WEB MINING ISSUES:

TOPIC FINDING AND FOCUSED CRAWLING EVALUATION

by

Eray Uluhan

Submitted to

The Institute of Graduate Studies in Social Science

in partial fulfillment of the requirements

for the degree of

Master of Arts

Boğaziçi University

2006

The thesis of Eray Uluhan

is approved by:

Asst. Prof. Bertan Badur _____
(Thesis Advisor)

Asst. Prof. Osman N. Darcan _____

Assoc. Prof. Can Özturan _____

June, 2006

ABSTRACT

WEB MINING ISSUES:

Topic Finding and Focused Crawling Evaluation

by

Eray Uluhan

Web mining is defined as the process of using data mining techniques to automatically discover and extract information from semi- or unstructured Web documents and services. This study on Web mining consists of two sections, covering Web structure mining and Web content mining. In the first section, most widely accepted focused crawling algorithms and simple tree traversing algorithms are compared based on their page relevance, keyword predicate satisfaction and hit ratio criteria. Using the URL tokens as an input resulted in higher performances for all criteria. In the second part, an automatic topic finding methodology through Web pages is proposed. Processing only list items on HTML pages returned from a search engine, it is expected to find related key concepts on a user-defined topic. The methodology is experimented using different parameters, such as number of pages, different keywords, stemming implementations, etc. The candidate concepts ordered in relevancy scores represent a high precision on user-defined topic.

ÖZET

ÖRÜN MADENCİLİĞİ KONULARI:

Konu Bulma ve Odaklanmış Arama Değerlendirmesi

Eray Uluhan

Örün madenciliği, veri madenciliği ve düzyazı madenciliği teknikleri kullanılarak yarı yapılanmış ya da hiç yapılanmamış örün dökümanları ve servislerinden otomatik olarak bilgi ortaya çıkarmak ve elde etmektir. Örün madenciliği hakındaki bu çalışma iki bölümden oluşmaktadır; örün yapı madenciliği ve örün içerik madenciliği. İlk bölümde, en çok kabul görmüş olan odaklanmış arama algoritmaları ile basit ağaç izleme algoritmaları, sayfa ilgililik derecelerine, anahtar kelime içermelerine ve isabet oranlarına göre karşılaştırılmışlardır. URL içerikleri girdi olarak kullanıldıklarında tüm kriterler için en yüksek performans değerlerine ulaşılmıştır. İkinci bölümde, örün sayfaları üzerinden bir otomatik konu bulma metodolojisi önerilmiştir. Bir ara motorundan dönen HTML sayfalarındaki sadece liste maddelerinin işlenmesiyle, kullanıcı tarafından belirlenmiş olan bir konu ile ilgili önemli başlıklar bulunabilir Bu metodoloji farklı parametreler – sayfa sayısı, farklı konular, kök bulma uygulaması, vb.- kullanılarak test edilmiştir. Bulunan aday kelimeler ilgililik puanlamalarına göre sıralandıklarında kullanıcının belirlediği kelime ile yüksek doğruluk oranları göstermişlerdir.

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge the enthusiastic supervision of my thesis advisor Bertan Badur, who encouraged me to choose this topic and began the thesis. Most of this thesis work has been done while working as the research assistant at the MIS department. I am grateful to everyone in the department, for helping and assisting me in many different ways, especially to my research assistant colleagues (especially Gonca Gülser for her support and motivation during all phases of this thesis).

Furthermore, I would like to thank Deniz Akay for his great friendship, over 15 years now and also Tolga Silivri, both for always being ready to help, for all the emotional support, entertainment, and the other whole nonsense time spent together! Also thanks to all my close friends – especially to mention “Kotkazak Camia”, for their encouragement and motivation they have given during this thesis.

Lastly, and most importantly, I wish to thank my parents Ayla Uluhan and Sabit Uluhan, and my brother, Koray Uluhan. They raised me, supported me, taught me, and loved me. To them I dedicate this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

The World Wide Web is the largest information repository that has become a shared global resource of information, knowledge and a means of collaboration among countless people and communities around the world. However, the largest information repository in existence lacks a schema, i.e. information on WWW has been characterized as either unstructured, meaning that it has no schema, or semi-structured, meaning that it has a very primitive structure (e.g., such as that induced by the HTML tags on each page). The use of data mining and text mining techniques to discover resources, patterns and knowledge from this *ill-structured* repository is called Web mining. With the increase in popularity of Internet and diversifications in its usage, implementations of Web mining techniques became a necessity.

My thesis is based on two important sub domains of Web mining, focused crawling techniques, and automatic discovery of information through Web pages. The two areas are handled separately throughout this study. Focused crawling algorithms will be an important concept in the coming years, especially as the specialized Web portals become popular. Internet users will try to stay away from unnecessary information chunks and will favor on high quality, informative pages. In addition, specialized applications that use Web pages as their databases will become more popular.

The main problem on working with the Web pages is their lack of structure. Although widely used HTML standard offers some structure through tags, it is rarely

intended to be used for building logical corpus on Web pages. Most of the Web site administrators use HTML tags just to *decorate* their pages, not considering the informative structure of a topic. In addition, browsers, which are fault tolerant to HTML, such as Internet Explorer, make Web designing task is easier, but also gave rise to increase in unstructured pages.

As long as satisfactory information can be found on Internet (using search engines, forums, blogs, or other applications), its popularity will not decrease. To overcome the problems with the unstructured nature of the Web documents new standards are being developed, such as XHTML, but still none of these are being used as much as HTML.

CHAPTER I.

INTRODUCTION

This thesis consists of two parts. First part is about evaluation of focused crawling algorithms, and the second part deals with finding topic specific information from Web documents. Both of these works are related to Web mining, defined as the process of using data mining techniques to automatically discover and extract information from Web documents and services.

With the exponential growth of Internet in the last ten years, World Wide Web has become one of the largest repositories of information available to people. When the Internet was first introduced and Web pages were small in quantity, information finding, categorizing and sorting was done mainly by humans, and there were no need for automated Web search engines.

Today, the main problem that the search engines face is the size and the rapid change of the Web. As the number of pages grows, it will be increasingly important to focus on the most "valuable" pages, as no search engine will be able to index the complete Web. Also, with the increase of information on the Web, need for dedicated search engines come out. To be able to create dedicated search engines, crawlers should not visit each link they encounter, but only those which are related to their goal.

This study focus on two main topics of Web mining, namely Web crawling and Web content mining. Firstly, we implemented most widely known focused crawling algorithms in the literature and analyzed their performances according to different criteria. Secondly, we developed a topic finder system that searches for most related concepts to a given keyword. Different parameters were selected and their effects were evaluated.

The following is an outline of the contents of this thesis. Both result and methodologies of topic finding and focused crawling evaluation studies are described in different chapters.

- Chapter 2 reviews selected publications related to both topics, including Web mining implementations, Web crawling infrastructures and algorithms, topic and definition finding.
- Chapter 3 introduces the methods of our focused crawler algorithm implementations and the problems faced in general crawling processes, as well as our solutions.
- Chapter 4 compares the results of the focused crawling algorithms.
- Chapter 5 details implementation issues related to the design of the topic finding application.
- Chapter 6 presents the results of topic finder with different parameters and discusses their effects on the results.
- Finally, Chapter 7 summarizes our contributions and provides guidelines for future work in this area. At the end, the bibliography includes over 35 references to publications in this area.

The next chapter is a literature survey about the most important ones in the context of this thesis.

CHAPTER II.

LITERATURE SURVEY

Internet was first introduced in the late 1960s (at that time called as ARPANET) for allowing computers to share information on a highly decentralized network for research development and military areas. As interest in wide spread networking grew and new applications for it arrived, the Internet's technologies spread throughout the rest of the world. In last ten years, the amount of information published through Web pages has grown so much that its sheer volume makes finding relevant information difficult. With the exponential growth of the Internet, collecting all the Web pages became nearly impossible even for search engines because of limitations in indexing and search technologies. (Broder, 2002), (Henzinger, et al. 2003) and (Tirri, 2003) summarizes some of the current challenges of search engines as fresh and complete indexing, malicious content, search engine spam, identifying good quality of pages, exploiting user feedback and identifying real need of the user, mirroring of Web sites, better query languages and ranking algorithms, classification of Web sites.

Today, the most widely used language in Web pages is HTML (Hypertext Markup Language), which first introduced in 1993 and accepted as a standard in 1995. Although HTML is still popular for disseminating informal documents and is designed for stepwise exploration and easy navigation through links, it is being used

to do things it was never designed for, such as formatting and displaying data. In addition, although Web pages are mainly used for information sharing, HTML provides little or no semantic structure at all. In contrast, most Web applications would benefit from an ability to represent data by meaning rather than by layout, where a more useful markup language would represent information in terms of its meaning, e.g. XML.

Traditional Information Retrieval (IR) is concerned with retrieving information about a subject from a collection of data objects. (Huang, 2000) compares classical information retrieval systems and Web information retrieval systems and summarizes the differences as size, dynamic structure of Internet, heterogeneity, variety of languages, duplication, high linkage, ill-formed queries, wide variance of users and specific user behaviors.

Web Mining

The large size and the dynamic nature of the Web made "Web mining" a necessity. Web mining is firstly defined in (Etzioni, 1996), as "the use of the data mining techniques to automatically discover and extract information from Web documents and services. It states a hypothesis that the information on the Web is sufficiently structured to facilitate effective Web mining. (Cooley, et al. 1997) categorizes the Web mining into two categories, Web usage mining and Web content mining, whereas (Madria, et al. 1999) and (Borges, et al., 1999), add one more category, Web structure mining.

Web Usage Mining

Web usage mining focuses on the techniques for finding general access patterns of people through Web pages. The data used in Web usage mining are

collected on Web logs and consist of user clicks, visited sites, referrer pages, time spent on a site, entry and exit pages to a domain, etc. According to (Cooley, et al. 1997), Web usage mining can help organizations "to determine the life time value of customers, cross marketing strategies across products, and effectiveness of promotional campaigns, as well as provide information on how to restructure Web site to create a more effective organizational presence".

Web Structure Mining

The goal of Web structure mining is to generate structural summary about the Web sites and Web pages. Technically Web content mining mainly focuses on the structure of inner-document, while Web structure mining tries to discover the link structure of the hyperlinks at the inter-document level. Based on the topology of the hyperlinks, Web structure mining try to categorize the Web pages and generate information using the similarities and relationships of different Web sites. Web structure mining can also be used for discovering the structure of Web documents rather than Web sites. This type of structure mining can be used to reveal the structure of Web pages, to make it possible to compare / integrate Web page schemes, and would serer for introducing database techniques for accessing information in Web pages by providing a reference schema.

Web Content Mining

Web content mining aims to extract/mine useful information or knowledge from Web page contents. Web content mining is related but different from data mining and text mining. It is related to data mining because many data mining techniques can be applied in Web content mining. It is related to text mining because much of the Web contents are texts. However, it is also quite different from data

mining because Web data are mainly semi-structured and/or unstructured, while data mining deals primarily with structured data. Web content mining is also different from text mining because of the semi-structure nature of the Web, while text mining focuses on unstructured texts. Web content mining thus requires creative applications of data mining and/or text mining techniques and also its own unique approaches.

In (Liu, 2003), it is intended to extract topic-specific knowledge, such as subtopics or definitions from the Web; a highly challenging task on an ill-structured domain area. The motivation behind automatic discovery of salient concepts or subtopics on the Web is guiding people to learn in-depth knowledge of a topic on the Web easily. Traditionally, when someone wants to learn about a particular topic, reads a book or a survey paper. The rapid growth of the Web, popularity and richness of information published on Web sites made learning in-depth knowledge about a topic from the Web very easy and also even essential because of the fast changing world, constant and rapid emerging of topics. Many Web pages often contain intuitive descriptions of the topic. To find such Web pages, one typically uses a search engine. However, current search techniques are not designed for in-depth learning. Top ranking pages from a search engine may not contain any description of the topic. Even if they do, the description is usually incomplete since it is unlikely that the owner of the page has good knowledge of every aspect of the topic.

Web Crawlers

A Web crawler is a program that automatically downloads pages from the Web, parse their context information and extract links for future crawling. A typical crawler starts with a seed of set pages. It then downloads these pages, extracts hyperlinks and crawls pages pointed to by these new hyperlinks. The crawler repeats this step until there are no more pages to crawl or some resources (e.g. time or

network bandwidth) are exhausted (Brandman, et al. 2000). Web crawlers are also called wanderers, robots, spiders, fish and worms (Pant, et al. 2004).

Framework / Crawler Infrastructure

(Menczer, et al. 2002), (Pant, et al. 2004) discuss general framework for crawlers. First of all, a general crawler has to have access to Internet to download Web pages for data gathering, a HTML parser to extract information and links to follow, and a queue to put links to be visited. Depending on the goal of the crawler, a database may be used for storing downloaded pages and querying them easily, or just for caching purposes.

To start crawling, a crawler needs to have *seed pages*, i.e. starting pages. Seed pages can be given by the user (Brin and Page, 1998), (Menczer, 2002a), or can be gathered via querying search engines (Srinivasan, et al., 2002), or using backlink queries of search engines (Srinivasan, et al., 2005). Backlinks refer to incoming links to a Web site and also called incoming links, inbound links, inlinks and inward links. Backlink pages can be obtained from search engines, such as Google – using the keyword *"link:"*. In (Srinivasan, et al., 2005), the user sets a depth value and queries the search engine with the keyword. The pages returned by the search engine sent again back to the search engine as backlink queries, and pages that have links to these pages are returned. This process continues iteratively as long as the predefined depth value.

The goal of the crawling process may differ according to the need of the user. For example, a crawler may run until a predefined number of pages are downloaded, until some time is elapsed, no more memory / space available left, or a more general goal is completed. While crawling, the crawler maintains a list of unvisited URLs in a queue. Each time a page is downloaded by the crawler, it is parsed to extract URLs

and application specific information. The unvisited URLs on that page are added to the queue ordered depending on the goal of the crawler and queue implementation. Before the URLs are added to the queue they may be assigned scores according to an algorithm, and then put the queue in order, so that higher scoring links are fetched first. If, in any time, the queue is empty and the crawler has finished fetching all the links encountered, than the situation signals a dead-end for the crawler.

Crawlers are mostly used by search engines to index and refresh their database information. Also, there are specific purpose crawlers (Srinivasan, et al., 2005), (Menczer, et al., 2001), (Diligenti, et al., 2000), (Chakrabarti, et al., 1999a). In (Chakrabarti, et al., 1999a), crawlers are used for classification of pages into categories using an existing document taxonomy and seed documents. For classification purposes, (Chakrabarti, et al., 1999b) builds a model with crawlers that online pre-trained by samples consisting of source page features and the relevance of the target page. The training process results in significant decrease of false positives in classification.

(Ehrig, et al., 2003) considers an ontology-based algorithm for relevance computation. After preprocessing, entities are extracted from page and counted. Relevance of the page with regard to user selected entities of interest is then computed by using several measures on ontology graph. The harvest rate is improved and compared to the baseline focused crawler (that decides on page relevance by a simple binary keyword match).

