TOPIC SPECIFIC SPIDER FOR TAXONOMIC DOCUMENTS

BY

Krishna Priya Kotcherlakota

B.Tech., Computer Science and Information Technology,

G.V.P. College of Engineering, Vizag – Jawaharlal Nehru Technological University,

India – May 2004

Master's Thesis

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master's of Science in Computer Engineering

Chairperson _____

Prof. Susan Gauch

Committee members* _____

Dr. David Andrews

_____

Dr. Donna Haverkamp

Date defended: _____

The Thesis Committee for Krishna Priya Kotcherlakota certifies

That this is the approved Version of the following thesis:

TOPIC SPECIFIC SPIDER FOR TAXONOMIC DOCUMENTS

Thesis Committee

_____

Chairperson: Prof. Susan Gauch

_____

Dr. David Andrews

_____

Dr. Donna Haverkamp

Date Approved: _____

Abstract

The overall goal of the project being developed at KU as a part of SEEK is to provide an interface tool for a taxonomist through which he/she can obtain taxonomic revisions depending for the species provided to the system. This thesis, in particular, aims to develop a subsystem that collects taxonomic documents available on the World Wide Web using a combination of spidering and document classification techniques. To increase the number of documents collected, two query expansion techniques have been considered and evaluated. We found that generic query expansion performed better than the taxonomically expanded queries. Also, we show that filtering helps improve the performance of the system by reducing the number of non-taxonomic documents that are presented to the end-user.

Acknowledgements

I would like to first thank Dr. Susan Gauch, my advisor and committee chair for her help, guidance and support throughout my Masters program. It was a wonderful experience for me to work under her for my thesis. Her timely help, support and encouragement were the primary driving forces that kept me on the right track and I will always be grateful to her.

I would also like to thank Dr. David Andrews and Dr. Donna Haverkamp for agreeing to be a part of my advisory committee and for taking time to review my report.

Special thanks to all my friends who stood by me whenever I needed them and helped me complete my Masters successfully. Last but not the least, I wish to express my gratitude to my parents who have been a strong pillar of support for me and I thank them for all their encouragement and support without which I wouldn't have been able to complete my thesis.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

With continuing advancements in the fields of computation and mathematics, biodiversity research has witnessed large amounts of change in its scientific process. Various fields of biology such as the molecular sciences, structural biology and cell biology, are adopting new technological evolutions and are reformulating themselves. This thesis is being done as a part of the SEEK (Science Environment for Ecological Knowledge) framework which provides ecologists and researchers access to large amount of information resources [25].

Our understanding to the natural world is dependant on our access to information and ecological data and obtaining information from the knowledge already present to obtain new concepts. The SEEK is a web-enabled technology framework which has been integrated to assist the field of ecology, wherein investigations of researchers encompass both life and physical sciences and biocomplexity studies [25]. The major improvements that it suggests include global access to information and data using distributed computational services. All these capabilities will facilitate the advancements in global research and access to distributed data required by the domains of environmental biology.

As a part of this project, a system is being developed at KU that uses the already existing taxonomic concepts to collect more information and then update the existing taxonomic information based on the information further extracted. In order to demonstrate this process, *Bats of the World 1993* is being used as an existing set of taxonomic concepts. In order to demonstrate our approach, we search for literature available online and suggest revisions since that publication. When searching for information, the taxonomist or an ecologist defines his need in the form of a query containing the species or genera on which he is working; he/she then submits it to our system. We first submit the query to a search engine(s), which in turn finds all the documents in its collection that match the query. However, in order to identify taxonomic publications, we apply classification-based filtering. This is followed the process of concept mining to obtain suggested revisions such as additions, deletions and modifications to the already existing concepts and the system is evaluated by comparing these suggestions to manually produced revisions.

1.2 Goals

The goal of the project being developed at KU is to develop a system that semi-automatically updates an existing taxonomy. To demonstrate this, we will use the *Bats of the World 1993* and search online literature for relevant taxonomic publications since then. We will text mine those documents to suggest revisions, that is, additions, deletions, and modifications. We will evaluate our system by comparing

the autosuggestions to upcoming manually produced revisions. A modular architecture has been proposed to implement this technique.

As a part of this project, a system is being developed which updates a taxonomy that already exists. This system involves document searching depending on concepts which already exist, and then filter the documents and use concepts of text mining to suggest any revisions to these documents. The concept miner is a subsystem which is used for extracting concepts from these documents and thus, forms new concepts. These concepts can be a revision of the already existing ones or even may contradict those or are fresh ones. We identify revisions based on these concepts and then automate these revisions. Finally, the user is presented with an interactive system which acts as an assistant to the taxonomist.

This thesis develops a subsystem, i.e., to obtain a taxonomic document collection, that is as an input to the concept miner that has been developed as part of a M.S. thesis [27]. The following are the goals:

- Develop a spider program which considers the URLs provided by Google™ Scholar as search results to query terms as seed URLs, then crawls through each of them iteratively and collects taxonomic documents during the process.
- Evaluate and compare the results produced by the two query expansion techniques: Generically Expanded and Taxonomically Expanded Queries.
- Evaluate the system that has been developed by considering the accuracy, precision, and recall measures.

- Confirm our hypothesis that document filtering helps us eliminate much of the irrelevant information.

CHAPTER 2

RELATED WORK

This thesis concentrates on the Spidering process in which a program downloads web pages from the World Wide Web in accordance to the query term provided to it. Such a program which retrieves the web pages is called a spider or a web crawler. However, the process of retrieval occurs in an orderly and automatic manner. A crawler starts out with a particular set of URLs which are called *seed URLs*. The process of spidering, the types of spiders available and topic specific spidering in particular, are discussed in section 2.2.2.

Chapter 2 discusses the already existing and ongoing research in the areas related to topic specific spidering.. Text classification studies the assignment of documents or text into predefined categories and the system which performs this assignment is called a classifier. Section 2.2 focuses on text classification and the different text classification algorithms available. In particular we review spidering research in section 2.1.1. Since topic specific spidering tries to identify pages related to a specific topic, most topic specific spiders employ classification techniques which are discussed in section 2.1.2

2.1 SPIDERING

2.1.1 Overview

Today, the Internet is a part and parcel of everyone's life. Given its site, locating information be surfing alone, i.e., following links from web page to web page, is infeasible. To more quickly locate information, search engines were developed to collect huge number of documents, and indexed them to provide fast keyword-based searching. Google™ , for example, contains a collection 8 billion documents.

The process of crawling the World Wide Web (WWW), using an automated script of program called a *web spider* or *web crawler* in a methodical and automated manner is called *Spidering* [16]. Spiders usually keep a log of all the crawled web pages for further processing which can be further indexed by search engines or can be used for link validations etc. A crawler or spider, which is a program, sets off on a set of URLs which are called the seed URLs and then starts crawling these seed URLs to gather more information. As it crawls through each of these pages, it keeps adding URLs in that web page to the list of links to be visited, which is called the *crawl frontier*.

Because of the huge amount of information available in the Internet, and its high rate of change, spidering is a continuous process. So, it is important to have a set of policies that help achieve efficiency. The following are the policies that must be created:

- *Selection Policy*: A policy which specifies the pages to be downloaded.
- *Re-Visit Policy:* A policy which specifies when to check for changes in the web pages.
- *Politeness Policy:* A policy which states as to how we can avoid overloading websites.
- *Parallelization Policy:* A policy which specifies as to how the process of spidering can be distributed on the web in a coordinated manner.

Selection Policy:

Spiders are given only a particular set of URLs (seed URLs) to start crawling. In order to maximize coverage of this set of URLs should be highly representative of the derived content. There are different selection policies available. In their study on policies for crawling on a single domain, [17] suggests that the crawler wants to download pages with High PageRanks first, i.e., important pages identified by link structure. Then their partial PageRank strategy collects pages by following links using breadth-first and backlink-counts.

[18] used breadth-first ordering to collect pages form the web. They crawled 328 million web pages and they found that this strategy downloads pages with high PageRank early in the crawl process, without explicit PageRank information. The reason for this is that all important URLs have backlinks to them form other hosts and hence, are found early. However, they did not compare their results to any other

strategy. The crawler can only crawl HTML pages and avoids other MIME types, it can collect web pages with extensions like .asp, .html, .jsp, .php, .aspx and /.

Re-Visit Policy:

The pages maintained on the Web are updated very frequently and crawling is a process which happens for a long time. In such cases, by the time the crawl is completed there might have been many updates on the Web and for certain crawls this new information might be very important. [19] came up with two measures *freshness* and *age* wherein freshness is a measure which indicates whether the information available locally is accurate or not and age is the measure which indicates out outdates the information is. They also came up with two policies, the *uniform policy* wherein every page is revisited at a uniform frequency and the other is the *proportional policy* wherein pages are revisited depending on the rate at which they are updated. These policies attempt to improve freshness of a search engine's collection.

