327	46,644
40	rugby	4,207	1,254	181	51,806
60	sport	10,000	7,649	292	59,503
100	competition	4,647	1,865	229	67,062

Table 3.1: Queries issued by the *perfect* policy

No	term	results	new links	new links in-topic	total links in-topic
1	results	10,000	9,490	4,065	4,065
2	statistics	10,000	8,054	3,512	7,577
3	roster	10,000	7,712	5,503	13,080
10	schedules	10,000	7,214	744	30,488
15	church	10,000	8,862	0	36,732
20	coaching	10,000	6,678	1,195	40,442
25	methodist	10,000	5,006	0	40,450
30	beliefs	9,091	3,233	327	41,347
40	stellt	10,000	8,775	0	41,370
60	bietet	10,000	4,959	0	41,377
100	nach	10,000	3,091	0	41,403

Table 3.2: Queries issued by the *do-nothing* policy

extraction process. On the other hand, an irrelevant document that "sneaks" its way into the Word Collection, may cause the selection of inappropriate terms for query submission.

For the *NaiveBayes* policy 45% of the documents added to the Word Collection, were actually categorized under another topic in the *dmoz* classification taxonomy.

The *CosineSimilarity* does not classify all the documents returned after the submission of

No	term	results	new links	new links in-topic	total links in-topic
1	results	10,000	9,490	4,065	4,065
2	statistics	10,000	8,054	3,512	7,577
3	roster	10,000	7,712	5,503	13,080
10	schedules	10,000	7,214	744	30,488
15	standings	4,882	1,158	787	37,513
20	baseball	6,375	1,938	742	41,430
25	records	10,000	7,781	652	44,966
30	membership	10,000	7,805	119	46,662
40	county	10,000	8,820	34	52,479
60	fc	4,080	1,963	107	60,033
100	standing	3,139	949	253	67,536

Table 3.3: Queries issued by the *NaiveBayes* policy

No	term	results	new links	new links in-topic	total links in-topic
1	results	10,000	9,490	4,065	4,065
2	statistics	10,000	8,054	3,512	7,577
3	roster	10,000	7,712	5,503	13,080
10	tables	9,258	5,011	281	34,153
15	player	8,262	4,638	1,130	40,167
20	players	10,000	5,437	1,832	46,964
25	hockey	6,943	2,897	1,277	55,734
30	baseball	6,375	1,683	542	57,086
40	race	7,315	3,955	576	60,744
60	conference	10,000	6,246	108	64,950
100	competitive	3,359	999	116	75,262

Table 3.4: Queries issued by the *CosineSimilarity* policy

each query. It orders them according to their relevance with the input document and adds only a small percentage (1% in our tests) of them to the Word Collection.

Most of them proved to be irrelevant to the topic in search (72.6%). However, this did not affect the policy in a negative way, which was not surprising. These documents, despite the fact that they belong to a different topic, had very high cosine similarities with the input

document, so naturally, the Word Collection was not altered in an undesirable way. As a result, the *CosineSimilarity* policy outperformed the other policies.

3.4 Comparison of policies under different topics of dmoz

In this section, we present the performance the policies *perfect*, *NaiveBayes* and *CosineSimilarity* had over five different topical areas belonging to the classification taxonomy *dmoz* uses.

The five topics used were:

- Computers (103,336 documents)
- Recreation (91,931 documents)
- Shopping (87,507 documents)
- Society (218,857 documents)
- Sports (90,639 documents)

As we can see, topic *Society* is of significantly bigger size, which needs to be taken under consideration while viewing its performance in contrast to that of the other topics.

3.4.1 Comparison of different topics using perfect policy

Figure 3.2 illustrates the behavior of our approach for these five different categories of *dmoz* using the *perfect* policy. Topics *computers* and *sports* proved to be the easiest to retrieve while *society* needed a significantly bigger amount of queries to surpass the 80% barrier.