Focused crawlers are firstly introduced by (Chakrabarti, et al., 1999a), (Chakrabarti, 2003). The basic idea of the crawler was to classify crawled pages with categories in topic taxonomy. At the beginning, the crawler requires a topic taxonomy such as Yahoo and ODP (Open Directory project a.k.a. DMOZ –

Directory.MOZilla.org). In addition, the user provides example URLs of interest. The example URLS get automatically classified onto various categories of the taxonomy. Through an interactive process, the user can correct the automatic classification, add new categories to the taxonomy and mark some of the categories as "of interest of the user".

(Aggarwal, et al., 2001) introduce a concept of "intelligent crawling" where the user can specify an arbitrary predicate (e.g. keywords, document similarity – anything that can be implemented as a function which determines documents and relevance to the crawl based on URL and page content) and the system adapts itself in order to maximize the percentage of the Web pages crawled satisfying the predicate. It is suggested that for some types of predicates the topical locality assumption of (Chakrabarti, et al., 1999a), (i.e. relevant pages are located close together) might not hold. A probabilistic model for URL priority prediction is trained using information about content of in-linking pages, URL tokens, short-range locality information and sibling information.

Crawler Algorithms

There are many different crawling algorithms in the literature, and most of them are not only specific to Web mining but also used in traditional information retrieval. The most common algorithms are explained below, briefly.

Breadth-First Algorithm

One of the earliest algorithms used in Web information retrieval, breadth-first algorithm is firstly explored in WebCrawler (Pinkerton, 1994). In most of the crawler performance studies such as (Cho, et al., 1998) and (Najork and Wiener, 2001) breadth first algorithm is used as the simplest and baseline crawler algorithm for

comparisons. In breadth-first algorithm, crawler visits every link of a single page, before moving to a next page, such that it does not use any information collected over the page, anchor texts etc.

Fish-Search Algorithm

Proposed by (De Bra, et al., 1994), fish-search algorithm tries to crawl those areas in the Web more extensively, in which relevant pages have been found. At the same time, the algorithm discontinues, if it does not encounter any important page along the path with a prespecified depth. That is, after following a number of links in a direction without finding a relevant document the search stops investigating that direction.

Relevancy of a document in fish-search algorithm refers to a binary evaluation, whether the document contains the predicate of not.

Shark-Search Algorithm

(Hersovici, et al., 1998) Shark-search is an improved version of Fish-Search algorithm. First of all, it uses "similarity measure" to evaluate the relevance of documents, instead of binary evaluation of fish-search algorithm. A document is relevant if its similarity is above a predefined threshold. In addition, links' potential scores are affected by the anchor text, text around the link tags, and also inherited score from ancestors.

Best-First Algorithm

In Best-First algorithm, a crawler fetches links from a page and gives all of the links the cosine similarity scores of the extracted page. In basic terms, cosine similarity is computed as the lexical similarity between a topic's keywords and the page, where both keywords and pages are treated as vectors.

$$sim_{(q,d)} = \frac{\sum_t w_{t,q} \times w_{t,d}}{\sqrt{\sum_t w_{t,q}^2} \times \sqrt{\sum_t w_{t,d}^2}} \quad (1)$$

When computing cosine similarities between two documents (*q, d*), both are seen as a pair of vectors in a space with as many dimensions as terms (*t*) as the vocabulary. In a space defined in this way, the similarity of two documents is cosine of the angle between these two document vectors (see Equation 1).

The crawler fetches the highest valued links first, so that the probability of fetching unrelated Web pages decreases significantly. Best-First Algorithm is used in (Hersovici, et al., 1998) and (Cho, et al., 1998).

HITS Algorithm

First proposed by (Kleinberg, 1998), Hypertext Induced Topic Search (HITS) algorithm is a query dependent ranking technique, in which the different (hub and authority) scores are produced. Authority page are those pages which have relevant information and discussions about a topic. Hub pages do not need to have in-depth information about a topic but have links to many authoritative pages. The relation between authoritative and hub pages are mutually reinforcing, while an authority page is a page that is pointed by many hubs and hubs are pages that point to many authorities.

PageRank

In (Brin and Page, 1998), the simple PageRank algorithm is introduced also became the basis of Google *(http://www.google.com)* search engine. PageRank is based on the link structure of the Web pages, and an "importance ranking" computed with iterative Web crawls. The importance of a page is represented as the probability that a random surfer (one who follows links randomly from page to page) will be on

that page at any given time. The ranking of the Web pages in PageRank algorithm is based completely on their location in the Web's graph structure. The more important and central Web pages are given higher rankings, whereas backlinks from "important" pages are more significant than backlinks from average pages.

Performance Evaluation Metrics

Crawling algorithms are evaluated using many different types of criteria, such as efficiency, harvest rate, hit ratio, recall, precision etc. We can categorize these criteria into two types of categories for a crawl process, i.e. importance of pages' crawled and general crawling performance (e.g. recall or precision).

For page relevance measures lexical criteria and link based criteria are taken into account. A page is considered relevant if it contains some or all of the keywords in the query. In addition, if the frequency with which the keywords appear on the body of the page exceeds a frequency threshold, the page may be considered relevant (Cho, et al., 1998). In (Amento, et al., 2000), a combined word set is created from the contents of target documents. This word set is used to compute relevancy (page quality) of each crawled page as the cosine similarity between the page's vector and the word set. (Chakrabarti, et al., 1999a) apply classifiers built using positive and negative example pages to determine page importance. (Aggarwal, et al., 2001) adopt a more generic framework that allows for user designed plug-in modules specifying relevance criteria. The modules that they use in their test require the presence of pre-specified words in particular parts of the page, such as the URL. Similarity to the topic measured using page text (Bharat and Henzinger, 1998) or the words surrounding a link (Chakrabarti, et al., 1998) may also be used to augment what are primarily link based relevance measures.

In-degree, out-degree, PageRank (Brin and Page, 1998), hubs and authorities are commonly used link based page importance measures (Amento, et al., 2000), (Ben-Shaul, et al., 1999), (Bharat and Henzinger, 1998), (Chakrabarti, et al., 1998), (Chakrabarti, et al., 1999a), (Cho, et al., 1998). (Cho, et al., 1998) consider pages with PageRank above a specified threshold as being relevant to the query. (Kleinberg, 1998) recursive notion of hubs and authorities has been extended by several others. For example, edge weights are considered important (Chakrabarti, et al., 1999a) and so are edges that connect different sites (Amento, et al., 2000), (Bharat and Henzinger, 1998), (Chakrabarti, et al., 1999a). Link based quality metrics rely on the generally reasonable notion of link reflecting the credibility of the target pages. (Amento, et al., 2000) show that in-degree, authority and PageRank are effective at identifying high quality pages as judged by human experts.

The literature shows a wide variety of summarization methods. Given a particular measure of page importance, (Cho, et al., 1998) examine the percentage of important pages retrieved over the progress of the crawl. (Menczer, et al., 2000) measure search length, i.e., the number of pages crawled until some predetermined fraction of important pages have been visited. (Chakrabarti, et al., 1999a), (Chakrabarti, et al., 1999b) compute the average "harvest rate," which is a running average, over different time slices of the crawl, of page relevance assessed using classifiers. (Aggarwal, et al., 2001) also use harvest rate, similarly defined as the rate at which crawled pages satisfy a given predicate; if a classifier is used to give numeric relevance values then a page is said to satisfy a predicate if the relevance value exceeds a certain threshold. (Rennie and McCallum, 1999) compute the percentage of relevant pages found.

In this chapter, we have surveyed selected publications from the related work that are relevant for this thesis. We discussed Web mining concepts and focused on the Web crawling algorithms, frameworks and evaluation metrics.

The next chapter starts the main part of this thesis by presenting the focused crawler implementation.

CHAPTER III.

FOCUSED CRAWLING METHODOLOGY

A simple Web crawler works as follows; crawler starts with a set of seed pages (given by the user or retrieved from an external source such as a search engine), and then uses external links of these pages to attend to other pages. The process continuously iterates with new pages offering more external links to follow, until a sufficient number of pages are retrieved or some higher level goal is reached. Iterations involve picking the next URL to crawl from the queue, fetching the page corresponding to the URL through HTTP, parsing the retrieved page to extract the URLs and application specific information, and finally adding the unvisited URLs to the crawl queue (see Figure 1). A topical crawler's goal is to fetch only those pages which are relevant to a query or topic, rather than downloading all accessible Web pages.

The main difference between a simple and a topical crawler is the implementation of crawl queue - priority queue. In topical crawlers, retrieved pages are evaluated for topic relevance; the extracted links are given scores and put into priority queues in order. This evaluation may range from a simple keyword matching to complex machine learning algorithms, may use information of previous runs, or even update score calculation weights during a crawl process.

```
Input: u1, u2, ..., un starting URLs, t topic
1: PQ = {u1, u2, ..., un}, priority queue of URLs to visit.
2: V = , visited URLs.
3: while PQ != empty && overall goal is not reached do
4:       Dequeue u from PQ, select u with highest priority.
5:       Fetch u as Page p.
6:       Add u into V.
7:       Parse p to extract text and extract outgoing links ux
8:       for each ux in p do
9:             if ux not in V then
10:                  Compute priority cp of ux for t
11:                  Add ux into PQ with cp
12:            end if
13:      end for
14: end while
```

Fig. 1. A general crawling algorithm

Before getting into detail, we need to emphasize some limitations related to the crawling algorithms. First of all, memory size prohibits crawl queues to be infinitely large. Although it is possible to store the queue completely on a disk, for performance reasons this is not preferred, especially if a multi-thread crawler runs. Alternatively, we can limit queue size by specifying a maximum number. In that case, we need to implement a decision mechanism to choose which URLs to remove or ignore when the size limit is reached. For a priority queue, it is obvious that whenever the queue is full and new URLs needs to be added, then URLs with the lowest priorities in the queue should be removed. In our study, we set the priority queue size to 512.

In addition, due to the limited size of the queue, we need to make sure that we do not add duplicate URLs to crawl queue. Considering a linear search – O(n) - to find out if a newly extracted URL is already in the queue is costly. So hash tables are preferred for holding unvisited URLs in the queue and also another hash table should be used for processed URLs, because search performance in hash tables is much more better – O(1), for unique hash keys – than linear and binary search. Once the

crawl queue's maximum size is exceeded, only URLs with highest priorities shall be kept in the queue. The problem with hash tables is that they can not store the keys in a specified order. Hence, we used Vectors to store URLs in priority order. If at any time the crawler finds the queue empty, when it needs the next URL, the process comes to a dead-end. However, with a large value of queue size and a set of seed URLs the crawl process will hard to encounter a dead-end.

In order to fetch a Web page, we need an HTTP client which sends an HTTP request for a page and reads the response. The client needs to have timeouts to make sure that an unnecessary amount of time is not spent on slow servers or in reading large pages. The client needs to parse the response headers for status codes and redirections. Error checking and exception handling is important during the page fetching process since we need to deal with millions of remote servers using the same code. Modern programming languages such as Java and Perl provide very simple and often multiple programmatic interfaces for fetching pages from the Web. In our applications we used ready-made java.net.HttpURLConnection class for our HTTP client.

Crawler Problems

Whenever a page has been fetched by the crawler, its content has to be parsed, URLs encountered has to be extracted and put into the crawl queue, which will ensure that the crawling process will continue. However, there are several problems, which need to be solved in a crawling process, related to network connections, spider traps, canonicalizing URLs, parsing HTML pages, and the ethics of dealing with remote Web servers.

Canonicalizing URLs

In order to extract hyperlink URLs from a Web page, we need a parser to find the anchor tags and extract their associated href attribute values. However, the structure of these attribute values can differ a lot, so that same page can be linked by different encoded URLs. So we have to convert all URLs encountered according to some criteria, in order to avoid fetching the same page many times. The conditions we applied are as follows:

- Converting each URL to lowercase. For example, HTTP://www.BOUN.edu.tr is converted to http://www.boun.edu.tr.
- Removing the 'anchor' or 'reference' part of the URL, because reference does not affect the output of a page and it is only used for navigational purposes. http://www.boun.edu.tr/calendar/index_tur.html#takvim2006 is reduced to http://www.boun.edu.tr/calendar/index_tur.html.
- HTML encoding of some commonly used characters such as '~' to '%7E', or ' ' '%20'. http://www.boun.edu.tr/~mis is transformed to http://www.boun.edu.tr/%7Emis.
- Adding '/'s to the end of an URL, if it ends with a directory name, so that "http://www.boun.edu.tr" and "http://www.boun.edu.tr/" are treated as the same URLs.
- If the host part of an URL does not have www prefix, we retrieve the page with the original URL but put it into the visited URL list by adding www as prefix, because nearly for all Websites both "www.xyz.com" and "xyz.com" names resolve to the same IP address, and have the same contents. So we do not download the same content twice.

Stoplisting and Stemming

Stoplisting and stemming are two common techniques also used in traditional information retrieval for working with higher quality text data. Removing commonly used words or stopwords from text is called *stoplisting*.

In addition to stoplisting, word stemming is also used to find the roots of words found in a page. The *stemming* helps to normalize the words by removing commonly used suffixes, and converting morphologically similar words to a single root form or stem. The most common stemming algorithm used in information retrieval is Porters algorithm and its implementation can be easily found in many programming languages (Porter, 1980).

Network and CPU - Multithreading

A sequential crawling loop spends a large amount of time in which either the CPU is idle (during network/disk access) or the network interface is idle (during CPU operations). Multi-threading, where each thread follows a crawling loop, can provide reasonable speed-up and efficient use of available bandwidth. Each thread starts by locking the queue to pick the next URL to crawl. After picking a URL it unlocks the queue allowing other threads to access it. The queue is again locked when new URLs are added to it. The locking steps are necessary in order to synchronize the use of the queue that is shared among many crawling loops (threads). In addition to crawl queue, a typical multithreaded crawler would also maintain a shared history data structure for a fast lookup of URLs that have been crawled. Hence, in addition to queue it would also need to synchronize access to the history.

The multi-threaded crawler model needs to deal with an empty queue just like a sequential crawler. However, if a thread finds the queue empty, it does not

automatically mean that the crawler as a whole has reached a dead-end. It is possible that other threads are fetching pages and may add new URLs in the near future. One way to deal with the situation is by sending a thread to a sleep state when it sees an empty queue. Whenever a thread access to the queue and put a new URL in it, also a wake signal is sent all of the sleeping threads. When the thread wakes up, it checks again for URLs. A global monitor keeps track of the number of threads currently sleeping. Only when all the threads are in the sleep state does the crawling process stop.

Spider Traps

The Web is usually considered as a collection of pages, in the same sense as in traditional Information Retrieval collections. The Web graph has a finite number of nodes in which measures such as diameter are well defined. However, the amount of information in the Web at any given time is certainly finite, but when a dynamic page leads to another dynamic page, the number of pages can be potentially infinite. Take for instance a dynamic page that implements a calendar, you can always click on “next month” and from some point on there will be no more data items in the calendar; humans can be reasonably sure that it is very unlikely to find events scheduled 50 years in advance, but a crawler can not. A second example would be a calculator, such as a dynamic page that calculates approximations of π using an iterative method. A crawler cannot tell when two pages reflect the same information.