Politeness Policy

The performance of a website being crawled can deteriorate because a crawler can generate multiple requests per second. Downloading large amounts of data from the site consumes network resources, and can cause the server to overload or even crash them. [20] suggests a standard for spider or robot exclusion, wherein site administrators are advised to include a protocol that states the parts of their website

which should not be accessed by the robots. Also, many search engines use a parameter called the *crawl delay* that indicates the time delay between each crawl request to a particular website.

Parallelization Policy:

A crawler which runs multiple processes is called a *parallel crawler* and the main purpose of a parallel crawler is to maximize the rate of download. Parallel crawlers must be careful to avoid repetitive downloads by adopting a policy for assigning URLs to a crawl process. [21] studied two policies, *dynamic assignment* wherein a central server assigns the kind of URLs which are supposed to different crawlers and *static assignment* wherein a static set of rules are laid out before starting the crawl process and it specifies the URLs to be collected by the crawlers.

2.1.1.1 Types of Spiders

In general, there are two types of spiders: Standard Spiders and Topic-Specific Spiders. Standard spiders are based on seed URLs and they try to collect and index all documents available on the Web in order to be able to answer all ad-hoc queries [23]. On the other hand, a topic-specific spider, as the name suggests, tries to acquire and index documents that are relevant to a set of predefined topics, and it tries to avoid those pages which seem to be irrelevant. This type of crawling is also called Focused Crawling [24] and though they are more efficient for a particular topic, the topic must

be known a prior. It is difficult to develop such crawlers because they must make additional decisions regarding the relevance of a document.

In order to traverse a link in a page, a crawler needs to select a particular strategy and this strategy selection also depends on the main purpose of the crawler and the kind of Web sites it traverses. Two such strategies are *link-based* and *similarity-based*. For link-base measurements, in-degree, out-degree, PageRank, and hubs and authorities are commonly used [22]. In-degree is the number of backlinks to the webpage, out-degree is the number of links that start from the web page and PageRank is the probability that a surfer might visit that Web page. In the similarity-based strategy, the document is checked for the query term, and depending on the frequency of the term in the page, a similarity score is given. Based on this score a decision is made. Other measurements can be based on the anchor text of a URL, the content around the URL, or *keywords* within the document. Researches suggest combining the link-based strategy with the similarity-based strategy because as PageRank alone does not consider the content of the page. Our project adopted the similarity-based strategy alone.

2.1.1.2 Topic-Specific Crawling

As specified above, if a page is downloaded by a spider based on a similarity score between a page and the given query, the spider is said to be a Topic-Specific Spider and such a process of crawling is called Focused Crawling. In Focused Crawling, we can predict the similarity of the text even before actually downloading it by considering the anchor text in links. Researchers have also developed crawlers which calculate a similarity score of a page based on its entire content and the given query term. However, the performance of focused crawling or a topic-specific spider depends mostly on the richness of links being crawled and it usually is also dependant on a general search engine, in our case Google™ Scholar, for seed URLs which are the starting points for the crawler.

[24] first introduced the concept of focused crawling. A focused crawler's goal is to seek pages relevant to a set of topics and these topics are specified using exemplary documents. A focused crawler analyses the crawler frontier to find the links which need to be crawled are likely to be most relevant and has to avoid irrelevant parts of the Web, thus, saving network and hardware resources. In order to achieve focused crawling, they developed a *classifier* which classifies all the hypertext documents and a *distiller* which identifies hypertext nodes which are access points to pages which are relevant and reported that when the same set of seed URLs were given to a standard crawler and to a focused crawler, the focused crawler acquired pages steadily while

the standard crawler lost its way easily. Thus, they suggest that focused crawling can be used to develop high quality collections of documents relevant to a specific topic. However, they designed a classifier which is dependent on a hierarchy of topics and their evaluation was also based on the same hierarchy. In this project, we use a normal simple set of topics which basically deal with documents related species of bats, i.e., the species relationships, the phylogeny of species, new species or genus names discovered and recovered, taxonomic revisions done, etc.

[22] developed a topic-specific crawler and evaluated them using, classifiers. They chose 100 topics and all the positive example of a particular topic become the negative examples of all the other topics. The classifiers are used to assess the pages recently crawled and a page which is positively classified by the classifier is considered a "good" page, in the sense that it can be used as a measure which is similar to the *precision* of a classifier. In general, a crawler is evaluated based on its ability to retrieve a "good" page, but the major problem lies in identifying such pages and in operational environments, real users need to judge the relevance of the pages and state whether or not the crawl was successful or not. In their design, they also used feature selection along with their classifiers. The classifier was designed to select the best 50 features for each topic. All topics were identified from the Yahoo! Hierarchy; however, the topics identified by them are the leaf nodes of the hierarchy. They evaluated their system using three classifiers, Rocchio, Widrow-Ho® (WH), and Exponentiated Gradient (EG). The results pertained to classifying each set of

1000 crawled pages. As mentioned before, content-based relevance is given by the classifier and in a sense the evaluation is based on the crawler precision. The best classifier results given were close to 13% precision using Rocchio and the reason for low precision is stated to be because there were a very few positive example pages per topic compared to an average of 700 pages used as negative examples.

2.2 TEXT CLASSIFICATION

2.2.1 Overview

Web Mining is often described as the discovery and the analysis of useful information from the World Wide Web [1] and text classification or document classification can be viewed as one such web mining technique. As described earlier, text classification or text categorization is the process of assigning a document to one or more of a set of predefined categories. Text classification can be used as information filtering tool which helps improvise the results retrieved from querying.

Text classification involves two phases, namely, the training and the classification phase. The training phase, as the name suggests, trains the classifier wherein each of the categories in the system are represented by information learned from respective document sets. Once the training phase is complete, a new document collection is provided to the classifier for classification, which is called the classification phase.

This phase compares each of the documents in the new document collection to the category representations and assigns them to one or more categories.

There are a wide range of text classification algorithms, but some of the widely used are : k-nearest neighbor [kNN] , Naïve Bayes classification algorithm [NB], the vector space model, Rocchio, Support Vector Machines [SVM], neural networks, and the linear least squares fit mapping [LLSF]. [7] explains each of these text classification algorithms and a gives a comparison of the performance analysis of each of the classification systems. The authors conclude that the SVM and the kNN perform better than all the other classifiers.

2.2.2 Selected Classification Methods

In ***kNN*** [2], in order to classify a new document, the classifier finds the *k* nearest neighbors among the training documents and calculates a similarity score between the documents to be classified and its neighbors. Based on these sorted scores, the first k matches are found and for each of these matches the known category names are used to find the category of the document. If two neighbor documents belong to the same category and are in the top k matches, the scores are added up to get an overall score. The performance of this classifier depends highly on the value of parameter k and the similarity function being used to calculate the scores [3].

Support Vector Machines [5], is considered as a binary classification algorithm which learns through optimization. For a given set of training data, it considers each document in the set to be a vector and SVM is a hyperplane which separates the training data. Vectors which lie on one side of the hyperplane are said to belong to class +1 and those on the other side belong to class -1. During learning, the classifier attempts to find a boundary between these two classes which separa:tes the documents in a best possible way. During classification, a document is classified by calculating its distance form the boundary. This classifier has been discussed in detail later in this section.

A ***Naïve Bayes*** classifier [4] calculates the probabilities of categories for a given document to be classified based on the joint probability of words and categories. This approach assumes that a parametric model generates the text data and the training data is used to calculate the Bayes estimates of the model parameters and based on these estimates, the classifier classifies the new documents.

Rocchio is an example of the classic vector-space model method for document filtering [7]. Here the basic idea is to develop a vector for each category used in the document set provided for training. For any given category, each of the documents belonging to that category are given positive weight and all the vectors of the remaining documents are given negative weights. By summing all theses weights, a prototype vector is developed and based on this vector the testing set is classified.

However, in this method an assumption of one centroid per category is considered a potential weakness.

Linear Least Squares Fit is a method which maps text to canonical terms [8]. It learns the association between the text and the terms through human-assigned matches and finds a mapping function termed as the Linear Least Squares Fit [LLSF], which represents the association in the form of weights. This function is used for projecting the text to the canonical word space and similarity scores are used to determine the relevance between the text and the terms.

[7] describes a ***neural network*** approach to text classification, where an appropriate choice of features is chosen to use in feature vectors and a feature vector representation of text is done. For each category, feature vectors are obtained from the text representing a particular category and a learning algorithm is developed which learns the association between the text and the category to build a classifier.