More explicitly, topic *Computers* returned 92.17% of the total documents after 210 queries. Topics *Sports* and *Recreation* discovered 86.43% and 80.76%, respectively, with the same amount of queries. Finally topics *Shopping* and *Society* collected only 77.82% and 62.06%, respectively.

3.4.2 Comparison of different topics using NaiveBayes policy

The results for the same topics using the *NaiveBayes* policy are presented in Figure 3.3. This policy is slightly worse than *perfect* for every one of the five topics. The ordering of the

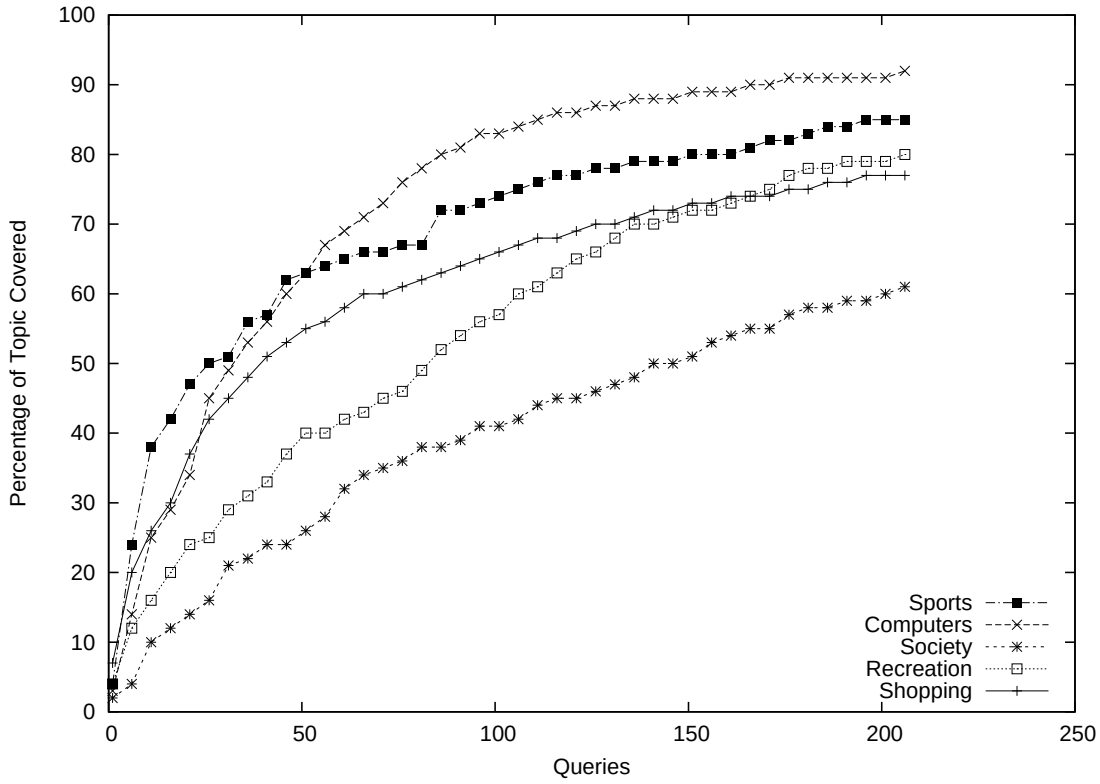

Figure 3.2: Comparison of different topics using *perfect* policy

topics is, however, a little different, since the *NaiveBayes* policy behaved very poorly for the topic *Recreation*, which was ranked fourth below the topic *Shopping*. More explicitly, after 210 queries, topics *Computers* collected 89.81% of the documents, topic *Sports* 83.42%, topic *Shopping* 73.34%, topic *Recreation* 70.86% and topic *Society* 54.86%.