To eliminate the effects of a spider trap, we can limit the number of pages the crawler sequentially accesses from a given domain. The code associated with the queue can make sure that every consecutive sequence of k (say 100) URLs, picked by the crawler, contains only one URL from a fully qualified host name (e.g.

www.cnn.com). As side-effects, the crawler is polite by not accessing the same Web site too often, and the crawled pages tend to be more diverse.

Another solution can be limiting the URL sizes to a number of characters, such 150 or 200. Most of the time the “dummy” URLs created by spider traps often become increasingly larger in size. In addition, a list of pages and sites can be supplied to crawler for excluding from the process to avoid infinitely large automatically generated crawler traps

HTML Parsing

HTML coding, when done by hand, tends to be syntactically very relaxed. Most HTML coders only check if the page can be seen in their browsers, without further checking for compliance. These result in malformed markups, and pose serious problem for HTML parsing. The parsing problems we have faced during our crawler implementation are as follows:

- Mixing single quotes, double quotes, and no quotes in attributes, e.g.: <IMG ALT="This is a photo" SRC=‘photo.jpg’ border=1>.
- Mixing empty tags in HTML form (such as
) and in XHTML form (such as
).
- Unbalanced tags, e.g.: <SMALL><B>...</SMALL></B>.
- Mixed case in tags and attributes, e.g.: <IMG src="...">. For HTML, the tags should be written in uppercase, and for XHTML, in lowercase.
- Unterminated strings, e.g.: <IMG ALT="...>. This can be very problematic, because it will cause a buffer overflow if the parser is not properly written. These unterminated or long strings can also appear in HTTP response codes.

HTTP Requests

In some cases, it is impossible to tell the type of a file just by looking at its URL. Some URLs have no extensions, and some URL have extensions that are ambiguous, e.g.: links to files ending in .exe could be either links to dynamically generated HTML pages in the server side, or links to programs that should be downloaded.

A user agent, such as a Web browser or a Web crawler, can have limited capabilities and only be able to handle some data types. If it cannot handle a file (e.g.: an image), then it should not download it. For instance, a Web crawler searching only for plain text and HTML pages should issue a request of the form:

```
GET /page.html HTTP/1.1
Accept: text/plain, text/html
...
```

This indicates that the Web crawler can only handle plain text or HTML documents. According to the HTTP specification, the server should send a 406 (not acceptable) response code, when a valid object of the desired type is not present at the given URL.

Several Web browsers simply issue a header of the form “Accept: */*”, so some Web server implementations do not check the “accept” header at all. It has somehow lost relevance, and today a Web server can send a response with almost any data type. A related concern is that some Web sites return a header indicating content-type HTML, but the information returned is a large binary file (such as a ZIP archive, etc.). The crawler can waste bandwidth downloading such a file.

We check the returned content-type header in the downloaded pages, as it might not be a data type that the Web crawler can handle. A download limit is

necessary because potentially any file type can be returned by the Web server, even when it is indicating HTML content type.

To ensure a good coverage of the Web, we must limit the amount of data that is downloaded from every Web server. This can be done by limiting both the maximum page size, and the number of Web pages that are downloaded from a single Web site. We set the maximum number of pages to download from a Web site to 100, and maximum page size to 400KB. In case the maximum page size is exceeded, the Web crawler must disconnect from the Web server and continue its process.

Dead links

It is hard to build a Web site without internal broken links, and the message shown by Web servers when a page is not found, i.e.: when the Web server returns a 404 (not found) response, is considered by many Web site administrators as too annoying for users. Indeed, the default message looses the context of the Web site, so the Web site administrators of some Web sites prefer to build error pages that maintain visual and navigational consistency with the rest of their Web sites.

The problem is that in many cases the response for a page that does not exists is just a normal redirect to a custom-built error page, without the response header signaling the error condition. These pages are called “soft-404”. The indexing process could consider a redirect to a “soft-404” error page as a link between the URL in which the page was not found and the error page, and this can increase the score of the later. For a generic crawler it is very hard to mark “soft-404” pages, but in a topic crawler, it is expected that the relevance of these pages would be very low, so that they would be omitted in the result. Also their outgoing links would have

small priorities, and most probably do not even added to the crawl queue for further processing.

Robot Exclusion Protocol

"Robot Exclusion Protocol" provides a mechanism for Web server administrators to communicate their file access policies; more specifically to identify files that may not be accessed by a crawler. This is done by keeping a file named robots.txt under the root directory of the Web server (such as http://www.biz.uiowa.edu/robots.txt). This file provides access policy for different *User-agents* (robots or crawlers). A User-agent value of '*' denotes a default policy for any crawler that does not match other User-agent values in the file. A number of *Disallow* entries may be provided for a User-agent. Any URL that starts with the value of a Disallow field must not be retrieved by a crawler matching the User-agent. When a crawler wants to retrieve a page from a Web server, it must first fetch the appropriate robots.txt file and make sure that the URL to be fetched is not disallowed. More details on this exclusion protocol can be found at *http://www.robotstxt.org/wc/norobots.html*. It is efficient to cache the access policies of a number of servers recently visited by the crawler. This would avoid accessing a robots.txt file each time you need to fetch a URL. Although compliance with 'Robots Exclusion Protocol' is not mandatory and can be administratively overridden on the crawler, we have implemented in our study.

Crawler Implementation

In this study, we developed a Web crawler and implemented several algorithms to analyze their efficiencies in focused crawling. These algorithms differ only in assigning priorities to new extracted links.

```
Input: u1, u2, ..., un starting URLs, t topic
1: PQ = {u1, u2, ..., un}, priority queue of URLs to visit.
2: V = visited URLs, a = some constant
3: while PQ != empty && overall goal is not reached do
4:       Dequeue u from PQ, select u with highest priority.
5:       Fetch u as Page p.
6:       Add u into V.
7:       Parse p to extract text and extract outgoing links ux
8:       for each ux in p do
9:             if ux not in V then
10:                  if ux in PQ then
11:                        ux.cp = max(ux.cp, u.cp + a);
12:                  else
13:                        ux.cp = u.cp + a;
14:                        Add ux into PQ with cp;
15:                  end if
16:            end if
17:      end for
18: end while
```

Fig. 2. Depth First Algorithm

Depth First

Depth first is a general algorithm for traversing or searching a tree, tree structure, or graph. Starting from a seed, depth first search tries to crawl as deep as possible. Depth first is not an algorithm suitable for focused crawling, because it is

```
Input: u1, u2, ..., un starting URLs with cp = 0, t topic
1: PQ = {u1, u2, ..., un}, priority queue of URLs to visit.
2: V = visited URLs, a = some constant
3: while PQ != empty && overall goal is not reached do
4:       Dequeue u from PQ, select u with highest priority.
5:       Fetch u as Page p.
6:       Add u into V.
7:       Parse p to extract text and extract outgoing links ux
8:       for each ux in p do
9:             if ux not in V then
10:                  if ux in PQ then
11:                        ux.cp = max(ux.cp, u.cp - a);
12:                  else
13:                        ux.cp = u.cp - a;
14:                        Add ux into PQ with cp;
15:                  end if
16:            end if
17:      end for
18: end while
```

Fig. 3. Breadth First Algorithm

not interested neither with the content of the page visited nor with the URL it extracted (Figure 2).

Breadth First

Breadth first is also a simple tree traversing algorithm, but in contrast it is used to prove topical locality of pages in literature. It uses the priority queue as FIFO, and crawl the links in the order in which they are encountered. Whenever the crawl queue gets full, only one more link can be added for each page. Since both breadth first and depth first do not use any knowledge about the topic, we expect their performance to provide a lower bound for any of the more complex algorithms (Figure 3).

```
Input: u1, u2, ..., un starting URLs with cp = 0, t topic
1: PQ = {u1, u2, ..., un}, priority queue of URLs to visit.
2: V = visited URLs, a = some constant
3: while PQ != empty && overall goal is not reached  do
4:        Dequeue u from PQ, select u with highest priority.
5:        Fetch u as Page p.
6:        Add u into V.
7:        Parse p to extract text and extract outgoing links ux
8:        for each ux in p do
9:                 if ux not in V then
10:                        if ux in PQ then
11:                                ux.cp = ux.cp + a;
12:                        else
13:                                Add ux into PQ with cp = a;
14:                        end if
15:                end if
16:       end for
17: end while
```

Fig. 4. In Degree Algorithm

In Degree

Indegree algorithm is another graph based algorithm and each time a link is extracted from a page, the link is checked on the priority queue and if found its

priority is incremented. This is somewhat similar but much more basic version of PageRank algorithm (Figure 4).

```
Input: u1, u2, ..., un starting URLs with cp = 0, t topic
1: PQ = {u1, u2, ..., un}, priority queue of URLs to visit.
2: V = , visited URLs
3: while PQ != empty && overall goal is not reached do
4:      Dequeue u from PQ, select u with highest priority.
5:      Fetch u as Page p.
6:      Add u into V.
7:      Parse p to extract text and extract outgoing links ux
8:      for each ux in p do
9:          if ux not in V then
10:             if ux in PQ then
11:                 ux.cp = max(ux.cp, cos_sim (description, p));
12:             else
13:                 ux.cp = cos_sim (description, p);
14:                 Add ux into PQ with cp;
15:             end if
16:         end if
17:     end for
18: end while
```

Fig. 5. Best First Algorithm

Best First

In best first algorithm, the links are ordered according to some estimation criterion. Typically an initial representation of the topic, in our case words collected from top 10 pages returned from Google for the keyword, is used to guide the crawl. For each page downloaded lexical similarity is computed between a topic's keywords and the downloaded page. Every link extracted from this page have assigned this similarity value as their priority and then added to the queue (Figure 5).

Shark Search

In Shark search, crawlers search more extensively in areas of the Web in which relevant pages have been found. At the same time, the algorithm discontinues searches in regions that do not yield relevant pages. In addition, it is capable of giving every link on a page separate priority values, since the potential score of links

is influenced by anchor text, text surrounding the links, and the page content it is extracted from. (Figure 6)

```
Input: u1, u2, ..., un starting URLs with cp = 0 and depth d = max_d,  t topic
1: PQ = {u1, u2, ..., un}, priority queue of URLs to visit.
2: V = visited URLs,
3: max_d = depth, r = relative imporance,
3: while PQ != empty && overall goal is not reached do
4:      Dequeue u from PQ, select u with highest priority.
5:      Fetch u as Page p.
6:      Add u into V.
7:      if u.d > 0 then
8:          Parse p to extract text and extract outgoing links ux
9:          for each ux in p do
10:             if ux not in V then
11:                 if (cos_sim (description, p) > 0) then
12:                     ux.d = d;
13:                 else
14:                     ux.d = u.d – 1;
15:                 end if
16:                 if ux in PQ then
17:                     ux.cp = max(ux.cp, (1-r) * neighborhood_score (ux)
                                    + r * neighborhood_score (ux));
18:                 else
19:                     ux.cp = (1-r) * neighborhood_score (ux)
                                + r * neighborhood_score (ux);
20:                     Add ux into PQ with cp;
21:                 end if
22:             end if
23:         end for
24:     end if
25: end while
```

Fig. 6. Shark Search Algorithm

SharkSearchN / BestFirstN

SharkSearchN and BestFirstN algorithms differ from SharkSearch and BestFirst only by selecting the links from priority queue in batches, not one at a time. The reason is that good quality links are commonly encountered in early phases of the crawl, because if a link is first encountered at a later step, then it also means there are not many pages linking to it (Figure 7).

URL Combined

URL Combined algorithm is similar to Shark Search by assigning each link encountered a different priority, but also uses the URL tokens rather than the neighbor words of the link. For each outgoing link, page similarity, anchor text similarity and URL token similarity are used to compute priorities (Figure 8).

```
Input: u1, u2, ..., un starting URLs with cp = 0, t topic
1: PQ = {u1, u2, ..., un}, priority queue of URLs to visit.
2: V = visited URLs, N = size of batch queue, Vu = batch priority queue
3: while PQ != empty && overall goal is not reached  do
4:      Dequeue N times u from PQ, into Vu, select u in top N highest priority.
5:      for each u in Vu do
6 :             Fetch u as Page p.
7:              Add u into V.
8:              Parse p to extract text and extract outgoing links ux
9:              for each ux in p do
10:                     if ux in PQ then
11:                             ux.cp = max(ux.cp, cos_sim (description, p));
12:                     else if ux not in V then
13:                             ux.cp = cos_sim (description, p);
14:                             Add ux into PQ with cp;
15:                     end if
16:             end for
17:     end for
18: end while
```

Fig. 7. Best First N Algorithm

Algorithm Comparisons

Our goal in this study is to evaluate different algorithms in a focused crawler process. For this reason we have implemented all of the algorithms explained in this section. We used both Google and Open Directory (dmoz.org) for creating descriptions for the keywords. Keyword descriptions from Google were collected from top 10 pages returned from Google, whereas Open Directory descriptions were collected from the search page, since these pages contain descriptive human entries.

```
Input: u1, u2, ..., un starting URLs with cp = 0, t topic
1: PQ = {u1, u2, ..., un}, priority queue of URLs to visit.
2: V = visited URLs
3: while PQ != empty && overall goal is not reached  do
4:        Dequeue u from PQ, select u with highest priority.
5:        Fetch u as Page p.
6:        Add u into V.
7:        Parse p to extract text and extract outgoing links ux
8:        for each ux in p do
9:                 if ux not in V then
10:                        if ux in PQ then
11:                                ux.cp = cos_sim (description, p) * 0.5
                                   + cos_sim(description, anchor_text) * 0.25
                                   + cos_sim(description, url) * 0.25;

12:                        else
13:                                ux.cp = cos_sim (description, p) * 0.5
                                   + cos_sim(description, anchor_text) * 0.25
                                   + cos_sim(description, url) * 0.25;
14:                                Add ux into PQ with cp;
15:                        end if
16:                end if
17:       end for
18: end while
```

Fig. 8. URL Combined

In addition seed sets and target sets were created by querying Google. Seed sets differ from depth 1 to depth 3 backlink pages and target set is the top 100 pages. For each run 3000 pages are crawled and information on target set encounters, predicate satisfaction and lexical similarity between crawled pages and Google / DMOZ keyword descriptions.

Our target sets consist of top 100 links returned by the Google for the specified keyword. It is important to say that some of these top 100 links were *pdf*, or *ppt* files. Our crawler implementation does not follow or download *pdf* or *ppt* links but assign them priorities just like a page link and add to the queue. However, each time a link is dequeued it is checked on the target set. Whether the link will be further processed or not if it is in our target set, we count it as a successful encounter.

All algorithms are compared at which steps they could encounter target links, and with what percentage.

Predicate satisfaction refers to whether the keyword passes on a page. It is used to show whether the downloaded pages are related with the keyword. The page may not be a target page, and also it may not have a high similarity with the descriptions, but satisfy the predicate, i.e. we can say that a search engine returns this page as a result of the keyword query.