2.2.3 Discussion of Rocchio and SVM

In this thesis, we use two classifiers for our document collection – SVM and Rocchio. The reason for using two classifiers is discussed in detail in Section 3.3. The document collection that is collected by the spider, is a combination of both HTML and PDF documents. The Rocchio Classifier is used for classifying all HTML

documents and the SVM classifier is used to classify all PDF documents. Section 4.1 discusses the evaluation of each of these classifiers for both the document types. Based on these results, the appropriate choice of the classifier is done. We make use of the classifier developed for the KeyConcept [11], which uses vector space model for classification, for classifying all HTML documents. In this approach, each of the documents is treated as a vector represented as term weights and for each document; the term weight in a document is calculated using the following equation:

$$TERM\_WEIGHT_{ik} = WORD\_FREQUENCY_{ik} \times \log\left(\frac{num\_docs}{doc\_freq_k}\right) \quad [10]$$

where $TERM\_WEIGHT_{ik}$ – the weight of the term *k* in the document *i*

$WORD\_FREQUENCY_{ik}$ – the frequency of the word *k* in the document *i*

num_docs – the number of documents in the collection

doc_freq_k – the number of documents in which *k* occurs

According to the above formula, it can be inferred that as the frequency of a word in a document increases, the term weight also increases, thus, improving the content representation.

In this classifier, we have training and testing phase, like all the other classifiers. For training, a set of documents, termed as the training set, are used to represent each of the categories in the classifier. In a vector space model, the centroid of all the training documents is considered as the category vector and when classifying a document, it

assigns one or more categories to it based on the terms the document contains. Using the cosine similarity measure between the document represented as a vector and the category centroids, the document is assigned to the most similar categories.

$$Cosine\ similarity\ measure = \frac{d \times d'}{|d||d'|} \text{ [12]}$$

Where d x d′ is the vector product of d and d′ which are both vectors and the cosine measure returns the angle between the vectors.

We also use the SVM classifier, which we have used is to classify PDF documents. We have used libSVM [13], is an integrated software system which is used for classification using support vector machines, developed at the National Taiwan University. libSVM provides an interface using which users can easily link it with their own programs. We have linked our main spidering program with this software. During, crawling, if we encounter a PDF document, we download the content of the document and send it to the classifier for classification. The classifier that we use here for this purpose is the libSVM software.

During training, the libSVM classifier is trained using a document collection of both taxonomic and non-taxonomic documents. During evaluation of this classifier, we concluded that an optimal value of 20 documents/category can be used for classification purposes. This evaluation has been discussed in detail in section 4.1.

Thus, we used the classifier trained with a total of 40 documents/category for classification purposes. SVM has been discussed in section 3.3.2.

2.3 SEEK (Science Environment for Ecological Knowledge)

SEEK is a ITR Collaborative Research Project funded by the NSF, its main challenges involving development of a framework for biodiversity researches by modeling, designing and implementing data discovery, integration and visualization components for the web-based framework in environmental science [25]. SEEK is an integration of IT researches into the web-based framework, that aims to fundamentally improve the manner in which researchers can

- gain global access to information and date
- locate and utilize distributed computational services
- develop new, powerful methods for capturing, reproducing and extending the analysis process.

SEEK also provides access to large-scale network, information, and computational resources that can be used by ecologists and other researchers via analysis tools which can be operated on desktops. The main aim of developing these capabilities is to more effectively address the issues like global research management and policies in environmental biology which require more efficient and automated access to

distributed and heterogeneous data [25]. A major problem with biodiversity and ecological data is that it is heterogeneous in syntax, schema, and semantics, which prevents discovering, integrating and synthesizing this data.

An ecologist wanting to search for information about a specific kind of virus in a species, usually searches a few science portals and other search engines to get the information. However, this process is time–consuming and, most of the time, he/she will end up with results which are inappropriate and a result set which needs to be cleaned up. Even if the relevant data is found by the ecologist, there might a lot of data access problems such as problems downloading the data and data format issues etc. To be even more specific, the query for *specific kind of virus in a species* might involve some taxonomic names which might change over time. Thus, these taxonomic names are important when dealing with scientific datasets.

The project which is being developed at KU aims to develop a system which semi-automatically updates an existing taxonomy to keep the information about species names and relationships up to date. In order to demonstrate this, we will use *Bats of the World 1993* and search online literature for relevant taxonomic revisions publications since then. To do this, we will compare the revisions we find with those that appear in *Bats of the World 2005*. We developed a modular architecture for the implementation of the system. The architectural details of the system are discussed in Chapter 3.

CHAPTER 3

APPROACH

3.1 Overview

The basic overview of the system being developed at KU is discussed and the subsystem that is being implemented for this thesis is elaborated. Finally, each of the modules in the subsystem is explained, followed by algorithms based on the entire subsystem.

3.1.1 SEEK Information Extraction Architecture

The main focus of this thesis is the two modules in the SEEK architecture being developed at KU – the topic specific spider and the document filtering modules shown in Figure 1. The approach to the SEEK project is as follows:

1. Extraction of content from the *Bats of the World 1993*
    - Extracting all the binomials from *Bats of the World 1993*.
    - Extracting all parent-child links
2. Creating concepts in the TCS using the content extracted.
3. Finding all relevant papers published since 1993 for concepts
    - Querying using binomials and genus names

- Filtering all the documents to find taxonomic treatments

4. Identifying the changes in the taxonomic literature
5. Provide revisions from the changes.
6. Compare the revision thus produced with the new concepts identified.
7. Provide an interactive system to the taxonomist who approves the updates.

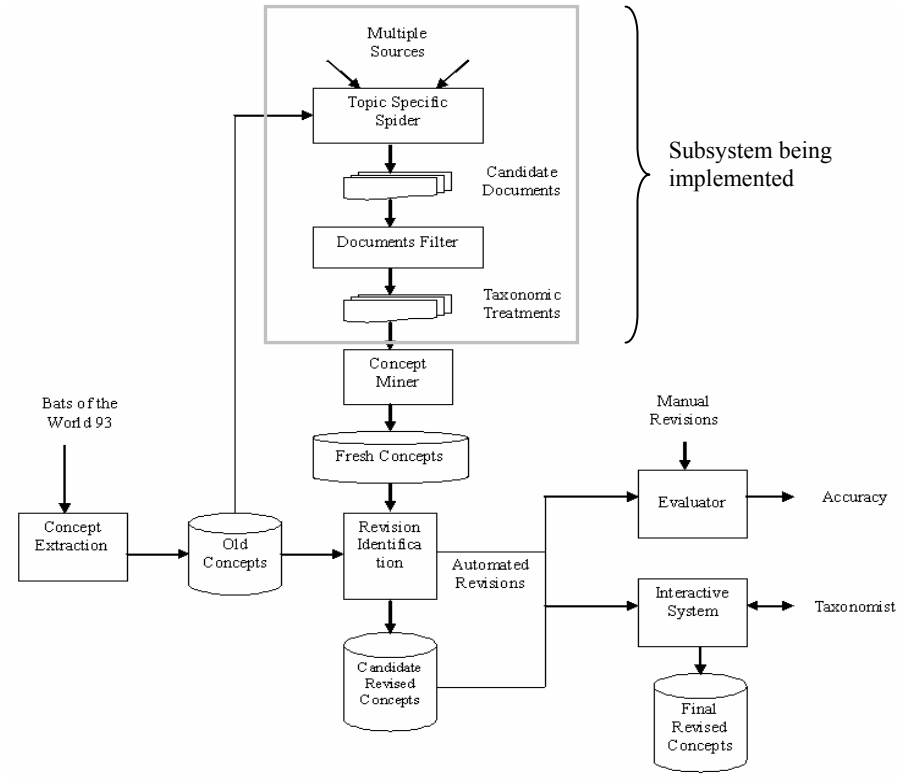


Figure 1: SEEK System Architecture

3.2 Subsystem Architecture

The architecture of the subsystem developed as a part of this thesis is shown in Figure 2. The two major modules of this system are the Topic Specific Spider and the Document Classifier. The Topic Specific Spider is used to collect information from multiple sources and the Document Classifier is trained to filter documents which are not considered as taxonomic treatments. The entire system starts off with a user specified query, which can be given by a taxonomist. There are two types of query expansion techniques evaluated for this system, and they are discussed later in this chapter. The entire system has been implemented in Java.

Topic Specific Spider

The topic specific spider, as discussed in Chapter 2, begins by crawling through a set of URL links which are called the seed URLs. The seed URLs, in this project, are the set of URLs which are given as search results when Google™ Scholar is queried by the taxonomist. Each of the Web pages pointed to by these URLs is downloaded by the crawler. Every URL crawled by the program, is added to a queue so that we can avoid duplicate downloads.

The two major document types we encountered were HTML pages and PDF (Portable Document Format) files and there were only 2 documents which were downloaded when evaluating the system that were not of these types, i.e. they were in Microsoft® Word format. However, such documents were not taken into consideration when

developing this system because the number of taxonomic documents available in this format on the Web is very low.