3.4.3 Comparison of different topics using CosineSimilarity policy

Similar results are shown in Figure 3.4 where the *CosineSimilarity* policy is examined. Topics *Computers* and *Sports* were again first with 91.84% and 89.66%, respectively, after issuing 210 queries. Topic *Recreation* collected 83.55% of the related documents, while topics *Shopping* and *Society* returned 79.02% and 62.97%, respectively, after the same amount of queries.

As we can see, the *CosineSimilarity* policy behaved slightly better then the *perfect* policy in 4 of the 5 topics examined, *Computers* being the only exception. Additionally, it out-scored the *NaiveBayes* policy for every one of the five different topics.

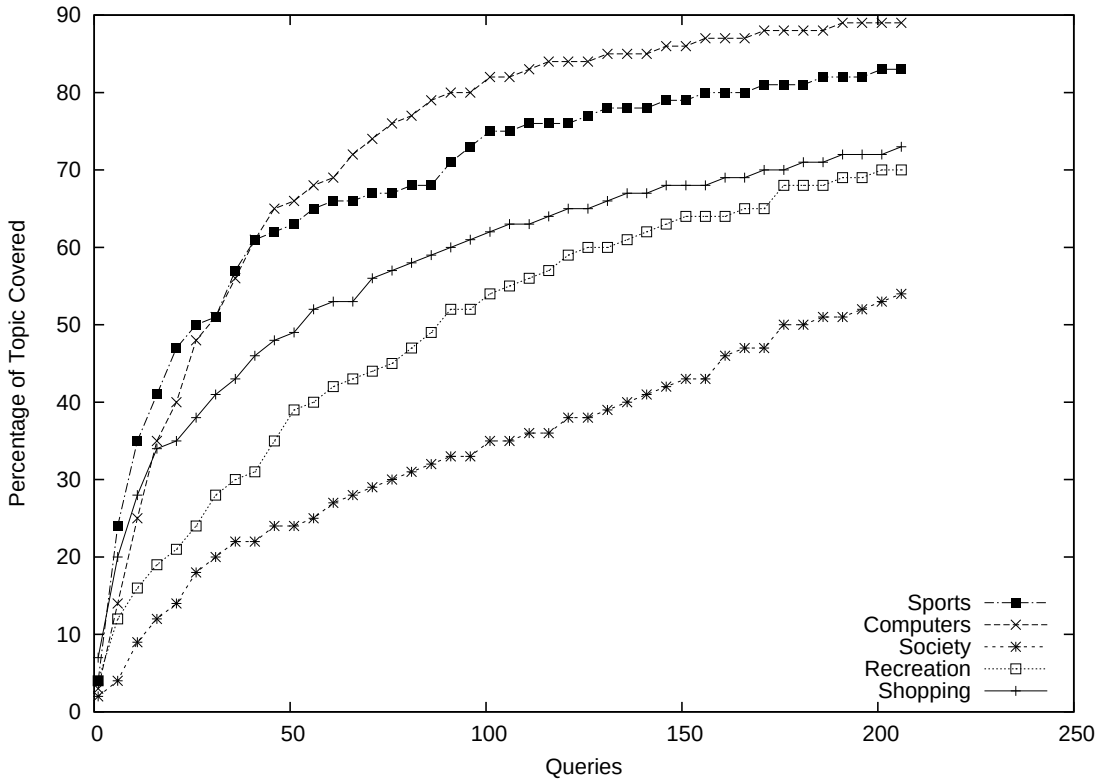

Figure 3.3: Comparison of different topics using *NaiveBayes* policy

Topic *Recreation* behaved better than *Shopping* after the 87th query, as it did with the *perfect* policy after the 162nd query.

3.5 Impact of input document size

The *CosineSimilarity* policy depends heavily on the input document, since it does not use it only to extract the first few queries to be issued, but to evaluate the result documents retrieved as well.

For this reason, it is necessary to examine the impact the size of this document has, on the behavior of this policy.