In this chapter, we give general characteristics and problems of crawlers, and explained our crawler implementation in detail. In the next chapter, Chapter 4, we present the results of our focused crawling algorithms

CHAPTER IV.

FOCUSED CRAWLING RESULTS

We have seen in Chapter 3 several crawling algorithms we have implemented for our performance evaluation, as well as the methods to compare the algorithms. For each visited page we collected information on the cosine similarity between its content and keyword description, whether the page satisfies our predicate and whether it is a target page. All of the algorithms have been run with the same keyword "artificial intelligence", description, seed and target pages. In addition, combined URL algorithm's performance checked with "data mining", as well.

In Table 1**,** the predicate satisfied number of pages and number of pages in target set which have been crawled are listed. For each algorithm, moving average of cosine similarity between page contents and Google and DMOZ descriptions are also presented in Figures 1-6.

According to Table 1, it is observable that the best performance in terms of predicate satisfaction and target set occurrence is in "Combined URL" algorithms. The main reason we can think of is that the descriptions created by Google and DMOZ also includes some domain names, such that the crawler assigns high priorities to the links with these domain names. In addition, we have observed that

for data mining keyword, most of the Web sites' URLs includes at least "data" or "mining" that have been visited.

Table 1. Predicate Satisfaction and Target Set Results

Algorithm	Predicate Satisfaction	Target Set
Breadth First	172	1
Depth First	60	0
InDegree	143	3
Best First	127	3
Best FirstN	179	9
SharkCrawler	158	3
SharkNCrawler	261	4
Combined URL – AI	264	9
Combined URL – DM	1283	18

The number of predicate satisfactory pages is lower than we had expected. About only 0.03% - 0.06% percentage of pages satisfy the predicates for the keyword "artificial intelligence". The depth and the distance of the seed set from the target pages can affect the numbers. We can say that depth 3 seed set collected from Google contains little information on "artificial intelligence". For "data mining" combined URL algorithm has found more than 40% satisfactory pages. First of all, the seed pages, and secondly, the overall topic and its general structure on the Web affect the percentage of satisfactory pages. This is also discussed in Chapter 6, "artificial intelligence" occurs ten times more than "data mining" on the Web (based on the Google estimated results). Although it may look like an advantage, weak connections over the graph may lead the crawler to undesirable paths.

We need also emphasize the number of pages found in target set for InDegree algorithm, since it is as good as a focused crawling algorithm. InDegree is a basic descendent of PageRank algorithm, and the pages in our target set are those pages with the highest PageRank values. Both PageRank and InDegree use similar logic when assigning scores to links. InDegree algorithm increments the importance of a

link by one whenever it finds the same link, whereas PageRank assigns the importance value according to the importance of the page itself.

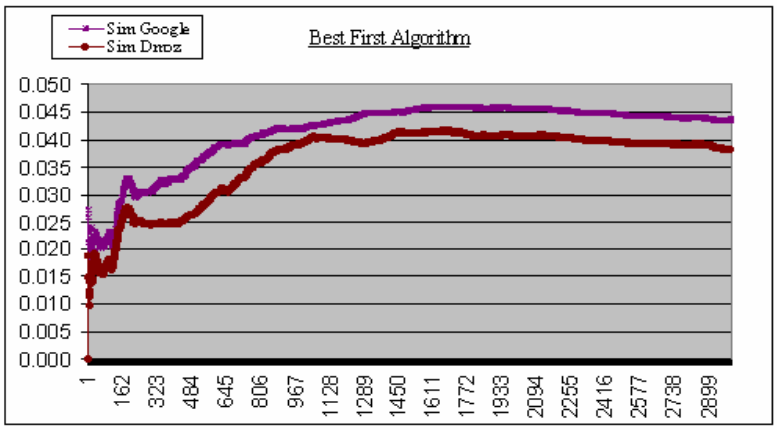

Fig. 9. Best First Algorithm – Cosine Similarity (moving average)

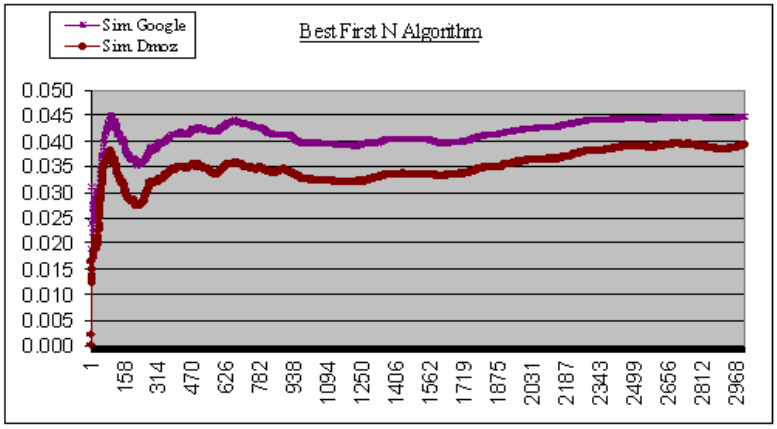

Fig. 10. Best First N Algorithm – Cosine Similarity (moving average)

Next, we compare the page similarity performances of the algorithms. It is important that during the crawling process, when assigning priorities to new extracted links, we computed cosine similarities with Google descriptions. DMOZ similarity only collected for informative purposes. Figure 9, Figure 10, Figure 11, and Figure 12 show performances of focused crawling algorithms, Best First, Best First N, Shark Search, Shark Search N, respectively. URL combined similarities for "data mining" and "artificial intelligence" is shown in Figure 13 and Figure 14, respectively. Lastly, performances of general tree traversing algorithms are shown.

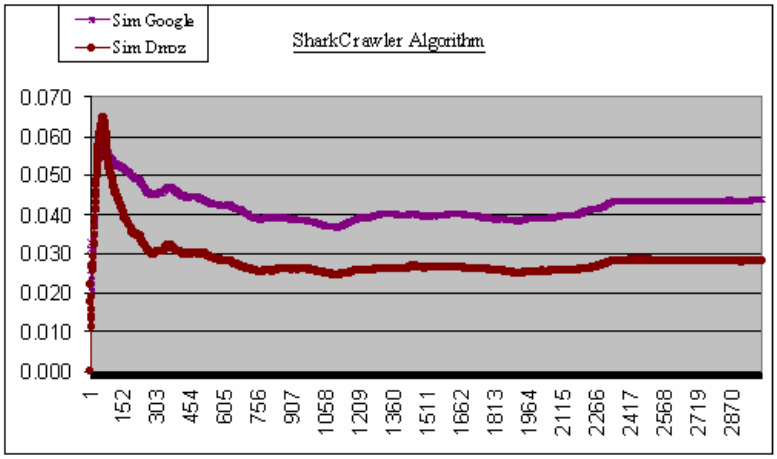

Fig. 11. Shark Crawler Algorithm – Cosine Similarity (moving average)

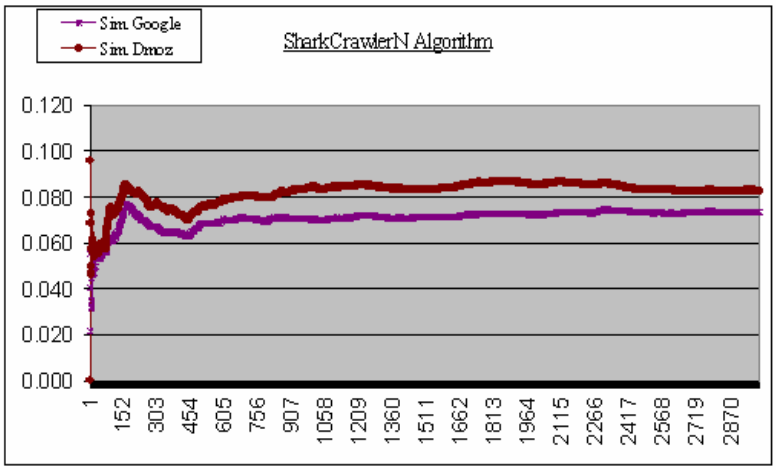

Fig. 12. Shark Crawler N Algorithm – Cosine Similarity (moving average)

Comparing the focused crawling algorithms, we can see the N batch queue algorithms have different behaviors, than their original versions. Overall page similarities are higher, and also at the beginning of the crawling process they crawl higher quality pages. This verifies the statement that the good quality pages are encountered at the early stages of the crawl. Other observation we make is that the SharkNCrawler, have higher similarity values than the BestNCrawler. The main reason is that the SharkNCrawler assign each link encountered different priorities according to their anchor text or words near the link.

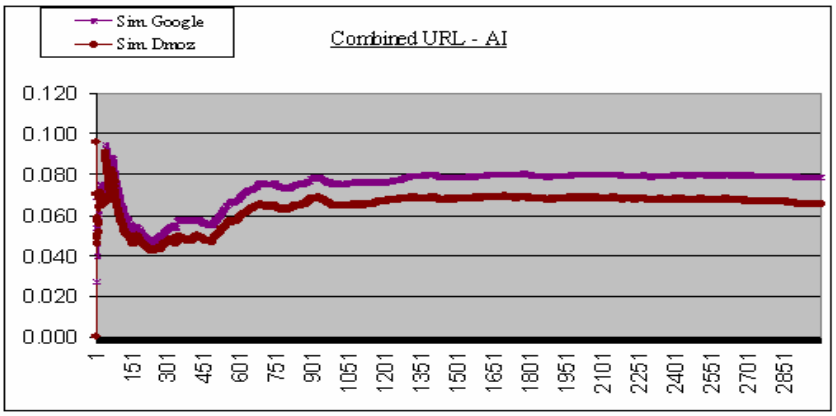

Fig. 13. Combined URL "Artificial Intelligence" – Cosine Similarity (moving average)

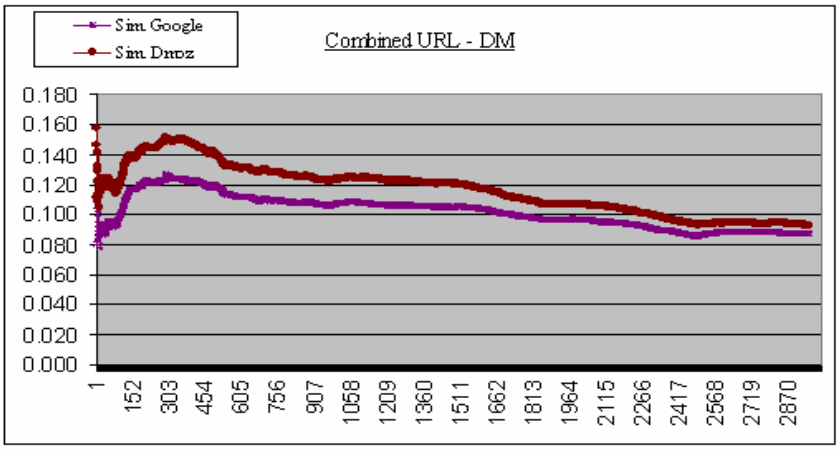

Fig. 14. Combined URL "Data Mining" – Cosine Similarity (moving average)

Combined URL algorithm has been run for "artificial intelligence" and "data mining" keywords. It is notable that all graphs on "artificial intelligence" start with a jump in few hundred pages and they fall back thereafter. However, as seen in Figure 14, this jump is not as sharp as it is in "artificial intelligence", and also similarity values are higher, reflecting the effect of the seed pages. In Figures 15-17, results of simple tree traversing algorithms are shown.

In this chapter we have presented the performance evaluations we have found during our crawler process. Most of the strategies tested were able to download important pages first. Generally, even a random strategy can perform well on the Web, i.e. a random walk on the Web is biased towards pages with high PageRank.

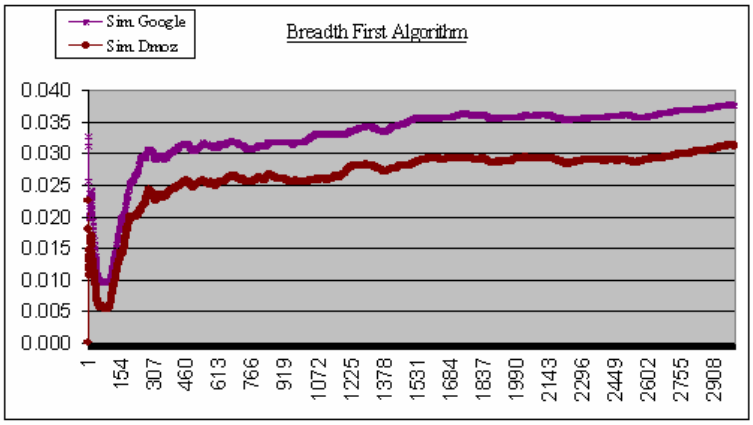

Fig. 15. Breadth First Algorithm – Cosine Similarity (moving average)

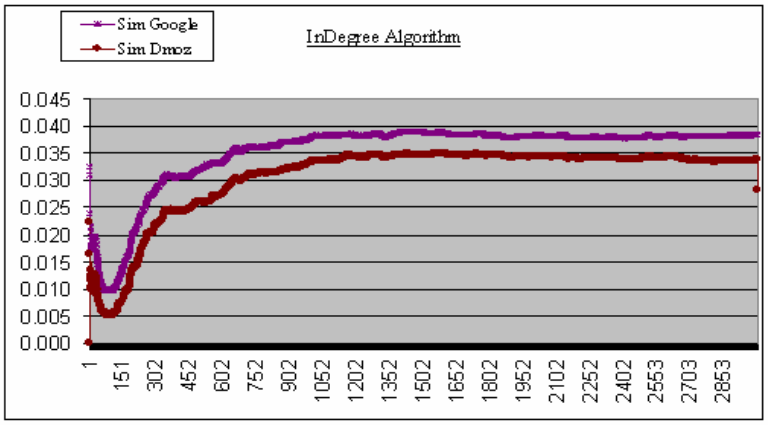

Fig. 16. InDegree Algorithm – Cosine Similarity (moving average)

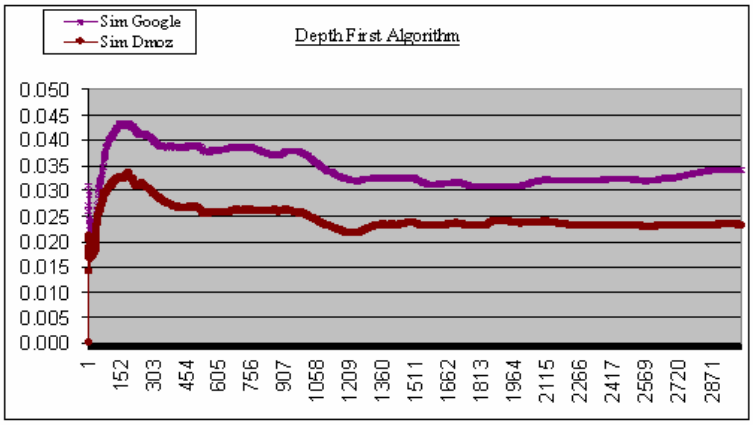

Fig. 17. Depth First Algorithm – Cosine Similarity (moving average)

All of the algorithms have been run several times to find comparable results, especially when the crawler encountered dead ends. The main reason of dead ends was that we have limited 100 pages to fetch from one domain to prevent spider traps. However, sometimes during our crawls the queue become full with the same domain

URLs, and to prevent falling into a spider trap all of the URLs skipped, resulting in a dead end. Although increasing the number of pages to download from the same domain may prevent dead ends, our goal was to visit as many different domains as possible, and do not want get in to a circle in a site, if even it is not a spider trap.