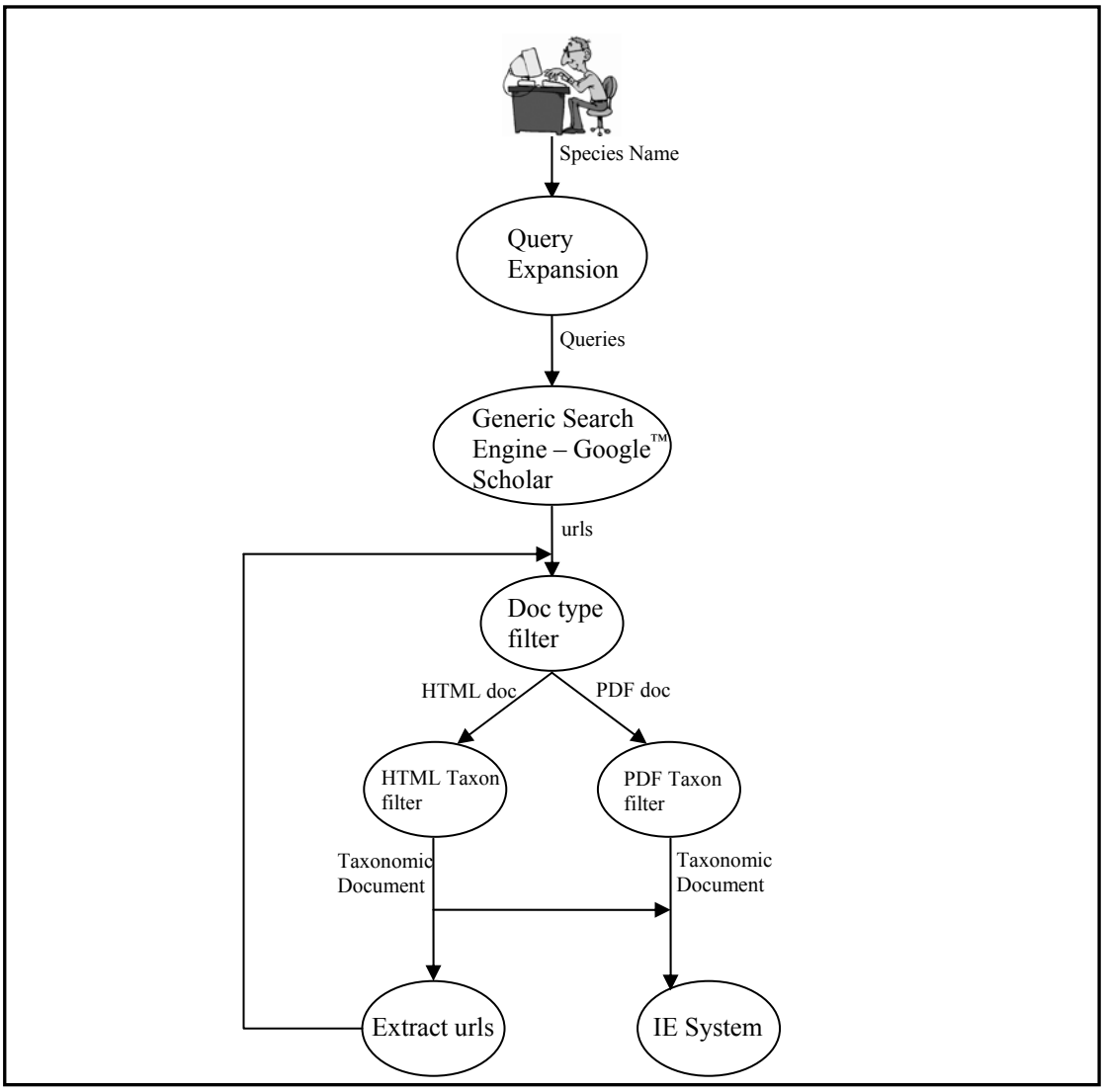


Figure 2: Subsystem Architecture

Document Type Filter

Once the Web pages are downloaded, they are passed through the document type filter which identifies whether the document is PDF or not. In this project, if the document is not a PDF file, we consider to be a HTML page by default. Depending on the document type, we send the document for classification by the appropriate classifiers. The reason that two classifiers have been considered for this purpose is that, all journal publications and research papers which are in the PDF format were full documents whereas the HTML pages usually consisted of only the abstract of the papers. Thus, there was a large difference in the size of each of those files and so we built a different classifier for each of the file formats.

Query Expansion

The taxonomist begins the search for information by supplying one or more queries containing species or genera names. In order to find more information we investigated two types of query expansion techniques. Given a *species* name by a taxonomist, the range of documents that the search engine, in this case Google™ Scholar, returns can be huge. However, after the first 100 search results or so, the probability of finding high quality, taxonomic documents is low. Thus, the query is expanded in order to include more information regarding the type of documents required.

We considered *generically expanded* queries wherein the query is the concatenation of a *species name* provided by the taxonomist and a general term like "new genus" or "relationship between species" etc. The second type of query expansion technique is taxonomically expanded queries wherein the query is a combination of the *species name* provided by the taxonomist and terms which are taxonomically related to the species name like the subspecies, order, suborder, genus names, etc., extracted from the domain specific related concepts.

Document Classification

This project evaluated two types of classifiers, SVM and Rocchio, for classifying documents. The accuracies of both the classifiers were compared using the same training and test collections for both the document types – HTML and PDF - and we found that, for PDF documents, the accuracy of SVM was better when compared to Rocchio whereas the accuracy of Rocchio was higher than that of SVM for HTML documents. For details, see experiment 1 in section 4.1.

The documents which are sent into the classifiers are then classified into 2 categories - taxonomic or non-taxonomic by the classifiers. If the document is a HTML page and is taxonomic, it is sent for further processing, i.e., the URLs are extracted from the page and are added to the *crawling frontier*, which are in turn sent to the spider or the crawler to be crawled. If the document is in the PDF format, it is converted to a text document and, if it is classified as a taxonomic document, it is sent directly to the *Information Extraction* System for the concepts to be extracted.

URL Extraction

All the HTML documents which are sent into the URL extraction module are parsed and then searched for the anchor tags in the document. Whenever the tag refers to another document using "href," the URL link to which the tag is referring to is extracted and added to the *crawling frontier* (url_queue), if and only if the URL is not present in the crawled list of URLs or is already not in the crawling frontier. To better explain the classification process, Section 3.2 entirely focuses on the two classification techniques used in this project.

Algorithm

```
Spider(query_term qt)
{
   seed_urls=search_topic(qt);
   foreach url (seed_urls)
   {
      enqueue(url_queue,url)
   }
    while(!empty(url_queue))
    {
       url=dequeue(url.queue)
        page=crawl_page(url);
        enqueue(crawled_url_queue, url);
        category=classify(page);
        if(category=”taxonomic”)
        {
            if(page.type=”HTML”) then
                foreach link=extract_urls(page)
                 if(link ! url.queue and link !
                    crawled_url_queue)
                 enqueue(url_queue,link)
             else
             if(page.type=”PDF”) then
                information_extract(page);
         }
         else discard(page);
     }
   }
}
```