The other three policies, use the initial document only in those first steps of the process, so its significance on them is negligible.

Figure 3.5 illustrates the results the *CosineSimilarity* policy had, with three different input documents of variable size, while trying to retrieve documents of *dmoz* that belong to the *Computers* category.

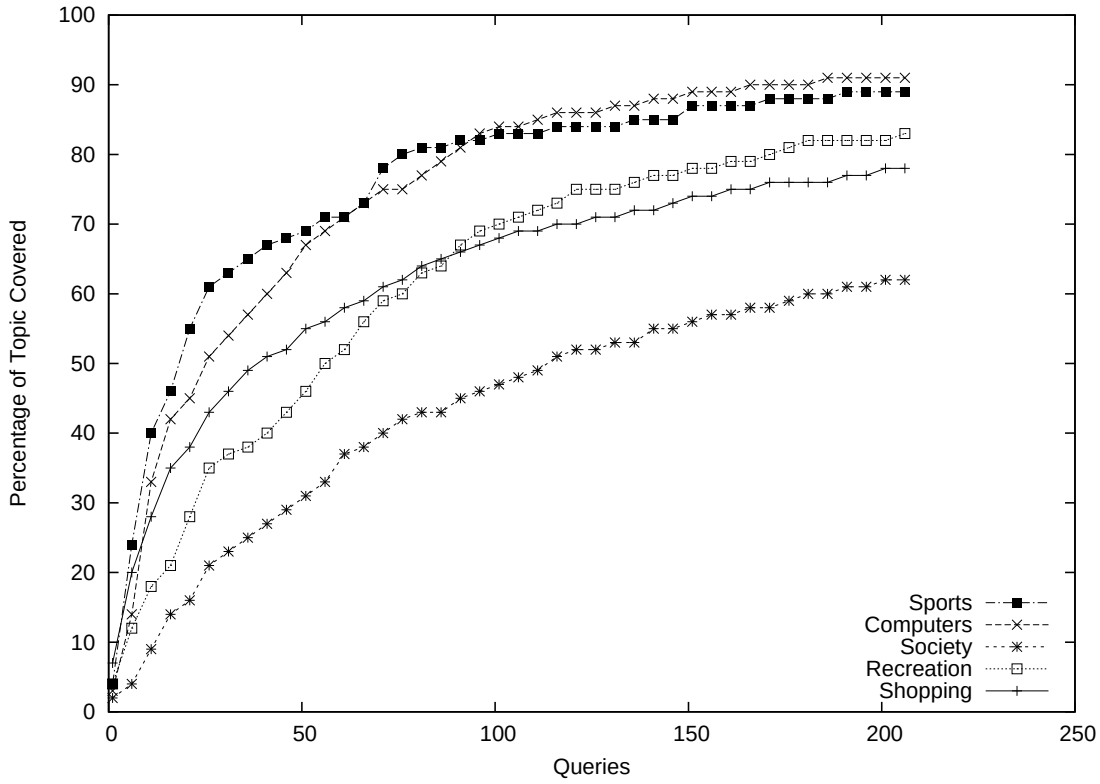

Figure 3.4: Comparison of different topics using *CosineSimilarity* policy

As we can see, as we limit the file's size, the performance deteriorates. However, even for a very small initial document the results are still very satisfactory.

More explicitly, after 190 submitted queries, and using an input document that consists of 1,000 titles and summaries of links indexed by *dmoz* and categorized under the topic *Computers*, *the CosineSimilarity* policy retrieved 91.14% of the relevant documents.

With an input document of 100 titles and summaries, the policy discovered 87.77% of the documents after the same number of queries and finally, with an input document of 50 titles and summaries, it retrieved 86.67% of them.