In the next chapter we present our automatic topic finding methodology using web pages.

CHAPTER V.

TOPIC FINDING METHODOLOGY

The motivation behind sub-topic or topic specific keyword discovery through Web pages is helping a user, who is insufficient in knowledge and experience about a topic, to find important concepts without much effort. Intuitively, a Web user would start searching the Web via querying search engines, and visiting some of the returned pages, reading and scanning, spending a lot of time on deciding what is important about the topic and what is not. In this study, we aim to develop an easy and systematic way of extracting key concepts from Web pages.

In general, when trying to extract information from Web pages automatically, one faces several problems. First of all, one of the most important limitations of automatic discovery of information through Web is that a portion of the Web, called as “the hidden Web” or the “deep Web”, can not be accessed by automated agents such as the crawlers used by search engines. This portion of the Web mainly consists of pages that require previous registration or some special authorization such as a password, or are only available when visited from within a certain network, such as a corporate intranet. In addition, there are also dynamic pages, which are only generated after a request has been made, and are inaccessible without certain parameters as input, e.g. query terms. With the above limitations all of the crawling and indexing of the search engines are done whether on static pages, or on dynamic

pages which do not require unknown parameters, or has an ingoing link with correct parameters.

Another problem is that Web content resides in a variety of formats, not just HTML, - e.g. text document formats such as *.pdf, .ps, .doc* or images*, .gif, .jpg*. An ideal Web crawler would parse all type of text documents successfully (even would process images); however, perfect parsing is not sufficient for extracting useful information from a set of documents. That is, only parsing a few Web pages may not provide sufficient information to a user, because these pages may not contain all the required information, in our case key concepts and/or sub-topics of the topic. For example, a good descriptive page about a topic may not include information about sub-topics, or the page might have not been created by the area experts, or in-depth researchers, and may only include unsatisfactory information, if even not any. In addition, the authors of the page may be only interested in only a small area of the topic.

Because of the above considerations searching only a few pages may not give satisfactory results to the users. There is need to search a lot of pages, which has to be related to the subject, has to be an informatory page, and contain subtopics or key concepts. Existing search engines are extremely useful for finding top ranked pages, containing related information about a topic. However, for a user, trying to extract information through all pages means, visiting each page, scanning if not reading the page, and get the basic idea, concepts, and key topics of the topic.

In this study, we try to mine important sub-topics or key concepts of a given topic automatically, through HTML based Web pages. Starting with a search query (a topic given by the user), the system gathers the set of top ranking pages returned from a search engine, and processes them further to discover sub-topics or keywords

of the search topic. Following that, the system filters those pages, which are unlikely to have sub-topic information, and identifies those informative pages, which may contain keywords, key concepts of the topic. These pages are processed further for extracting important phrases and then applied data mining techniques on these phrases to find candidate sub-topics. At the end, each candidate phrase are given scores based on the relevance with the input topic over the Web space, such that unrelated candidates can be filtered out in a post-processing step. Using the proposed technique, the user should be able to quickly learn sub-topics or key concepts about a topic without going through the ordeal of browsing through a large number of non-informative pages, e.g. commercial or promotional Web sites (which give little useful knowledge), returned by the search engine.

It is important to clarify that, although our main goal is finding sub-topics or important concepts of a topic, we do not use natural language processing techniques. The reason is that we also try to analyze tag information of Web pages, which the common text documents are lack of. HTML tags serve two general purposes on Web pages. First, they are used for designing the outline of Web pages, and second they are also used specifically to emphasize important information. The emphasized texts do not only highlight the important concepts to the reader, but also organize the information on a page. Consequently, both information and its format on the Web have great diversity.

There are several problems that need to be solved before working with Web pages:

- Web pages returned by the search engine may be published not for informative purposes, but for promotions, or commercials, etc. Only a small amount of pages contain definitions, descriptions or related academically

information about topics. Even in our case, only some of these descriptive, informative pages contain sub-topics or key concepts.

- Web pages are often very noisy; they typically contain many pieces of unrelated information. Thus, many unrelated text segments may be emphasized.
- Web page authors may emphasize those phrases or even long text segments that are not key concepts of the domain. For example, they tend to emphasize text segments that are related to their work or products, which may not be important sub-topics or key concepts of the domain.

To find those true sub-topics or key concepts of the domain, we need to deal with the above problems. Data mining techniques come to help naturally because they are able to find those frequent occurring word phrases, i.e., those phrases that appear in many pages. Thus, we can eliminate those peculiar ones that appear rarely. Those frequent word phrases form the candidate sub-topics or key concepts of the topic or the domain.

The proposed system consists of four main parts, namely “Data Gatherer”, “Data Extraction / Preprocessing”, “Data Processing”, and “Post Processing” which are explained in detail below:

Data Gatherer

Data Gatherer is responsible for collecting all necessary data to process. The system can work with data both from Internet and from a database which can be used for caching and fast retrieval purposes. The user specifies the main topic (keyword) to be searched for, decides whether a domain limitation would be applied, such as only .edu, or .com sites, when retrieving the list of pages, and the number of pages to be searched. In this study, we use Google (http:www.google.com/) as the search

engine to query Internet. Google provides GoogleWebApi[1] for research purposes. Using this API, the system collects links of top ranking pages from the search engine. Google API returns at most 10 links for every query, so that the system recursively requests more pages, until the user specified amount is retrieved.

In data gathering phase, if a page downloaded does not have HTML based information, i.e. the page does not contain <HTML> or <BODY> tags, or the page is a .pdf, or .ps formatted document or any error occurs during the preprocessing steps then that page is omitted, and a new page is requested from the search engine.

Data Extraction / Preprocessing:

Before starting to work with the pages downloaded, some filtering is applied. The pages, which generally do not contain sub-topics or key concepts, such as publication listing pages of researchers, forum discussion pages, university or departmental pages, do not need to be processed. For this reason, if any of the following phrases ("in proceeding", "journal", "next message", "previous message", "reply to") exists more than three times in any of the pages, than that page is considered to be a noisy document, and its contents are skipped entirely. The resulting documents serves as the source for sub-topic or key concept discovery.

If a page is not filtered out in result of above filtering conditions, then the page content needs to be parsed. The parsing of an HTML document is both extracting the tag information and also textual information. Web page authors use several HTML markup tags to emphasize important terms or concepts in their documents. Examples of these emphasizing tags include:

<h1>,<h2>,<h3>,<h4>,<b>,,<big>,<i>,,<u>,<li>,<dt>.

[1] For detailed information, http://www.google.com/apis/.

We suggest that among these tags, the most important tag for our task is "list" tag, because HTML supports changing the visualization of header tags (<h1>,...,<h4>), or font tags (<b>, <i>) using cascading style sheets. However, list tags are used to organize the structure of the Web page, not only for visualization purposes. Also, we propose that, if a page contain in-depth information about a topic, it also contains a list of important topics in anywhere of the page.

As a last step in data preprocessing; the extracting of list items is not sufficient to start processing. We also ignore some of the list items that can not be sub-topic or key concepts of a topic. A list item is removed if it contains:

- A salutation title (e.g., Mr., Dr, Professor)
- An URL or an email address
- Terms related to a publication (conference, proceedings, journal)
- Digits (e.g., WWW10, KDD2003, SIGMOD99)
- Too lengthy (that is it is unlikely to describe a sub-topic), we use 15 words as the upper limit for a useful list item.

Data Processing

We perform two different analyses on list items. Firstly, considering that we are looking for the sub-topics or key concepts, they are complete phrases and accepted in all of the topic area as they are. So, we may not need to apply stemming to the words, and also each list item indicates may be a candidate for our sub-topic. We collect list items from Web pages and check their overall frequency with our threshold, and those who pass the threshold represent our candidates.

Secondly, we split all the list item phrases into words, and perform stopwords removal and word stemming, and at the end obtain a word list. Stopwords are words that occur too frequently in documents and have little informational meanings.

Stemming finds the root form of a word by removing its suffix. We use Porter's algorithm for stemming.

Mine frequent occurring phrases: Each piece of texts extracted in data preprocessing step is stored in a dataset called a *transaction* set. We then apply an association rule mining algorithm (which is based on the Apriori algorithm) to find frequent itemsets.

When using each word separately, than our itemsets can be defined as a set of words those occur together in list items. In both ways, we apply Apriori algorithm to our itemsets, and also analyze the effects of the number of pages, or site domain restrictions, etc, to our findings.

The Apriori algorithm works in two steps. In the first step, it finds all *frequent itemsets* 2 from a set of *transactions* that satisfy a user-specified *minimum support*. In the second step, it generates rules from the discovered frequent itemsets. For our task, we only need the first step, i.e., finding frequent itemsets, which are candidate sub-topics. In addition, we only need to find frequent itemsets with three words or fewer in this work as we believe that a salient concept contains no more than three words.

Post Processing

After applying Apriori algorithm, we perform some post-processing steps to clear out the unrelated / unnecessary findings and results.

We need to explain that when using separate words in itemsets, then another problem arises after applying association, namely the order of the words. The reason is that the Apriori algorithm handles each transaction and frequent itemset as a bag of items (or words) without the notion of sequence. To overcome this problem, we check our list items and get the most frequent usage as our order.

The list of candidate phrases are checked with the entire list items. The following heuristic is applied thereafter: If there is not any list item that is equal to the candidate phrase, then this candidate phrase is unlikely to be a main sub-topic and it is thus removed. This heuristic is obvious because if the words in the itemset always appear with some other words together as list items, it is unlikely to be an important concept.

Another step to eliminate itemsets that are unlikely to be sub-topics, we perform a scoring rule to the itemsets. Unlike (Liu, 2003), who removed some generic frequent words heuristically such as, “abstract”, “introduction”, “summary”, “acknowledgement”, “conclusion”, “references”, “projects”, and “research”, we try to give topic-relevancy scores to candidate sub-topics and list them in order to the user.

The scoring process works as follows; using Google Web API, we collect the number of estimated pages containing the keyword, the candidate phrases, and both. If the number of pages containing a candidate phrase is more than the number of pages containing the keyword, then these candidate phrases are removed. By this way, we systematically remove frequent, generic words used in Internet from our candidate list. Moreover, by default Google Web API also throws exceptions if a search query is a common word used in Web pages. In results chapter, we explain the scoring and candidate elimination process more in detail.

Additionally, in this study, we build HTML tag tree for the pages containing any of the resulting candidate phrases, so that we are able to give some more emphasis on those phrases whose parent nodes includes the query keyword. Basically, an HTML tag tree consists of a root <html> tag, and different tags and text as tree nodes. However, many Web pages contain ill-structured HTML tags, of

which the problem is explained in detail on our previous methodology. The process of converting a "dirty" HTML document into a well-formed one is called tidying an HTML page[2]. It includes insertion of missing tags and reordering of tags in the "dirty" page. In our study, we use HTML tag tree just another reference for emphasizing some of the candidate phrases.

[2] http://www.w3.org/People/Raggett/tidy/ and http://tidy.sourceforge.net

CHAPTER VI.

TOPIC FINDING RESULTS

In this study, our main goal is finding important keywords or sub-topics related to a given query. We experimented our proposed model with different queries, different number of pages, different domains and also with and without stemming implementations. The queries used in the experiments are selected from both traditional and new emerging computer science related topics, such as “artificial intelligence”, “bioinformatics”, “data mining”, “Web mining” and “computer vision”.

All query words are experimented in 100 pages without domain limitations, stemming implementations, and in word based associations. In addition, to analyze the effects of variables such as domain selection, stemming and word / phrase based associations, “artificial intelligence” is selected as our base case. Table 2 shows our test cases in detail. Appendix A consists of the candidate phrases founded for “artificial intelligence” with different parameters.

Both as a candidate removal post-processing step and as a way to order the candidates in order of importance, we compute different relevance scores (*S*) using number of pages returned from search engine for keyword query (*k*), for candidate phrase (*CP*), and both combined (*b*). Using these three numbers, we computed six

different relevance scores shown in Table 3. S_5 and S_6 computed as the together occurrence probabilities of phrases. In $S_{1,}$ "*sqrt (n(CP) * n(k))*", and in $S_{3,}$ "$n(b)^2$", is used to eliminate the effects of units, and compared to the equation S_2 the computed scores were more reliable.

Table 2. Topic Finding Test Cases

Table No	Keyword	Word/Phrase	Stemming	# of Pages	Domain
4	Artificial Intelligence	W	N	100	all
5	Bioinformatics	W	N	100	all
6	Artificial intelligence	W	N	100	.com
7	Artificial intelligence	W	N	100	.edu
8	Data mining	W	N	100	all
9	Web mining	W	N	100	all
10	Computer vision	W	N	100	all
11	Artificial intelligence	W	N	200	all
12	Artificial intelligence	W	N	50	all
13	Artificial intelligence	W	Y	100	all
14	Artificial intelligence	P	N	100	all
15	Data mining	P	N	100	all

In Table 4, all candidate phrases for artificial intelligence collected from 100 pages without stemming implementations and using word based associations are listed. This list is divided into three parts. Type III candidates are those phrases which occur more than query keyword and thus eliminated at first step. Google does not even return an estimated number of pages for the candidates at the bottom of the list. Since we are looking for key concepts or related sub-topics, we do not want to include more general domains into our results. For example, robotics is a highly related concept with artificial intelligence (relevance scores are also high), but we can say that it is a more general research area than the AI.

Table 3. Scoring Measures CP (# candidate phrase), k (# keyword), b (# both)

$S_1 = n(b) / \text{sqrt}(n(CP) * n(k))$	(2)
$S_2 = n(b) / n(CP) * n(k)$	(3)
$S_3 = n(b)^2 / n(CP) * n(k)$	(4)
$S_4 = n(b) / \max(n(k), n(CP))$	(5)
$S_5 = n(CP) / n(k)$	(6)
$S_6 = n(b) / n(CP)$	(7)

The difference between the Type I and Type II phrases are determined by the relevance scores. After our experiments are completed, we analyzed the relevancy scores and concluded that using a 0.005 threshold score in S_1 would give results with high precision. For Type II candidates we can say that, most of them are not widely accepted as a relevant topic, and thus their relevance scores are low. Although "example systems" may be a related topic, our selection system removes it from our result set. Type I candidates are high occurring phrases, especially related to the keyword. These Type I candidates are listed in relevance score S_1 descending order. According to our results in Table 4, a total of 14 candidates are available, and machine learning is the most related and used topic in artificial intelligence with the given experimental conditions. All of these 14 Type I candidates are related concepts of "artificial intelligence". For Type I candidate phrases, we observe that they have high precision. However, we can not know all concepts related to "artificial intelligence", so that we are not able to comment on the recall of these results, but in our opinion, 10 or more Type I candidate phrases of a topic would be satisfactory for our study.