Figure 3: Algorithm for the spidering process

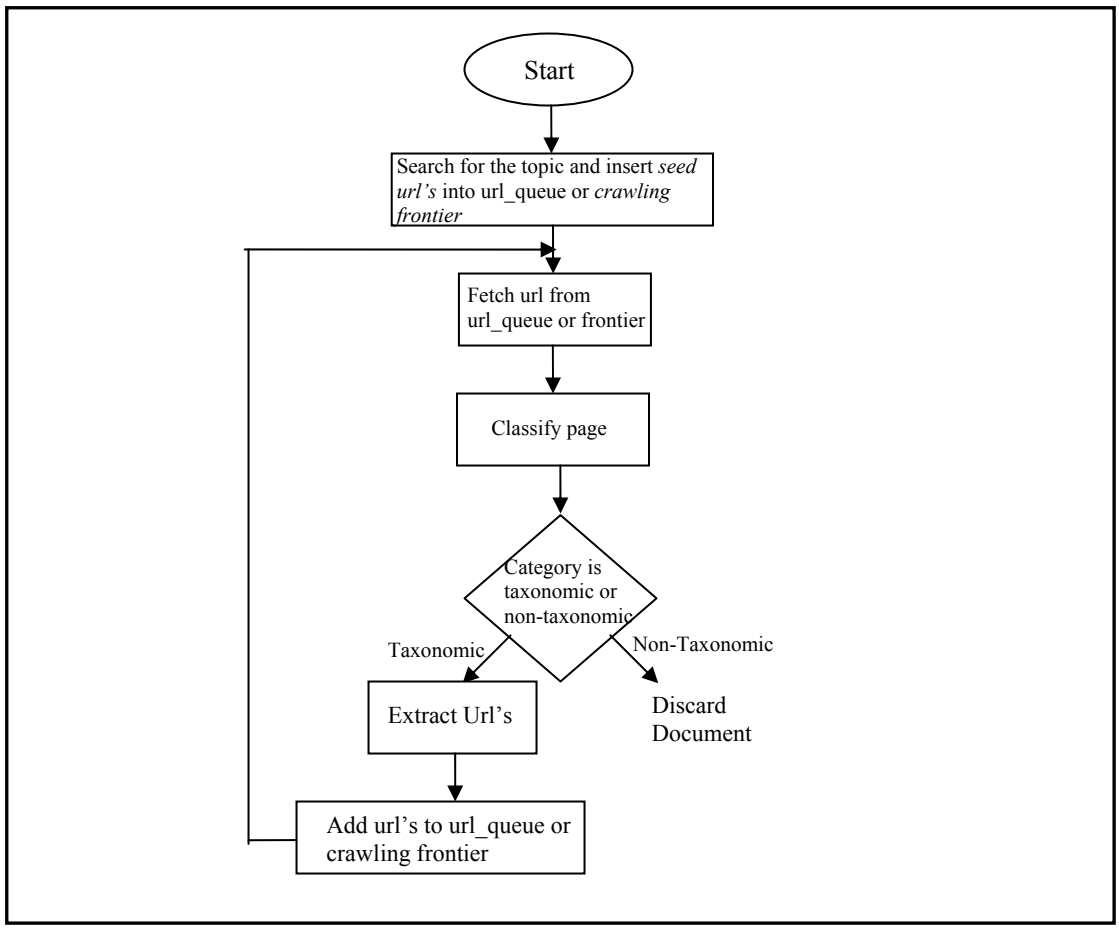


Figure 4: Flowchart of the topic specific spider

The algorithm in Figure 3 and the flowchart in Figure 4 outline the spidering process. The process of URL extraction has been implemented using Java and the queuing process has been implemented using queue implementation of Vectors in Java.

3.2 DOCUMENT FILTERING

One of the goals of this project and a topic specific spider is to minimize the number of useless documents encountered during the process of crawling. We employ document filtering based on text classification.

3.2.1 Rocchio Classifier

With the increasing number of web pages, users experience difficulty retrieving documents relevant to their search. In the case of spidering, a spider encounters a number of documents which are irrelevant to the user's interests. In a case like ours, wherein the search interests of the user are very specific and the number of documents relevant to the search and those available on the Web is very low. Thus, it will be very difficult for the user to find a relevant document.

KeyConcept

The process of spidering begins with a set of seed urls's which are the search results of a search engine. However, most search engines find matches based on keywords, regardless of their meanings and thus, there is every possibility that the all the results of the search engine are relevant and so they need to be filtered. For this process of filtering, we chose Rocchio and SVM classifiers. However, the Rocchio classifier has been implemented as a part of the KeyConcept, a search engine which retrieves

documents based on a combination of keyword and conceptual matching. Documents are classified to determine the concepts to which they belong. In the KeyConcept, when a query is given, the concept to which it belongs to can be specified explicitly or it can be determined by a user profile.

3.2.1.1 KeyConcept Categorizer

The indexing process of the KeyConcept consists of two phases: classifier training and collection indexing. During the training of the classifier, a fixed number of sample documents are collected for each of the categories, in our case 2 categories, and merged. The resulting documents are preprocessed and indexed using the vector space model which essentially represents each of the concepts by a centroid of the training set for that concept. During the indexing of the collection, new documents are indexed using the vector-space method to create a word based index. Then, the documents are classified by comparing the document to the centroids of each concept. The similarity values are then computed and a concept based index is then formed.

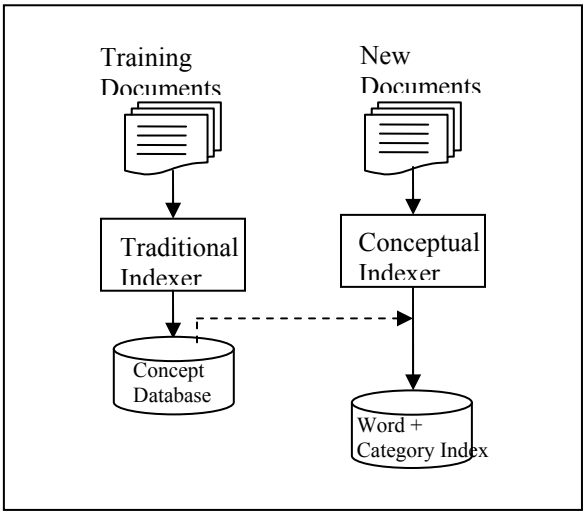


Figure 5: Indexing in KeyConcept

For this project, we used the KeyConcept categorizer to classify HTML documents. However, all the HTML tags have been ignored while indexing the documents.

3.2.1.2 Training Phase

For training the KeyConcept Classifier, we have collected a set of 50 HTML pages belonging to each of the categories, Taxonomic and Non-taxonomic. In order to index the data for each category, all the documents available as training data for a category are merged together into a superdocument, which is then indexed. However, before indexing all the stop words like *a, the, is, are, this, was, that* etc., are removed from the documents. Then, the documents go through the indexer to generate the keyword vectors. The indexing process results in the generation of the dictionary, postings and the documents files which keep information about the frequency of the keywords for each concept and are used by the classifier to match a document to the closest category. Essentially, the training phase generates an inverted index which, for each category, stores the weight of each term in that category.

Vector-Space Model

When the classifier is given a document for classification, it returns the category id along with the weight of the match. The highest weighted category is the resulting category for that document. Similarity between the categories and the document is determined using the Vector-Space Model, wherein all superdocuments and the documents are treated as n-dimensional vectors where n represents the number of

unique terms t_i in the vocabulary. Each of the documents, d_j is represented as a vector of term weights (w_{ij}), normalized by the length of the vector. The term weights are calculated by the equations given below:

$$wi_j = tf_{ij} * idf_i$$

Where tf_{ij} is the frequency of the occurrence of term t_i in document d_j

The inverted document frequency is calculated using the following equation:

$$idf_i = \log \frac{\#\text{ of documents in the collection}}{\#\text{ of documents in the collection that contain term } t_i}$$

The final term weight for each term t_i in document d_j is calculated by:

$$wij = \frac{wij}{\sqrt{\sum_{ti \in dj} wij^2}}$$

3.2.1.3 Classification Phase

Now that the documents have been represented as vectors with tf*idf weight components for each term, the similarity between a new user document and a category can be computed using the cosine angle between the two vectors using the following equation:

$$similarity(dj, qk) = \frac{q_k d_j}{\|q_k\| \|d_j\|} = \frac{\sum d_j(i).q_k(i)}{\sqrt{\sum_{i=1}^{n} d_j(i)^2 . \sum_{i=1}^{n} q_k(i)^2}}$$

where n – number of unique terms in the dataset

$d_j(i)$ – the i^{th} term in the vector of the document j

$q_k(i)$ – the i^{th} term in the vector for query k

The smaller the angle, the larger the cosine, the more similar are the two vectors and therefore the documents that each of the vectors represent. After calculating the similarity between the document and each of the categories, the top-matching category is assigned to the document. This process is repeated for each of the URL documents downloaded by the spider.

3.2.2 SVM Classifier

Considering a two-class classification problem, the goal is to separate the two classes by a function which is induced from available examples [26]. The goal is also to produce a classifier which also works well on unseen examples and thus, generalize the classification process. Figure 6 shown below shows that there can be many linear classifiers possible which separate the data, but there is only one which maximizes the distance between it and the nearest data point in each class. Such a linear classifier is termed the optimal separating hyperplane.

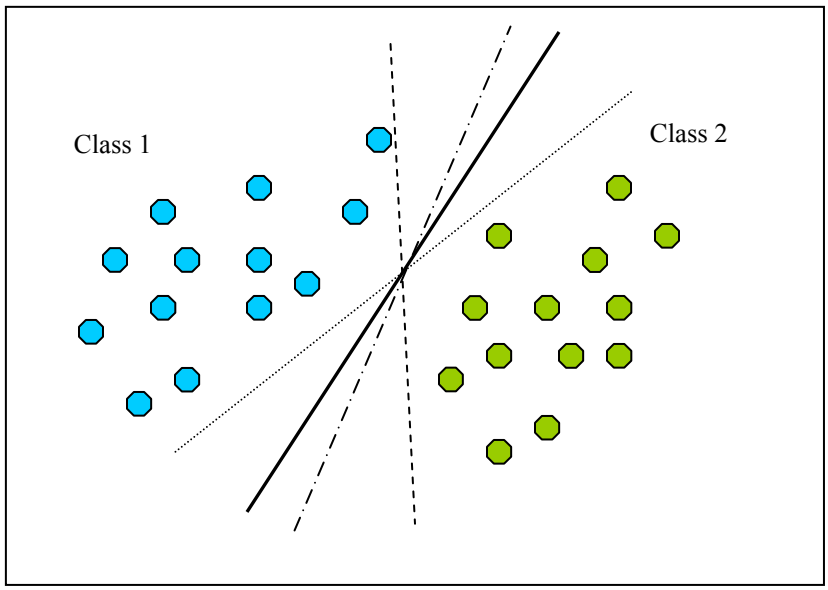


Figure 6: Optimal Hyperplane Seperation

The classification task usually involves with training and testing data which consist of data instances [5]. Each instance in the training set contains one target value i.e. the class labels or the category id and several attributes or features. The goal of SVM is to produce a model which predicts the target value of data instances in the testing set which are given only the attributes.