3.6 Impact of the NOT operator

The upper limit that *dmoz* enforces on the number of returned results plays an extremely significant role in the performance of our algorithm. We retrieve only a subset of the documents that match with every chosen query and we have to issue new queries to retrieve the rest of them. In order to deal with this issue, we examined the use of the NOT operator in

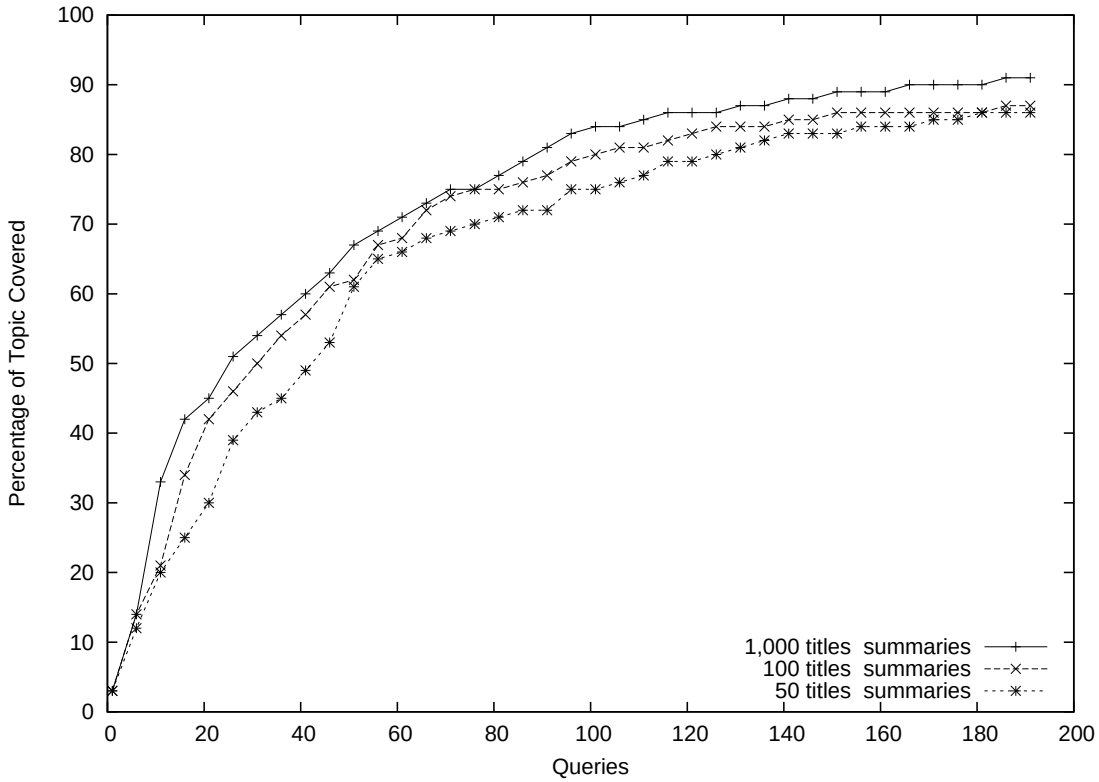

Figure 3.5: Comparison of document size using *CosineSimilarity* policy on Topic *Computers*

our queries.

For every term that has returned a number of documents smaller than the upper limit, we can be sure that we have retrieved every document, and thus, by using the NOT operator with this term for every query for the rest of the execution of our algorithm, we can improve our results.

It is clear however, that we cannot do the same thing for terms that have returned the maximum number of links, because in that way we would exclude from our search the rest of their matching documents. Since a lot of terms have reached that limit in our tests it was expected that the impact of the NOT operator would not be significant.

We tested this approach with the *perfect* and *CosineSimilarity* policies for the topic *Sports* and noticed little improvement for the former and very limited for the latter policy.

3.7 Comparison with simple Hidden Web Crawling

In [13], an experiment on crawling all of the *dmoz* site's documents is presented. It takes about 700 queries to retrieve a little over 70% of its contents.