Table 4. Artificial intelligence – Word based – w/o stemming - 100 pages / n(k) = 76,700,000

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	machine learning *	16,300,000	6,670,000	0.1886402441	0.0000000053	0.0355851417	0.0869621904	0.0869621904	0.4092024540
	knowledge representation *	8,590,000	3,080,000	0.1199931495	0.0000000047	0.0143983559	0.0401564537	0.0401564537	0.3585564610
	neural networks *	19,000,000	3,880,000	0.1016382651	0.0000000027	0.0103303369	0.0505867014	0.0505867014	0.2042105263
	İnference	43,000,000	5,170,000	0.0900240679	0.0000000016	0.0081043328	0.0674054759	0.0674054759	0.1202325581
	natural language processing *	9,450,000	2,240,000	0.0832021276	0.0000000031	0.0069225940	0.0292046936	0.0292046936	0.2370370370
	fuzzy logic	5,280,000	956,000	0.0475054295	0.0000000024	0.0022567658	0.0124641460	0.0124641460	0.1810606061
	Cybernetics *	9,460,000	1,270,000	0.0471476956	0.0000000018	0.0022229052	0.0165580183	0.0165580183	0.1342494715
	artificial life	3,160,000	700,000	0.0449631591	0.0000000029	0.0020216857	0.0091264668	0.0091264668	0.2215189873
	turing test	925,000	363,000	0.0430961106	0.0000000051	0.0018572747	0.0047327249	0.0047327249	0.3924324324
	speech recognition *	14,800,000	1,140,000	0.0338357893	0.0000000010	0.0011448606	0.0148631030	0.0148631030	0.0770270270
	data mining *	54,900,000	2,190,000	0.0337489496	0.0000000005	0.0011389916	0.0285528031	0.0285528031	0.0398907104
	Prolog	19,000,000	1,050,000	0.0275052006	0.0000000007	0.0007565361	0.0136897001	0.0136897001	0.0552631579
	fuzzy set theory	323,000	81,500	0.0163741463	0.0000000033	0.0002681127	0.0010625815	0.0010625815	0.2523219814
	procedural knowledge	325,000	60,700	0.0121576414	0.0000000024	0.0001478082	0.0007913950	0.0007913950	0.1867692308
TYPE II	example systems	165,000	13,800	0.0038791755	0.0000000011	0.0000150480	0.0001799218	0.0001799218	0.0836363636
	formatting instructions	269,000	16,200	0.0035664915	0.0000000008	0.0000127199	0.0002112125	0.0002112125	0.0602230483
	neural networks faq	639	345	0.0015583715	0.0000000070	0.0000024285	0.0000044980	0.0000044980	0.5399061033
	fuzzy logic faq	305	166	0.0010853267	0.0000000071	0.0000011779	0.0000021643	0.0000021643	0.5442622951
	artificial life faq	1,520	191	0.0005593890	0.0000000016	0.0000003129	0.0000024902	0.0000024902	0.1256578947
	remaining topics	46,000	438	0.0002331833	0.0000000001	0.0000000544	0.0000057106	0.0000057106	0.0095217391
TYPE III	Automation *	258,000,000	6,820,000	0.0484815883	0.0000000003	0.0023504644	0.0264341085	0.0889178618	0.0264341085
	Bibliography	214,000,000	1,670,000	0.0130350260	0.0000000001	0.0001699119	0.0078037383	0.0217731421	0.0078037383
	Courses	1,030,000,000	5,920,000	0.0210622815	0.0000000001	0.0004436197	0.0057475728	0.0771838331	0.0057475728
	Dates	764,000,000	1,640,000	0.0067748469	0.0000000000	0.0000458986	0.0021465969	0.0213820078	0.0021465969
	Discussion	1,910,000,000	16,800,000	0.0438929647	0.0000000001	0.0019265924	0.0087958115	0.2190352021	0.0087958115
	Exercises	151,000,000	718,000	0.0066717273	0.0000000001	0.0000445119	0.0047549669	0.0093611473	0.0047549669
	Generation	926,000,000	14,500,000	0.0544082587	0.0000000002	0.0029602586	0.0156587473	0.1890482399	0.0156587473

	Groups	1,850,000,000	9,170,000	0.0243436553	0.0000000001	0.0005926136	0.0049567568	0.1195567145	0.0049567568
	Materials	1,690,000,000	11,000,000	0.0305528446	0.0000000001	0.0009334763	0.0065088757	0.1434159061	0.0065088757
	Membership	1,010,000,000	1,280,000	0.0045988750	0.0000000000	0.0000211497	0.0012673267	0.0166883963	0.0012673267
	Philosophy	491,000,000	7,500,000	0.0386476272	0.0000000002	0.0014936391	0.0152749491	0.0977835724	0.0152749491
	Robotics	80,300,000	6,390,000	0.0814226820	0.0000000010	0.0066296531	0.0795765878	0.0833116037	0.0795765878
	Sponsors	601,000,000	870,000	0.0040521390	0.0000000000	0.0000164198	0.0014475874	0.0113428944	0.0014475874
	Summary	1,150,000,000	5,180,000	0.0174414729	0.0000000001	0.0003042050	0.0045043478	0.0675358540	0.0045043478
	Workshops	530,000,000	1,340,000	0.0066461354	0.0000000000	0.0000441711	0.0025283019	0.0174706649	0.0025283019
	Article	---							
	Publications	---							
	Print	---							
	Resources	---							
	Help	---							
	History	---							
	Home	---							
	Links	---							
	Search	---							
	Software	---							
	News	---							
	Code	---							
	Events	---							

It is observable in Table 4**,** that some related concepts are not considered even as a Type II concept, such as “robotics”, and “philosophy”, although they are highly related to artificial intelligence in many sub domains. However, considering our main goal is to find sub-topics or sub-concepts, it does not pose any problem. We can just imagine them as being more general concepts, having artificial intelligence as a sub-domain.

In Table 5, Table 6, Table 7, and Table 8, Type I and Type II results for keywords, “bioinformatics”, “data mining”, “Web mining” and “computer vision” is given respectively. Those candidate phrase lists have similar characteristics. First of all it is notable that number of Type I and Type II phrases for “bioinformatics”, “Web mining” and “data mining” keywords are very low. There are two main reasons we can think of. First, the keyword domain is quite small compared to artificial intelligence. Hence, some of the related concepts are easily eliminated by the first removal step as being a more general concept. Secondly, the pages retrieved by the system may not contain many informative lists, and those who have lists may not contain common concepts. If we would increase the number of pages to crawl than more candidates could be found; we can see that candidates from 200 pages for AI in Table 11, is about two times more than the candidates from 100 pages, Table 4.

In addition, “data mining” Type II results ‘zerosum game theory’ display another fact. Because of the parser implementation in our study, the ‘-‘ character is removed from candidates and thus, search engine results only 2 pages for our candidate.

Table 5. Bioinformatics – Word – w/o stemming - no domain limitations / n(k) = 92,600,000

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	sequence analysis *	15,000,000	5,340,000	0.1432815049	0.0000000038	0.0205295896	0.0576673866	0.0576673866	0.3560000000
	Genomes	27,900,000	2,220,000	0.0436762536	0.0000000009	0.0019076151	0.0239740821	0.0239740821	0.0795698925
	Motif *	40,000,000	1,170,000	0.0192242928	0.0000000003	0.0003695734	0.0126349892	0.0126349892	0.0292500000
	Genome databases	283,000	82,800	0.0161745284	0.0000000032	0.0002616154	0.0008941685	0.0008941685	0.2925795053
	Matlab	30,800,000	318,000	0.0059545133	0.0000000001	0.0000354562	0.0034341253	0.0034341253	0.0103246753
	biomolecular modeling	29,900	9,430	0.0056672270	0.0000000034	0.0000321175	0.0001018359	0.0001018359	0.3153846154
TYPE II	gcg help page	23	11	0.0002383545	0.0000000052	0.0000000568	0.0000001188	0.0000001188	0.4782608696
	new technology search	76,200	130	0.0000489396	0.0000000000	0.0000000024	0.0000014039	0.0000014039	0.0017060367
	safari books online	399,000	920	0.0001513547	0.0000000000	0.0000000229	0.0000099352	0.0000099352	0.0023057644
	biotechnology information resources	422	345	0.0017452489	0.0000000088	0.0000030459	0.0000037257	0.0000037257	0.8175355450
	Networ	1,360,000	649	0.0000578322	0.0000000000	0.0000000033	0.0000070086	0.0000070086	0.0004772059
	academic solutions	2,600,000	580	0.0000373797	0.0000000000	0.0000000014	0.0000062635	0.0000062635	0.0002230769
	home networking	15,400,000	16,700	0.0004422325	0.0000000000	0.0000001956	0.0001803456	0.0001803456	0.0010844156
	resources home	7,100,000	78,700	0.0030693051	0.0000000001	0.0000094206	0.0008498920	0.0008498920	0.0110845070
	dsc form	263	23	0.0001473820	0.0000000009	0.0000000217	0.0000002484	0.0000002484	0.0874524715
	ncbi repository *	283	206	0.0012725314	0.0000000079	0.0000016193	0.0000022246	0.0000022246	0.7279151943
	genbank overview	710	155	0.0006045010	0.0000000024	0.0000003654	0.0000016739	0.0000016739	0.2183098592
	account request	1,060,000	817	0.0000824639	0.0000000000	0.0000000068	0.0000088229	0.0000088229	0.0007707547

Table 6. Data Mining – Word – w/o stemming – no domain limitations / n(k) = 61,000,000

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	Clustering	49,000,000	4,340,000	0.0793828656	0.0000000015	0.0063016393	0.0711475410	0.0711475410	0.0885714286
	neural networks *	19,700,000	1,370,000	0.0395205097	0.0000000011	0.0015618707	0.0224590164	0.0224590164	0.0695431472
	Datamining	2,510,000	327,000	0.0264268850	0.0000000021	0.0006983802	0.0053606557	0.0053606557	0.1302788845
	bayesian networks *	1,390,000	185,000	0.0200909009	0.0000000022	0.0004036443	0.0030327869	0.0030327869	0.1330935252
	search algorithms	1,900,000	71,400	0.0066321829	0.0000000006	0.0000439858	0.0011704918	0.0011704918	0.0375789474
TYPE II	nonlinear regression methods *	776	87	0.0003998743	0.0000000018	0.0000001599	0.0000014262	0.0000014262	0.1121134021
	zerosum game theory	2	0	0.0000000000	0.0000000000	0.0000000000	0.0000000000	0.0000000000	0.0000000000
	collection development	11,600,000	23,200	0.0008721558	0.0000000000	0.0000007607	0.0003803279	0.0003803279	0.0020000000
	Gaussians	793,000	26,100	0.0037526581	0.0000000005	0.0000140824	0.0004278689	0.0004278689	0.0329129887
	Crossvalidation	56,500	839	0.0004519317	0.0000000002	0.0000002042	0.0000137541	0.0000137541	0.0148495575

Table 7. Web mining – Word – w/o stemming – no domain limitations / n(k) = 708,000

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	Web usage mining *	132,000	49700	0.1625746375	0.0000005318	0.0264305128	0.0701977401	0.0701977401	0.3765151515
	user profiling	208,000	12900	0.0336156423	0.0000000876	0.0011300114	0.0182203390	0.0182203390	0.0620192308
	Web information extraction *	28,300	769	0.0054327096	0.0000000384	0.0000295143	0.0010861582	0.0010861582	0.0271731449
TYPE II	bamshad mobasher usa	12	12	0.0041169348	0.0000014124	0.0000169492	0.0000169492	0.0000169492	1.0000000000
	intelligent agent links	134	6	0.0006160021	0.0000000632	0.0000003795	0.0000084746	0.0000084746	0.0447761194
	traditional data mining	14,600	241	0.0023704128	0.0000000233	0.0000056189	0.0003403955	0.0003403955	0.0165068493
	selected pdfs	606,000	90	0.0001374009	0.0000000002	0.0000000189	0.0001271186	0.0001271186	0.0001485149
	multimedia elements	676,000	72	0.0001040741	0.0000000002	0.0000000108	0.0001016949	0.0001016949	0.0001065089
	xml pages	231,000	135	0.0003338191	0.0000000008	0.0000001114	0.0001906780	0.0001906780	0.0005844156

Table 8. Computer vision – Word – w/o stemming – no domain limitations

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	object recognition *	2,390,000	532,000	0.0876904546	0.0000000145	0.0076896158	0.0345454545	0.0345454545	0.2225941423
	feature extraction	2,540,000	483,000	0.0772271330	0.0000000123	0.0059640301	0.0313636364	0.0313636364	0.1901574803
	machine vision	6,720,000	554,000	0.0544583862	0.0000000054	0.0029657158	0.0359740260	0.0359740260	0.0824404762
	medical image analysis	281,000	106,000	0.0509556499	0.0000000245	0.0025964783	0.0068831169	0.0068831169	0.3772241993
	digital image processing *	2,180,000	246,000	0.0424567259	0.0000000073	0.0018025736	0.0159740260	0.0159740260	0.1128440367
	random sample consensus	29,500	22,300	0.0330851795	0.0000000491	0.0010946291	0.0014480519	0.0014480519	0.7559322034
	geometric hashing	52,400	28,900	0.0321715391	0.0000000358	0.0010350079	0.0018766234	0.0018766234	0.5515267176
	Egomotion	57,200	30,000	0.0319641294	0.0000000341	0.0010217056	0.0019480519	0.0019480519	0.5244755245
	particle filtering	126,000	35,100	0.0251977338	0.0000000181	0.0006349258	0.0022792208	0.0022792208	0.2785714286
	scene interpretation	67,500	24,200	0.0237357587	0.0000000233	0.0005633862	0.0015714286	0.0015714286	0.3585185185
	Preprocessing	6,180,000	222,000	0.0227561284	0.0000000023	0.0005178414	0.0144155844	0.0144155844	0.0359223301
	Convolution	6,520,000	219,000	0.0218554602	0.0000000022	0.0004776611	0.0142207792	0.0142207792	0.0335889571
	hough transforms	30,800	15,000	0.0217798829	0.0000000316	0.0004743633	0.0009740260	0.0009740260	0.4870129870
	sobel operator	41,400	16,100	0.0201634733	0.0000000253	0.0004065657	0.0010454545	0.0010454545	0.3888888889
	image acquisition *	2,460,000	117,000	0.0190089421	0.0000000031	0.0003613399	0.0075974026	0.0075974026	0.0475609756
	morphological image processing	40,900	12,700	0.0160022742	0.0000000202	0.0002560728	0.0008246753	0.0008246753	0.3105134474
	multiresolution analysis	235,000	27,300	0.0143505437	0.0000000075	0.0002059381	0.0017727273	0.0017727273	0.1161702128
	fourier transform	10,700,000	171,000	0.0133212157	0.0000000010	0.0001774548	0.0111038961	0.0111038961	0.0159813084
	affective computing	161,000	13,700	0.0087005613	0.0000000055	0.0000756998	0.0008896104	0.0008896104	0.0850931677
	optical character recognition *	3,040,000	50,600	0.0073952586	0.0000000011	0.0000546898	0.0032857143	0.0032857143	0.0166447368
TYPE II	vision list digest	16,900	670	0.0013133216	0.0000000026	0.0000017248	0.0000435065	0.0000435065	0.0396449704
	extracting corner features	22	20	0.0010865715	0.0000000590	0.0000011806	0.0000012987	0.0000012987	0.9090909091
	medical image faq	708	95	0.0009098014	0.0000000087	0.0000008277	0.0000061688	0.0000061688	0.1341807910
	stereo matching notes	3	3	0.0004413674	0.0000000649	0.0000001948	0.0000001948	0.0000001948	1.0000000000
	detecting blob features	10	9	0.0007252407	0.0000000584	0.0000005260	0.0000005844	0.0000005844	0.9000000000
	second order methods	40,400	230	0.0002915928	0.0000000004	0.0000000850	0.0000149351	0.0000149351	0.0056930693
	mmvl mediawiki	18	15	0.0009009375	0.0000000541	0.0000008117	0.0000009740	0.0000009740	0.8333333333
	phd theses	1,150,000	15,300	0.0036356507	0.0000000009	0.0000132180	0.0009935065	0.0009935065	0.0133043478
	pixelwise thresholding	22	15	0.0008149286	0.0000000443	0.0000006641	0.0000009740	0.0000009740	0.6818181818

	create project	11,300,000	501	0.0000379785	0.0000000000	0.0000000014	0.0000325325	0.0000325325	0.0000443363
	general resources	6,150,000	1,060	0.0001089201	0.0000000000	0.0000000119	0.0000688312	0.0000688312	0.0001723577
	priority support	12,000,000	266	0.0000195673	0.0000000000	0.0000000004	0.0000172727	0.0000172727	0.0000221667
	masters theses	184,000	395	0.0002346539	0.0000000001	0.0000000551	0.0000256494	0.0000256494	0.0021467391