For this project, we chose to use libSVM [5], an implementation of Support Vector Machines for classifying of documents. Here Training vectors, $x_{i,}$ are mapped into a higher dimensional space by a function Φ, then the SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. The parameter $C > 0$ is a penalty parameter of the error term and the function Φ is called the kernel function. The procedure for the classification is described in steps.

1) Transform data to the format of SVM
2) Conduct simple scaling on the data
3) Use cross-validation to find the best values for C and γ
4) Use the best parameters C and γ to train the training set
5) Test the testing data or classify new documents

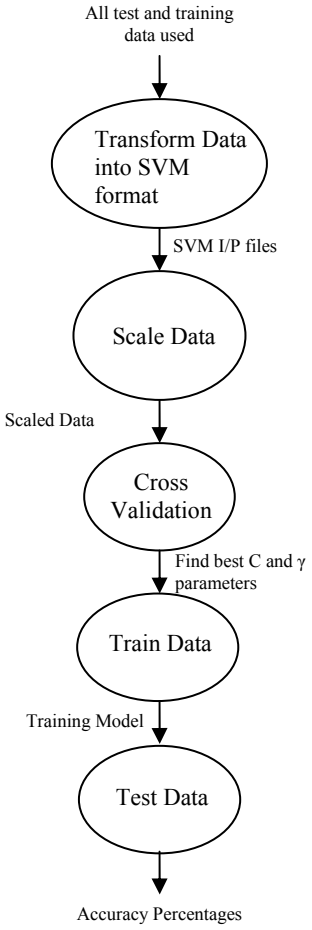


Figure 7: SVM Classification Process

3.2.2.1 Preprocessing of Data

SVM requires that each data instance be represented as a vector of real numbers. Hence all the categorical attributes need to be converted to numeric data. The following is the format of a SVM input file

```
<target> <feature>:<value>
```

where target, the category id, and the feature:value pairs are separated by a space.

The feature:value pairs must be ordered by increasing feature number and features with a zero value can be skipped. We have considered "value" to be the *tf* values which is computed by the following equation:

$$idf_i = \log \frac{\#\text{ of documents in the collection}}{\#\text{ of documents in the collection that contain term } t_i}$$

A program was written to convert the Rocchio input files to SVM format. For constructing a training file, we tokenize the each of the documents and construct the accumulator table with the word and the weight values. For each of the highest weighted words, the *word_id* is obtained from the global hash table and the value of the number of documents which contain the word is obtained inverted document frequency table. Thus, the *idf* values and the term frequency are calculated. Finally, when the training file is being built in the SVM format, the category_id, the term_id and the term_frequency are printed. This process repeats for each of the files in the training set. The global hash table and the idf tables are written to files so that they can be used when converting test data for SVM.

For generating the test data in the SVM format, we use the global hash table and the idf tables constructed during training and repeat the same process of calculating the idf and the tf values and thus, print the files in the SVM file format. However, we sort the accumulator table by the token frequencies and then keep the top words whereas during training we consider all the tokens.

We scale the data before applying SVM. The main advantage of doing this is to avoid attributes in greater numeric ranges to dominate those in the smaller numeric ranges [5]. Also, in the linear kernel which we are using, larger attribute values can cause numeric problems. Thus, each of the attributes is scaled down to the range [-1,+1]. libSVM, however, has tools which automatically scale the data and this tool has been used to scale the testing and the training data in this project.

3.2.2.2 Cross-validation

There are two parameters which need to be considered for SVM classification – C and γ. However, the best values for each of these parameters are not known beforehand. Thus, the goal is to identify these values so that the classifier can accurately predict the testing data. Classifiers can accurately predict training data for which the category values are already known. Thus, the training set is divided into two parts one of which is considered unknown when training the classifier. Then predicting this set of the training collection can reflect the performance of the classifier on testing data to an extent. This process is termed *cross-validation*.

The values of parameters C and γ are found using the grid selection method. Different pairs of C and γ are tried and the one with the best cross-validation accuracy is chosen. Parameter selection is also a tool which has been provided with the libSVM package and we have used it for this project.

CHAPTER 4

EVALUATION

We need to evaluate two key processes: the ability of the spider to find potentially relevant material and the ability of the classifier to filter out irrelevant material. We will compare the effectiveness of to classifiers, SVM and Rocchio, by measuring their accuracies on a collection of biological documents, some of which are a collection of biological documents and some of which are not, whose category membership is known in advance. We will measure the effectiveness of the spider by means of different metrics, which are elaborated below.

4.1 EVALUATION METRIC

Confusion Matrix:

For evaluating the system's performance i.e. the spider program and the classifiers performance together, we chose four metrics the *precision, recall,* and *accuracy*. However, the definition of these metrics is considered from a confusion matrix's perspective.

We evaluated the accuracy of the classifier by comparing the results of the classifier for each of the documents in the test set with the truth file. All the files in the test set are classified into either taxonomic or non-taxonomic documents. The truth file is a

manually built file which contains the category of each of the documents in the test set. Ideally, the classifier should classify the test document into a category from which the document was actually selected. Here we have considered the possibility of using two classifiers; Rocchio and SVM (Support Vector Machines) to classify our documents.

A confusion matrix [14] is a matrix of both the predicted and the actual values of the classification. Shown below is the representation of a confusion matrix.

		Predicted	
		Positive	Negative
Actual	Positive	TP	FP
	Negative	TN	FN

Figure 8: Confusion Matrix

Where TP is a true positive, FN is a false negative.

4.1.1 Classifier Metrics

We evaluate the classifier based on the number of correct decisions it makes. In other words, how many documents get properly identified as taxonomic documents and how many non-taxonomic documents get rejected. This measured using accuracy where:

$$\text{Accuracy} = \frac{TP + TN}{(TP + FP + TN + FN)}$$

4.1.2 Spider Evaluation Metrics

We evaluate the overall process of spidering based on its ability to find the available relevant documents while rejecting the irrelevant ones. Recall measures the spider's ability to collect many taxonomic documents while precision measures the spider's ability to show the user only relevant documents.

$$recall = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$precision = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

As recall increases, precision generally declines, and vice versa. To have a single metric, we also evaluate the accuracy of the spider as a whole.

4.2 EXPERIMENT 1 – Evaluating the Classifier

This experiment is designed to evaluate the accuracy of SVM vs. Rocchio classifiers on PDF and html documents and because these documents differ from each other in length and content, they are tested separately.

4.2.1 Experiment 1a: PDF documents

Data Sets

The data sets used for this experiment were BioOne Journal Documents which are in PDF format. An algorithm has been designed to crawl the BioOne website *http://bioone.org*, by querying it with species names present in the TCS and downloading the documents. We collected a total of 700 PDF documents with an average length of 45,000 characters. All the documents collected were published in the Bioone Journal from 1993 to 2005. The training and testing data sets, with 50 and 40 documents respectively in each of the data sets, were randomly selected from the document collection. All the documents in the test set were classified manually to build the truth file with the document name and the category id.

Training Model

The number of training documents was varied from 5 to 50 in steps of 5. The number of documents in the test collection was set to 40 and the training and testing documents are randomly selected from the document collection. Training models were built using the same training sets in both the classifiers and the same test set was

used to evaluate their performance. While building the training models, we have considered the 50 highest weighted words. We ran experiments, not reported further here that varied the number of words per document from 50 through 200 and selected 50 as an the best performing solution. A stopword list was used to eliminate commonly used words when indexing. No stemming operations were performed when training the classifier.

Results

Pilot experiments, not repeated here, were conducted on PDF documents categorizing them into four different categories - taxonomic, non-taxonomic, phylogenic and false drops. The accuracy values of both the classifiers from these experiments are provided in the appendix. Since their accuracy values were almost the same as the results reported here, they are not discussed further. In this experiment, we considered only two categories, taxonomic and non-taxonomic, varied the training set size only, and compared the accuracy of the SVM and Rocchio classifiers. Table 1 below shows the results and they are shown graphically in Figure 9..