The *CosineSimilarity* policy proposed in this thesis was able to download 70% of topics *Sports* and *Computers* after 52 and 60 submitted queries respectively. Therefore, it is clear that our algorithm can significantly reduce the cost of crawling.

Chapter 4

Related Work

Research on the Hidden Web has emerged during the last years. In [14], Raghavan et al. first introduced the problem of Hidden Web crawling by presenting an architectural model for a Hidden Web crawler. Their efforts in this work, mainly focus on learning Hidden-Web query interfaces. The potential queries are either provided manually by users or collected from the query interfaces.

Similar work is presented in [5] where some new strategies for effective discovery of Hidden Web forms.

In [13, 4, 16] the problem of automatically producing meaningful queries that can return large fractions of a document collection. [13] provides a theoretical framework for analyzing the process of generating queries for a document collection as well as examining the obtained result. In addition, the framework is applied to the problem of *Hidden Web crawling* and the efficiency of the approach is quantified. In citeBarbosa04siphoninghidden-web, a number of methods for building multi-key word queries is presented and experimentally evaluated to show that a large fraction of a document collection can be returned. Wu et al. [16] focus on the core issue of enabling efficient Web Database crawling through query selection and propose a theoretical framework that transforms the database crawling procedure into a graph-traversal problem; in the latter, the proposal is based on following up “relational” links. Our work differs from the above efforts as we retrieve only portions of a *Hidden Web* site that are relevant to a specific topic.

Chakrabarti et al. studied ways to selectively search for pages relevant to a pre-defined set of topics [7]. A complete process for discovering topic-specific resources from the Web is presented. A classifier and a distiller are used for the evaluation of crawled pages based on their hypertext. However, this methodology is obviously only applicable on the publicly indexable web, since hyperlinks are not used at all on the Hidden Web. Since then, there has been many similar work made [12, 8, 11, 3], again for the publicly indexable web.

In [18], Yang et al. propose a method for extracting topic-specific documents by issuing queries. Candidate phrases are extracted from an input document and a score is assigned to each one of them, using two different policies. The first is a linear combination of the total

TF/IDF score of all the terms of the phrase and their degree of coherence. The second is based on the probability of pairs of terms occurring in the same document (mutual information based). Our approach is different because we try to capture the general topical area of the input document and discover all the relevant fraction of the document collection, instead of retrieving just the most relevant pages.

In [2], a focus crawler for Hidden Web is presented. Its main focus is on extracting information from labels of form elements, to associate Hidden Web sites to topic domains and to analyze the forms to find ways to execute queries automatically. However, the queries to be issued are not produced automatically, but are picked from predefined resources.

Bergholz et al. have also addressed this matter in [6]. However, as the case was before, they use predefined sets of keywords name "querying resources" in order to retrieve results from the collections it discovers.

In [10], Ipeirotis et al. attempted to categorize Hidden Web Databases by probing them with queries and evaluating the amount of the results. They used pre-classified documents to train a rule-based document classifier and transformed the rules it produced, to queries for use.

Chapter 5

Conclusion and Future Work

Crawling of the Hidden Web has recently attracted a lot of attention. Most of these efforts have focused either in discovering forms of Hidden Web sites or automatically retrieving their contents. The observed and well-documented growth of the Hidden Web renders the downloading of a pertinent such site an expensive and time-consuming exercise.

In this thesis, we examine how we can build a Topic-Sensitive Hidden Web Crawler, that given an exemplary document, can effectively retrieve those documents that are relevant to a certain topic. By avoiding the downloading of irrelevant pages, we limit the crawling requirements in terms of both hardware and network resources. Therefore, crawling becomes less expensive and the search engines that index the downloaded pages can more frequently update their data. We presented an algorithm for a Topic-Sensitive Hidden Web Crawler that uses the *TF/IDF* weighting system to extract appropriate terms. We also proposed and experimented with a variety of policies deployed to help us measure the relevance of the returned documents with the searched topic. Documents that successfully go through the check of our policies are used to continuously enrich the pool of words used in crawling. Our experimental evaluation indicates that our suggested algorithm has great potential for harnessing topic-specific documents. In the context of our work, we managed to successfully retrieve the majority of the documents related to a number of topics from the *dmoz* site. This took place with a significantly smaller amount of queries than it would be required to retrieve the site in its entirety.