For “Bioinformatics” keyword, we have found 6 Type I candidate phrases and 12 Type II candidate phrases. “Sequence analysis” is found to be the most related concept with bioinformatics. In my opinion, using S_1 score with a threshold 0.005 eliminated unrelated concepts successfully. We think that the candidate “Matlab” shows that it is the most widely used application in “bioinformatics”, and thus come out to be a Type I candidate.

The Type I candidate phrases for “Data mining” and “Web mining” (shown in Table 6 and Table 7, respectively) are found to be related with their respective concepts. We can also see that some related phrases in Type II candidates, but with S_1 score threshold we were able to eliminate those candidates, such as “gaussians” and “crossvalidation”. Candidate phrase lists of these keywords also do not contain some very important concepts like “classification”, “association” for “data mining”, and “web structure mining”, “web content mining” for “web mining”. We can say that for some cases, although high precision is observed, the candidate phrases may have low recall. The main reason is that the web pages processed do not contain many list items.

“Computer vision” resulted in 20 Type I candidate phrases and 13 Type II candidate phrases. Again, our 0.005 threshold resulted in successful elimination unrelated candidates. In my opinion, all of Type I candidates are highly related to “computer vision”.

According to the Table 9 and Table 10 shown in Appendix A, *.edu* domain have quite different candidate phrases compared to the *.com* domain. While educational sites have information about agent related topics about AI, the commercial sites have more candidates in cybernetics, speech recognition or neural

networks. Most probably the commercial sites advertise either their tools for speech recognition security etc., or neural network application for knowledge discovery.

With the increase of the number of pages processed, the number of candidate phrases is also increased, because the proposed system uses the same support count for both cases. However, still the results are satisfactory and most of the candidates are highly relevant with artificial intelligence. The same reasoning applies when small number of pages is crawled. We can see that when only 50 pages processed the resulting list does not contain any Type II candidates.

In tables, starred (*) candidates are those list items, whose parent nodes include the keyword. That is, HTML tag trees are created for pages containing any of the candidate phrases, and on this tree it is checked whether any of the list's parent nodes have the queried keyword. Those candidates can be emphasized more for user notice. However, the ill-structured tags on pages make it difficult to create clean HTML tag trees, and even enforce to skip some.

In this chapter, we presented the results of our topic finding methodology. The relevance scoring used was very successful in eliminating unrelated concepts. These results show the methodology result in high precision, i.e. Type I of candidate phrases are found to be highly related to the keywords. However, only a few Type I candidate phrases could be found for some keywords, such as "Web mining", "bioinformatics" or "data mining".

In the last chapter, we conclude our results and give some general suggestions for future work, on both topic-finding methods and focused crawler evaluations.

CHAPTER VII.

CONCLUSIONS

In this thesis, we studied focused crawling algorithm performances and topic finding through Web pages. In the first part, we implemented several algorithms in the literature and compared their overall performances at different levels. In the second part, an automatic topic discovery methodology have developed and checked its consistency for different parameters. This section points out what we have done and what could be done as future work, for each of the parts separately.

We started by surveying related work to our thesis in Chapter 2, and then explained our two studies in detail. The first study dealt with focused crawling and we worked in a series of problems that appeared during the design and implementation of a Web crawler. In Chapter 3, we described Web crawling in the context of information retrieval, and the problems we have solved for the crawler process and given the algorithms we have implemented.

In Chapter 4, we presented our performance results and explained our observations and conclusions. For future work in focused crawling algorithm evaluation, we can observe the performance of adaptive or intelligent crawlers, or pre-trained algorithms. In addition, different keywords, different seed and target sets, network and CPU usage can be included into the test cases. In literature, several studies have been done on the comparison of short and long running crawls.

For topic crawling study we have developed and implemented an automatic discovery methodology. The results are presented in Chapter 6, and highly satisfactory in terms of their precision. For finding importance of candidate phrases we used Google's estimations of results, and computed several scores. Small domain areas did not result well in terms of number of candidates, but again with high precision. This study can be improved, by using a recursive approach for candidate finding, such that each candidate phrase's list can be queried on Google. That would improve the number of candidate phrases overall. In addition, rather than using limitation to number of pages, we can use number of lists to be processed, such that more candidate phrases can be evaluated.

There are also some general problems for Web mining, such as the high amount of information on the Web, but only few in quality information. Web search is difficult today and likely that Web crawling will continue to be a difficult problem, at least in the next years, and we expect several challenges. Multimedia information usage increases over the Web sites and number of Web posting blogs, forums will be larger than the number of informative Web pages, further reducing the signal-to-noise ratio of the Web. Finally, pages with semantic markup could become a significant fraction of Web pages, radically changing the problem of Web search.

BIBLIOGRAPHY

Aggarwal, C., Al-Garawi, F. & Yu, P. (2001). Intelligent Crawling on the World Wide Web with Arbitrary Predicates. *Proceedings of the 10th International Word Wide Web Conference*, Hong Kong, May 2001.

Amento, B., Terveen, L. & Hill, W. (2000). Does "authority" mean quality? Predicting expert quality ratings of Web documents. *Proceedings of the 23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

Ben-Shaul, I., Herscovici, M., Jacovi, M., Maarek, Y., Pelleg, D., Shtalhaim, M., Soroka, V., & Ur, S. (1999). 'Adding support for Dynamic and Focused Search with Fetuccino'. *Computer Networks* 31(11–16), 1653–1665.

Bharat, K., & Henzinger, M. (1998). 'Improved Algorithms for Topic Distillation in Hyperlinked Environments'. *Proc. 21st ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 104–111.

Brandman, O., Cho, J., Garcia-Molina, H., & Shivkumar, N. (2000). Crawler friendly Web Servers. *Proceedings of the Performance and Architecture of Web Servers Workshop*, Santa Clara, California , June 2000.

Borges, J., & Levene, M. (1999). Data mining of user navigation patterns. In *Proceedings of the WEBKDD '99 Workshop on Web Usage Analysis and User Profiling, August 15, 1999, San Diego, CA, USA*, pages 31-36.

Brin, S. & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30 (1-7): 107-117.

Broder, A. (2002). *A taxonomy of Web search.* SIGIR Forum, 36 (2):3-10, 2002.

Chakrabarti, S. (2003). *Mining the Web*, USA, Morgan Kaufmann.

Chakrabarti, S., Van den Berg, M., & Dom, B. (1999a). Focused crawling: A new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11-16):1623-1640.

Chakrabarti, S., Punera, K., & Subramanyam, M. (1999b). Accelerated focused crawling through online relevance feedback". *Proceedings of the 8th International WWW Conference, Toronto*, Canada, May. 1999

Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., & Kleinberg, J. (1998). 'Automatic resource compilation by analyzing hyperlink structure and associated text'. *Computer Networks* 30(1–7), 65–74.

Cho, J., Garcia-Molina, H., & Page, L. (1998). Efficient crawling throuh URL ordering. *Computer Networks*, 30(1-7):161-172, 1998.

Cooley, R., Mobasher, B., & Srivastava, J. (1997). Web mining: Information and pattern discovery on the world wide Web. *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97).*

De Bra, P.M.E., & Post , R.D.J. (1994). Information Retrieval in the World-Wide Web: Making Client-based searching feasible. *Proc. 1st International World Wide Web Conference*.

Diligenti, M., Coetzee, F., Lawrence, S., Giles, C., & Gori, M.. (2000). Focused Crawling using Context Graphs'. In *Proceedings of the 26th Internatioanl Conference on Very Large Databases (VLDB 2000), Cairo, Egypt*, September 2000.

Ehrig, M., & Maedche, A. (2003). Ontology-Focused Crawling of Web Documents. *Proceedings of the 2003 ACM symposium on Applied computing, Merlbourne, Florida.*, 2003.

Etzioni, O. (1996). The world wide Web: Quagmire or gold mine. *Communications of the ACM*, 39 (11): 65-68, 1996.

Hersovici, M., Jacovi, M., Maarek, Y.S., Pelleg, D., Shtalhaim, M., & Ur, S. (1998). The shark-search algorithm – An application : Tailored Web site mapping. *In Proceedings of the WWW7*, 1998.

Huang, L. (2000) *A survey on Web information retrieval technologies.* Tech. Rep., ECSL, 2000.

Henzinger, M., Motwani, R., & Siverstein, C. (2003). Challenges in Web search engines. *In Proc. 18th International Joint Conference on Artificial Intelligence*, (pp.1573-1579)

Kleinberg, J. (1998) Authoritative soruces in a hyperlinked environment. *Journal of the ACM* 46, 5 (November), 604-632.

Liu, B., Chin, C. W., & Ng, H. T. (2003). Mining Topic-Specific Concepts and Definitions on the Web, *In WWW 2003, May 20-24*, 2003, Budapest, Hungary.

Madria, S.K., Bhowmick, S.S., Ng, W.K., & Lim., E. P. (1999). Research issues in Web data mining. In *Proceedings of Data Warehousing and Knowledge Discovery, First International Conference, DaWaK '99*, pages 303-312, 1999.

Menczer, F., & Belew, R. (2000), 'Adaptive Retrieval Agents: Internalizing Local Context and Scaling up to the Web'. *Machine Learning 39(2–3), 203–242*.

Menczer, F., Pant, G., Srinivasan, P., & Ruiz, M. (2001). Evaluating topic-driven Web crawlers. *Proceedings of the 24th Annual International ACM/SIGIR Conference, New Orleans, USA*, 2001.

Menczer, F. (2002). Complementing search engines with online Web mining agents. *Decision Support Systems 35(2): 195-212*, 2002a.

Menczer, F., Pant, G., & Srnivasan, P. (2002). Topic-driven crawlers: machine learning issues. *ACM TOIT*, 2002b.

Najork, M., & Wiener, J.L. (2001) Breadth-first search crawling yields high-quality pages. *In Proc. 10th International World Wide Web Conference*, 2001.

Pant, G., Srnivasan, P., & Menczer, F. (2004). Crawling the Web. *Web Dynamics 2004* (pp.153-178)

Pinkerton, B. (1994). Finding what people want: Experiences with the Webcrawler. In *Proceedings of the First International World Wide Web Conference, Geneva, Switzerland*,.

Porter, M.F. (1980). An algorithm for suffix stripping. *Program.* 14(3) pp 130-137.

Rennie, J., & McCallum, A. (1999). 'Using reinforcement learning to spider the Web efficiently'. *In Proc. 16th International Conf. on Machine Learning*. pp. 335–343, Morgan Kaufmann, San Francisco, CA.

Srinivasan, P., Menczer, F., & Pant, G. (2005). A general evaluation framework for topical crawlers. *Information retrieval 8*(3): 417-447

Srinivasan, P., Mitchell, J., Bodenreider, O., Pant, G., & Menczer, F. Web crawling agents for retrieving biomedical information. In *Proc. Int. Workshop on Agents in Bioinformatics (NETTAB-02)*, 2002.

Tirri, H. (2003). Search in vain: challenges for internet search. *IEEE Computer* (January) (pp.115-116)

APPENDIX A

Table 9. Artificial intelligence – Word – w/o stemming – .com domain

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	machine learning *	16,300,000	6,670,000	0.1886402441	0.0000000053	0.0355851417	0.0869621904	0.0869621904	0.4092024540
	knowledge representation *	8,590,000	3,080,000	0.1199931495	0.0000000047	0.0143983559	0.0401564537	0.0401564537	0.3585564610
	neural networks *	19,000,000	3,880,000	0.1016382651	0.0000000027	0.0103303369	0.0505867014	0.0505867014	0.2042105263
	İnference	43,000,000	5,170,000	0.0900240679	0.0000000016	0.0081043328	0.0674054759	0.0674054759	0.1202325581
	natural language processing *	9,450,000	2,240,000	0.0832021276	0.0000000031	0.0069225940	0.0292046936	0.0292046936	0.2370370370
	fuzzy logic	5,280,000	956,000	0.0475054295	0.0000000024	0.0022567658	0.0124641460	0.0124641460	0.1810606061
	Cybernetics *	9,460,000	1,270,000	0.0471476956	0.0000000018	0.0022229052	0.0165580183	0.0165580183	0.1342494715
	artificial life	3,160,000	700,000	0.0449631591	0.0000000029	0.0020216857	0.0091264668	0.0091264668	0.2215189873
	speech recognition *	14,800,000	1,140,000	0.0338357893	0.0000000010	0.0011448606	0.0148631030	0.0148631030	0.0770270270
	data mining *	54,900,000	2,190,000	0.0337489496	0.0000000005	0.0011389916	0.0285528031	0.0285528031	0.0398907104
TYPE II	fuzzy logic faq	305	166	0.0010853267	0.0000000071	0.0000011779	0.0000021643	0.0000021643	0.5442622951
	neural networks faq	639	345	0.0015583715	0.0000000070	0.0000024285	0.0000044980	0.0000044980	0.5399061033
	artificial life faq	1,520	191	0.0005593890	0.0000000016	0.0000003129	0.0000024902	0.0000024902	0.1256578947
	remaining topics	46,000	438	0.0002331833	0.0000000001	0.0000000544	0.0000057106	0.0000057106	0.0095217391