SVM and Rocchio Analysis (with 2 categories - Taxonomic and Non-taxonomic)

no. of docs per cat	5	10	15	20	25	30	35	40	45	50
SVM Accuracy (%)	75	60	67.5	77.5	57.5	60	52.5	57.5	70	55
Rocchio Accuracy (%)	35	57.5	50	50	57.5	62.5	52.5	60	70	62.5

Table 1: SVM and Rocchio Accuracies – PDF Documents

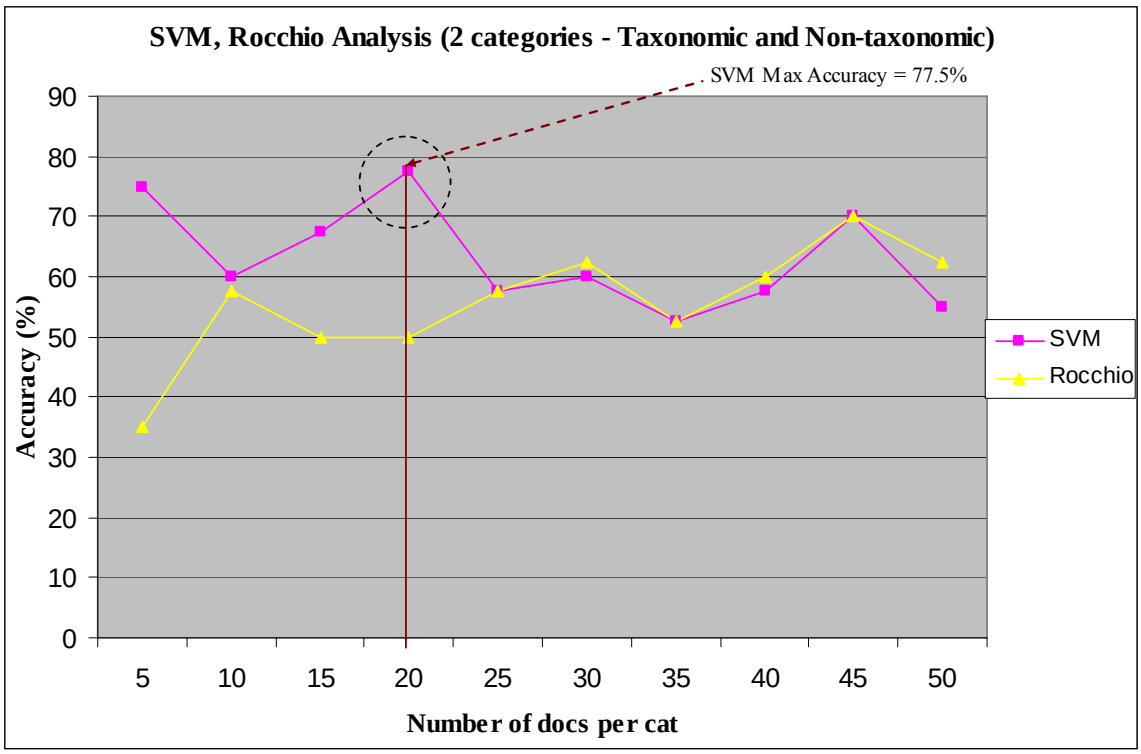


Figure 9: SVM, Rocchio Analysis – PDF Documents

Discussion:

The results show that the SVM classifier has given a maximum accuracy of 77.5% when the training set has 20 documents per category whereas the Rocchio classifier achieved a maximum accuracy of 70% when the training set contained 45 documents per category. It can also be observed that, with 25 or more training documents per category, the Rocchio and the SVM classifiers achieve almost the same accuracy .Based on these results, we concluded that the SVM classifier should be used to classify PDF documents collected by the Topic Specific Spider and it should be trained with 20 documents per category.

4.2.2 Experiment 1b: HTML documents

Data Sets

The data set used for this experiment was collected from Google Scholar. Using a software script, we issued broad domain specidic queries like “bats species”, “bats new species”, “bats new genus” etc., in order to collect both taxonomic and non-taxonomic data sets. These resulting URLs were followed and the associated HTML documents were collected and the off-topic documents were all manually filtered. A total of 150 documents were collected from which 50 training documents and 40 testing documents were randomly selected. Similar to the training document collection in experiment 1a), the training document collection in this experiment had equal number of taxonomic and non-taxonomic documents. The testing set, comprising of a collection of 40 documents, was also collected randomly and are different from the training data set. All the testing documents wee classified manually to build the truth file used for the evaluation process.

Training Model

As before, the training document set was varied from 5 to 50 documents per category in steps of 5. A stopword list was used to eliminate commonly used words and the 50 highest weighted words were used per document. No stemming operations were performed when training the classifier, but they were preprocessed to remove HTML markups.

Results

Table 2 below shows the evaluation of the both the classifiers with HTML documents classified into two categories – taxonomic and non-taxonomic. As before, the training set size was varied from 5 to 50 in steps of 5 and the test set size is 40 documents.

SVM and Rocchio Analysis (with 2 categories - Taxonomic and Non-taxonomic)

no. of docs per cat	5	10	15	20	25	30	35	40	45	50
Rocchio Accuracy (%)	52.5	67.5	72.5	82.5	80	80	85	80	85	82.5
SVM Accuracy (%)	55	77.5	57.5	77.5	82.5	82.5	82.5	82.5	82.5	82.5

Table 2: SVM and Rocchio Accuracies – HTML Documents

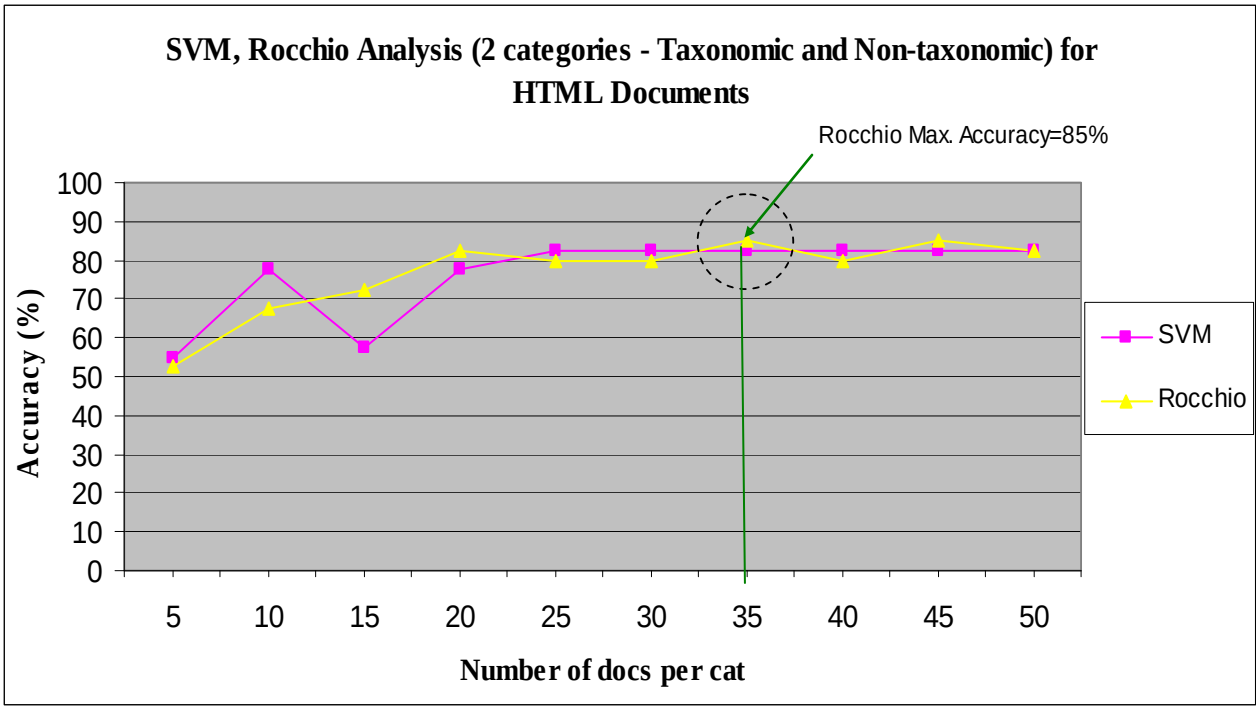


Figure 10: SVM and Rocchio Accuracies – HTML Documents

Discussion

The Rocchio classifier's accuracy increased with the number of training documents and reached a plateau after 20 documents per category at 82.5%. The SVM Classifier's accuracy showed a similar behavior and also plateaued at 82.5% accuracy. However, the results show that the Rocchio classifier gave a maximum accuracy of 85% when considering 35 documents per category whereas the SVM classifier gave a maximum accuracy of 82.5%. From this experiment, we thus conclude that in our system, when classifying html documents, we use the Rocchio classifier, although both classifiers are comparable.

4.3 EXPERIMENT 2 – Collecting Taxonomic Treatments

The goal of these experiments is to evaluate how well the Topic Specific Spider collects a large collection of taxonomic documents. For this experiment, we submitted a group of queries and evaluated the relevant and irrelevant information collected. For each of the iterations the spider iterates over, we log the no. of total documents collected, no. of documents that passed the filter and no. of documents which did not pass the filter. Based on these values and through manual review of the taxonomic documents collected by the spider, we analyze the performance of the spider.