In the future, we plan to exploit diverse query formulations in order to further reduce the overheads involved in the process, to experiment with additional Hidden Web sites that may freely provide their data, and finally, to explore techniques that may incrementally retrieve recently updated content.

Terminology

Ξενόγλωσσος Όρος	Ελληνικός Όρος
crawler	εφαρμογή προσκομιδής ιστοσελίδων
Hidden Web	Κρυμμένος Παγκόσμιος Ιστός

Acronyms and Abbreviations

Abbreviation	Full Name
IDF	Inverse Document Frequency
TF	Term Frequency

References

[1] The open directory project `http://www.dmoz.org`.

[2] Manuel Álvarez, Juan Raposo, Alberto Pan, Fidel Cacheda, Fernando Bellas, and Víctor Carneiro. Deepbot: a focused crawler for accessing hidden web content. In *Proc. of the 3rd Int. Workshop on Data Enginering Issues in E-commerce and Services*, DEECS '07, pages 18–25, New York, NY, USA, 2007. ACM.

[3] Niran Angkawattanawit and Arnon Rungsawang. Learnable crawling: An efficient approach to topic-specific web resource discovery, 2002.

[4] Luciano Barbosa and Juliana Freire. Siphoning hidden-web data through keyword-based interfaces. In *In SBBD*, pages 309–321, 2004.

[5] Luciano Barbosa and Juliana Freire. An adaptive crawler for locating hidden-web entry points. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 441–450, New York, NY, USA, 2007. ACM.

[6] André Bergholz and Boris Chidlovskii. Crawling for domain-specific hidden web resources. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, WISE '03, pages 125–, Washington, DC, USA, 2003. IEEE Computer Society.

[7] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. In *Proceedings of the eighth international conference on World Wide Web*, WWW '99, pages 1623–1640, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[8] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 527–534, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[9] Panagiotis G. Ipeirotis and Luis Gravano. Distributed search over the hidden web: hierarchical database sampling and selection. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 394–405. VLDB Endowment, 2002.

[10] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: categorizing hidden web databases. *SIGMOD Rec.*, 30:67–78, May 2001.

[11] Filippo Menczer, Gautam Pant, and Padmini Srinivasan. Topic-driven crawlers: Machine learning issues. *ACM TOIT, Submitted*, 2002:http://dollar.biz.ui, 2002.

[12] Relevance Sanguk Noh, Sanguk Noh, Youngsoo Choi, Haesung Seo, Kyunghee Choi, and Gihyun Jung. An intelligent topic-specific crawler using degree of.

[13] Alexandros Ntoulas, Petros Zerfos, and Junghoo Cho. Downloading textual hidden web content through keyword queries. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '05, pages 100–109, New York, NY, USA, 2005. ACM.

[14] Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 129–138, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[15] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. In *INFORMATION PROCESSING AND MANAGEMENT*, pages 513–523, 1988.

[16] Ping Wu, Ji-Rong Wen, Huan Liu, and Wei-Ying Ma. Query selection techniques for efficient crawling of structured web sources. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 47–, Washington, DC, USA, 2006. IEEE Computer Society.

[17] Wensheng Wu, Clement Yu, AnHai Doan, and Weiyi Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 95–106, New York, NY, USA, 2004. ACM.

[18] Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis Ipeirotis, Nick Koudas, and Dimitris Papadias. Query by document. In *Proc. of the 2nd ACM Int. Conf. on Web Search and Data Mining*, WSDM '09, pages 34–43, New York, NY, USA, 2009. ACM.