Table 10. Artificial intelligence – Word – w/o stemming - .edu

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	machine learning *	16,300,000	6,670,000	0.1886402441	0.0000000053	0.0355851417	0.0869621904	0.0869621904	0.4092024540
	expert systems *	8,810,000	3,530,000	0.1357966540	0.0000000052	0.0184407312	0.0460234681	0.0460234681	0.4006810443
	natural language processing *	9,450,000	2,240,000	0.0832021276	0.0000000031	0.0069225940	0.0292046936	0.0292046936	0.2370370370
	multiagent systems	702,000	406,000	0.0553298886	0.0000000075	0.0030613966	0.0052933507	0.0052933507	0.5783475783
	virtual reality	37,200,000	2,720,000	0.0509213316	0.0000000010	0.0025929820	0.0354628422	0.0354628422	0.0731182796
	data mining *	54,900,000	2,190,000	0.0337489496	0.0000000005	0.0011389916	0.0285528031	0.0285528031	0.0398907104
	object recognition	2,610,000	259,000	0.0183055098	0.0000000013	0.0003350917	0.0033767927	0.0033767927	0.0992337165
	intelligent software agents	148,000	53,500	0.0158790765	0.0000000047	0.0002521451	0.0006975228	0.0006975228	0.3614864865
	distributed computing	32,400,000	661,000	0.0132596259	0.0000000003	0.0001758177	0.0086179922	0.0086179922	0.0204012346
	Chess	68,800,000	956,000	0.0131603061	0.0000000002	0.0001731937	0.0124641460	0.0124641460	0.0138953488
	information integration	11,100,000	192,000	0.0065802449	0.0000000002	0.0000432996	0.0025032595	0.0025032595	0.0172972973
	program committee	31,300,000	307,000	0.0062656845	0.0000000001	0.0000392588	0.0040026076	0.0040026076	0.0098083067
TYPE II	research overview	888,000	13,000	0.0015752126	0.0000000002	0.0000024813	0.0001694915	0.0001694915	0.0146396396
	Knowledgebased systems	12,400	597	0.0006121612	0.0000000006	0.0000003747	0.0000077836	0.0000077836	0.0481451613
	formatting instructions	303,000	16,200	0.0033604390	0.0000000007	0.0000112926	0.0002112125	0.0002112125	0.0534653465
	fuzzy logic faq	305	166	0.0010853267	0.0000000071	0.0000011779	0.0000021643	0.0000021643	0.5442622951
	neural networks faq	639	345	0.0015583715	0.0000000070	0.0000024285	0.0000044980	0.0000044980	0.5399061033
	java version	5,650,000	23,800	0.0011432864	0.0000000001	0.0000013071	0.0003102999	0.0003102999	0.0042123894
	biological motion	90,000	885	0.0003368405	0.0000000001	0.0000001135	0.0000115385	0.0000115385	0.0098333333
	intelligent transportation systems	2,650,000	61,400	0.0043067311	0.0000000003	0.0000185479	0.0008005215	0.0008005215	0.0231698113
	deepak kumar homepage	15	1	0.0000294820	0.0000000009	0.0000000009	0.0000000130	0.0000000130	0.0666666667
	lisa meeden homepage	8	4	0.0001614795	0.0000000065	0.0000000261	0.0000000522	0.0000000522	0.5000000000

Table 11. Artificial intelligence – Word – w/o stemming – 200 pages

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	machine learning *	16300000	6670000	0.1886402441	0.0000000053	0.0355851417	0.0869621904	0.0869621904	0.4092024540
	expert systems *	8810000	3530000	0.1357966540	0.0000000052	0.0184407312	0.0460234681	0.0460234681	0.4006810443
	neural networks *	19000000	3880000	0.1016382651	0.0000000027	0.0103303369	0.0505867014	0.0505867014	0.2042105263
	Aaai	4330000	1610000	0.0883454619	0.0000000048	0.0078049206	0.0209908735	0.0209908735	0.3718244804
	natural language processing *	9450000	2240000	0.0832021276	0.0000000031	0.0069225940	0.0292046936	0.0292046936	0.2370370370
	natural language	38700000	3730000	0.0684629593	0.0000000013	0.0046871768	0.0486310300	0.0486310300	0.0963824289
	fuzzy logic	5280000	956000	0.0475054295	0.0000000024	0.0022567658	0.0124641460	0.0124641460	0.1810606061
	image processing	48400000	2790000	0.0457913914	0.0000000008	0.0020968515	0.0363754889	0.0363754889	0.0576446281
	case based reasoning *	793000	355000	0.0455191431	0.0000000058	0.0020719924	0.0046284224	0.0046284224	0.4476670870
	artificial life	3160000	700000	0.0449631591	0.0000000029	0.0020216857	0.0091264668	0.0091264668	0.2215189873
	automated reasoning *	685000	314000	0.0433198237	0.0000000060	0.0018766071	0.0040938722	0.0040938722	0.4583941606
	cognitive science *	22800000	1810000	0.0432826128	0.0000000010	0.0018733846	0.0235984355	0.0235984355	0.0793859649
	turing test	925000	363000	0.0430961106	0.0000000051	0.0018572747	0.0047327249	0.0047327249	0.3924324324
	artificial neural networks	2250000	563000	0.0428567717	0.0000000033	0.0018367029	0.0073402868	0.0073402868	0.2502222222
	Ontology	43900000	2100000	0.0361900641	0.0000000006	0.0013097207	0.0273794003	0.0273794003	0.0478359909
	speech recognition *	14800000	1140000	0.0338357893	0.0000000010	0.0011448606	0.0148631030	0.0148631030	0.0770270270
	data mining *	54900000	2190000	0.0337489496	0.0000000005	0.0011389916	0.0285528031	0.0285528031	0.0398907104
	intelligent agent *	1040000	268000	0.0300068524	0.0000000034	0.0009004112	0.0034941330	0.0034941330	0.2576923077
	alan turing	1060000	171000	0.0189646790	0.0000000021	0.0003596590	0.0022294654	0.0022294654	0.1613207547
	fuzzy set theory	323000	81500	0.0163741463	0.0000000033	0.0002681127	0.0010625815	0.0010625815	0.2523219814
	procedural knowledge	325000	60700	0.0121576414	0.0000000024	0.0001478082	0.0007913950	0.0007913950	0.1867692308
	optical character recognition *	2090000	93200	0.0073611402	0.0000000006	0.0000541864	0.0012151239	0.0012151239	0.0445933014
	program committee	31300000	307000	0.0062656845	0.0000000001	0.0000392588	0.0040026076	0.0040026076	0.0098083067
	graph searching	46100	9910	0.0052701807	0.0000000028	0.0000277748	0.0001292047	0.0001292047	0.2149674620
TYPE II	intelligent transportation systems	2650000	61400	0.0043067311	0.0000000003	0.0000185479	0.0008005215	0.0008005215	0.0231698113
	fuzzy logic faq	305	166	0.0010853267	0.0000000071	0.0000011779	0.0000021643	0.0000021643	0.5442622951
	neural networks faq	639	345	0.0015583715	0.0000000070	0.0000024285	0.0000044980	0.0000044980	0.5399061033
	artificial life faq	1520	191	0.0005593890	0.0000000016	0.0000003129	0.0000024902	0.0000024902	0.1256578947

	example systems	165000	13800	0.0038791755	0.0000000011	0.0000150480	0.0001799218	0.0001799218	0.0836363636
	technical reports	67700000	269000	0.0037330195	0.0000000001	0.0000139354	0.0035071708	0.0035071708	0.0039734121
	formatting instructions	303000	16200	0.0033604390	0.0000000007	0.0000112926	0.0002112125	0.0002112125	0.0534653465
	course home	3930000	14200	0.0008178890	0.0000000000	0.0000006689	0.0001851369	0.0001851369	0.0036132316
	discussion group	38400000	84800	0.0015625451	0.0000000000	0.0000024415	0.0011056063	0.0011056063	0.0022083333
	Website game	43000	1520	0.0008369734	0.0000000005	0.0000007005	0.0000198175	0.0000198175	0.0353488372
	remaining topics	46000	438	0.0002331833	0.0000000001	0.0000000544	0.0000057106	0.0000057106	0.0095217391
	Syllabus	58300000	279000	0.0041722685	0.0000000001	0.0000174078	0.0036375489	0.0036375489	0.0047855918
	Webbased	1500000	11300	0.0010535014	0.0000000001	0.0000011099	0.0001473272	0.0001473272	0.0075333333
	İdo	77300000	57200	0.0007428628	0.0000000000	0.0000005518	0.0007399741	0.0007457627	0.0007399741

Table 12. Artificial intelligence – Word – w/o stemming – 50 pages

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	machine learning *	16,300,000	6,670,000	0.1886402441	0.0000000053	0.0355851417	0.0869621904	0.0869621904	0.4092024540
	neural networks *	19,000,000	3,880,000	0.1016382651	0.0000000027	0.0103303369	0.0505867014	0.0505867014	0.2042105263
	natural language processing *	9,450,000	2,240,000	0.0832021276	0.0000000031	0.0069225940	0.0292046936	0.0292046936	0.2370370370
	fuzzy logic	5,280,000	956,000	0.0475054295	0.0000000024	0.0022567658	0.0124641460	0.0124641460	0.1810606061
	artificial life	3,160,000	700,000	0.0449631591	0.0000000029	0.0020216857	0.0091264668	0.0091264668	0.2215189873
	turing test	925,000	363,000	0.0430961106	0.0000000051	0.0018572747	0.0047327249	0.0047327249	0.3924324324

Table 13. Artificial intelligence – Word – Stemming – 100 pages

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	machine learning *	16,300,000	6,670,000	0.1886402441	0.0000000053	0.0355851417	0.0869621904	0.0869621904	0.4092024540
	expert systems *	8,810,000	3,530,000	0.1357966540	0.0000000052	0.0184407312	0.0460234681	0.0460234681	0.2165644172
	knowledge representation *	8,590,000	3,080,000	0.1199931495	0.0000000047	0.0143983559	0.0401564537	0.0401564537	0.1889570552
	neural networks *	19,000,000	3,880,000	0.1016382651	0.0000000027	0.0103303369	0.0505867014	0.0505867014	0.2380368098
	İnference	43,000,000	5,170,000	0.0900240679	0.0000000016	0.0081043328	0.0674054759	0.0674054759	0.3171779141
	natural language processing *	9,450,000	2,240,000	0.0832021276	0.0000000031	0.0069225940	0.0292046936	0.0292046936	0.1374233129
	fuzzy logic	5,280,000	956,000	0.0475054295	0.0000000024	0.0022567658	0.0124641460	0.0124641460	0.0586503067
	Cybernetics *	9,460,000	1,270,000	0.0471476956	0.0000000018	0.0022229052	0.0165580183	0.0165580183	0.0779141104
	artificial life	3,160,000	700,000	0.0449631591	0.0000000029	0.0020216857	0.0091264668	0.0091264668	0.0429447853
	turing test	925,000	363,000	0.0430961106	0.0000000051	0.0018572747	0.0047327249	0.0047327249	0.0222699387
	Ontology	43,900,000	2,100,000	0.0361900641	0.0000000006	0.0013097207	0.0273794003	0.0273794003	0.1288343558
	speech recognition *	14,800,000	1,140,000	0.0338357893	0.0000000010	0.0011448606	0.0148631030	0.0148631030	0.0699386503
	data mining *	54,900,000	2,190,000	0.0337489496	0.0000000005	0.0011389916	0.0285528031	0.0285528031	0.1343558282
	Prolog	19,000,000	1,050,000	0.0275052006	0.0000000007	0.0007565361	0.0136897001	0.0136897001	0.0644171779
	face recognition *	2,720,000	193,000	0.0133621141	0.0000000009	0.0001785461	0.0025162973	0.0025162973	0.0118404908
TYPE II	formatting instructions	269,000	16,200	0.0035664915	0.0000000008	0.0000127199	0.0002112125	0.0002112125	0.0009938650
	example systems	165,000	13,800	0.0038791755	0.0000000011	0.0000150480	0.0001799218	0.0001799218	0.0008466258
	fuzzy logic faq	305	166	0.0010853267	0.0000000071	0.0000011779	0.0000021643	0.0000021643	0.0000101840
	neural networks faq	639	345	0.0015583715	0.0000000070	0.0000024285	0.0000044980	0.0000044980	0.0000211656
	artificial life faq	1,520	191	0.0005593890	0.0000000016	0.0000003129	0.0000024902	0.0000024902	0.0000117178

Table 14. Artificial intelligence – Phrase – w / o Stemming – 100 pages

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	optical character recognition *	2,090,000	93,200	0.0073611402	0.0000000006	0.0000541864	0.0012151239	0.0012151239	0.0445933014
	information integration	6,580,000	226,000	0.0100599961	0.0000000004	0.0001012035	0.0029465450	0.0029465450	0.0343465046
	natural language processing *	9,450,000	2,240,000	0.0832021276	0.0000000031	0.0069225940	0.0292046936	0.0292046936	0.2370370370
	expert systems *	8,810,000	3,530,000	0.1357966540	0.0000000052	0.0184407312	0.0460234681	0.0460234681	0.4006810443
	machine learning *	16,300,000	6,670,000	0.1886402441	0.0000000053	0.0355851417	0.0869621904	0.0869621904	0.4092024540
	data mining *	54,900,000	2,190,000	0.0337489496	0.0000000005	0.0011389916	0.0285528031	0.0285528031	0.0398907104
	program committee	31,300,000	307,000	0.0062656845	0.0000000001	0.0000392588	0.0040026076	0.0040026076	0.0098083067

Table 15. Data Mining – Phrase – w / o Stemming – 100 pages

	Candidate Phrase (CP)	n(CP)	n(b)	S_1	S_2	S_3	S_4	S_5	S_6
TYPE I	text mining *	2,600,000	590,000	0.0468490102	0.0000000037	0.0021948298	0.0096721311	0.0096721311	0.2269230769
	neural networks	19,700,000	1,370,000	0.0395205097	0.0000000011	0.0015618707	0.0224590164	0.0224590164	0.0695431472
	Datamining	2,510,000	327,000	0.0264268850	0.0000000021	0.0006983802	0.0053606557	0.0053606557	0.1302788845
	lecture notes *	27,000,000	465,000	0.0114579297	0.0000000003	0.0001312842	0.0076229508	0.0076229508	0.0172222222
	about spss	66,300	17,800	0.0088511219	0.0000000044	0.0000783424	0.0002918033	0.0002918033	0.2684766214
	software and solutions	584,000	31,500	0.0052776333	0.0000000009	0.0000278534	0.0005163934	0.0005163934	0.0539383562
TYPE II	course home *	1,840,000	844	0.0000796652	0.0000000000	0.0000000063	0.0000138361	0.0000138361	0.0004586957
	nonlinear regression methods	776	87	0.0003998743	0.0000000018	0.0000001599	0.0000014262	0.0000014262	0.1121134021
	visualizing text mining	24	24	0.0006272500	0.0000000164	0.0000003934	0.0000003934	0.0000003934	1.0000000000
	study materials *	2,710,000	791	0.0000615215	0.0000000000	0.0000000038	0.0000129672	0.0000129672	0.0002918819