4.3.1 Experiment 2a: Evaluating the quality of the documents collected by the Topic Specific Spider.

Procedure:

The main aim of this thesis is to help the taxonomist, focusing on particular genera, with enough taxonomic literature to update the genera. In this thesis, in order to test the effectiveness of our system from the user's perspective, we submitted a group of queries as seeds for the spider and evaluated the number of relevant and irrelevant documents collected. To test our system, we initiated the spider 7 times, beginning each run with a species specific query. The queries used were:

- Pteropus vampyrus (1- iteration only)
- Pteropus vampyrus phylogenetic
- Pteropus + "new genus"
- Bats+ "new species"
- Pteropus + chiroptera phylogenetic
- Pteropus + "phylogenetic"

For each of the queries, the spider collects the URLs when collecting each of the pages and parses them and collects the list of URLs leading out to other pages from within each page, checks whether the URL has already been crawled or not and also that it is not already present in the URL queue, and adds them to the queue of URLs which need to be crawled. The spider keeps crawling until the URL queue is empty,

this completes an iteration. After this the URLs extracted from the previous iteration are assigned to the URL queue and the process repeats.

For each of the iterations, we logged the number of total documents collected, the number of documents which passed the filter, the number of documents which did not pass the filter. The number of taxonomic documents collected was evaluated by manual review. We present detailed results for one representative query and summary of all the results. For the confusion matrices and the accuracy, recall and precision values for each of these individual queries, please refer to the Appendix.

Results: Tables 3 and 4 below show the precision and recall values for each of the iterations 0, 1, 2, and 3 for the documents collected before and after filtering. The results shown below are the average values considered over all the 7 queries.

Pre-Filtering

Iteration	# Taxonomic Documents	# Bad Documents	Recall* (%)	Precision (%)	Accuracy (%)
0	39	172	79.59	18.48	3.05
1	44	1070	89.80	3.95	6.32
2	49	2898	100.00	1.66	12.87
3	49	8183	100.00	0.60	39.46

Table 3: Pre-Filtering Performance Measures

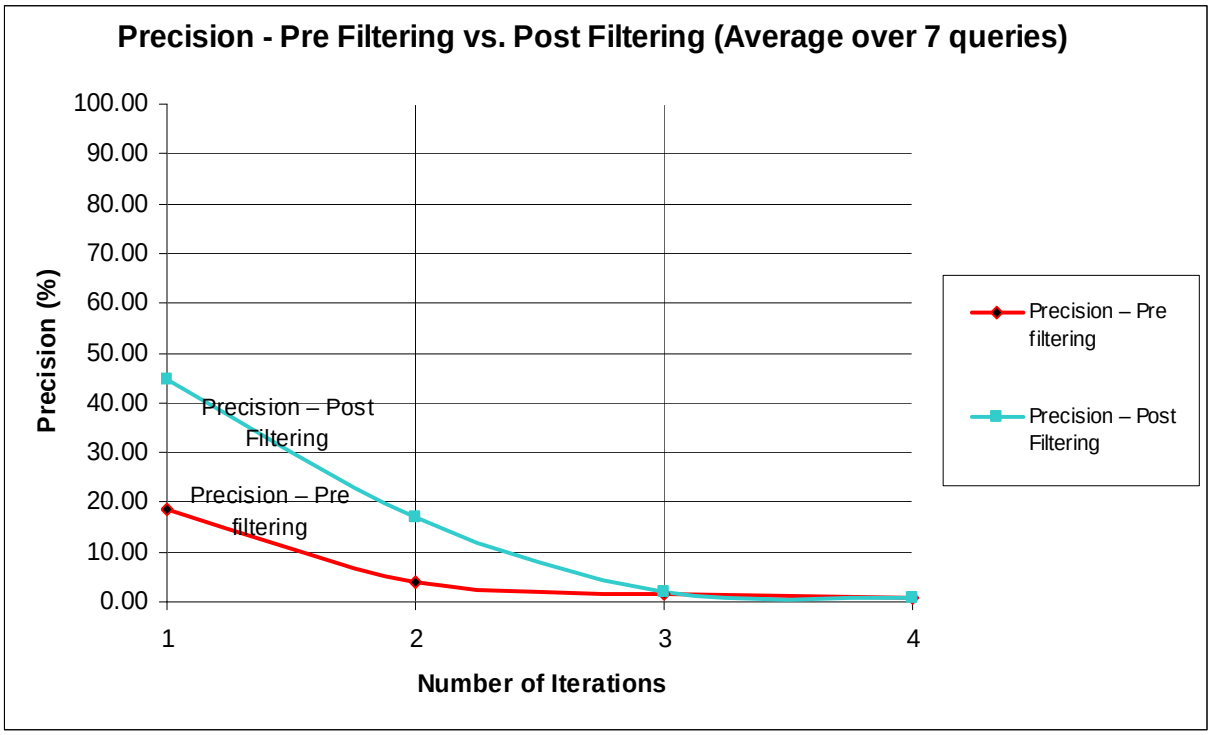
Post-Filtering

Iteration	# Taxonomic Documents	# Bad Documents	Recall* (%)	Precision (%)	Accuracy (%)
0	**17**	**21**	**68.00**	**44.74**	**7.55**
1	**22**	**107**	**88.00**	**17.05**	**15.65**
2	**25**	**1250**	**100.00**	**1.96**	**31.85**
3	**25**	**3510**	**100.00**	**0.71**	**97.67**

Table 4: Post-Filtering Performance Measures

	#Good	#Bad	Recall (%)	Precision (%)
iteration 0				
Pre-Filering	39	172	79.59	18.48
Post-Filtering	17	21	68.00	44.74
iteration 1				
Pre-Filering	44	1070	89.80	3.95
Post-Filtering	22	107	88.00	17.05
iteration 2				
Pre-Filering	49	2898	100.00	1.66
Post-Filtering	25	1250	100.00	1.96
iteration 3				
Pre-Filering	49	8183	100.00	0.60
Post-Filtering	25	3510	100.00	0.71

Table 5: Comparison of Pre/Post Filtering performances over all the 7 queries



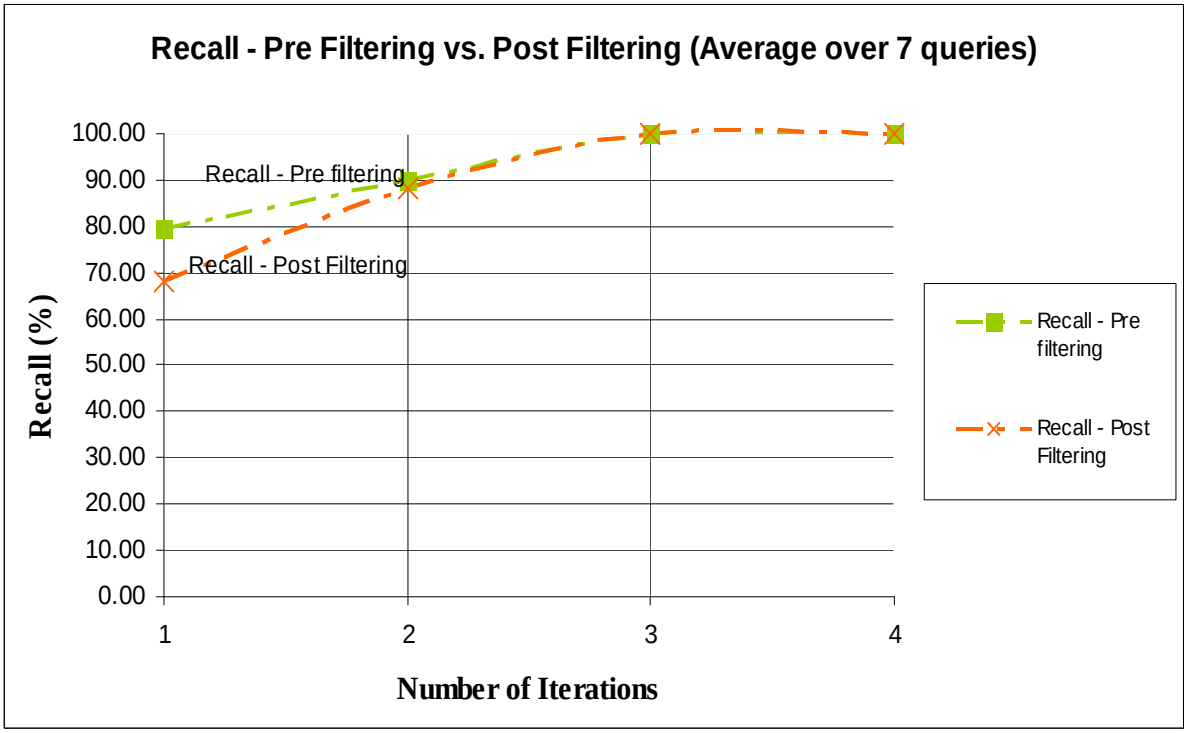


Figure 11: Performance Comparison – Precision/Recall Pre-filtering and Post-filtering averaged over all the 7 queries

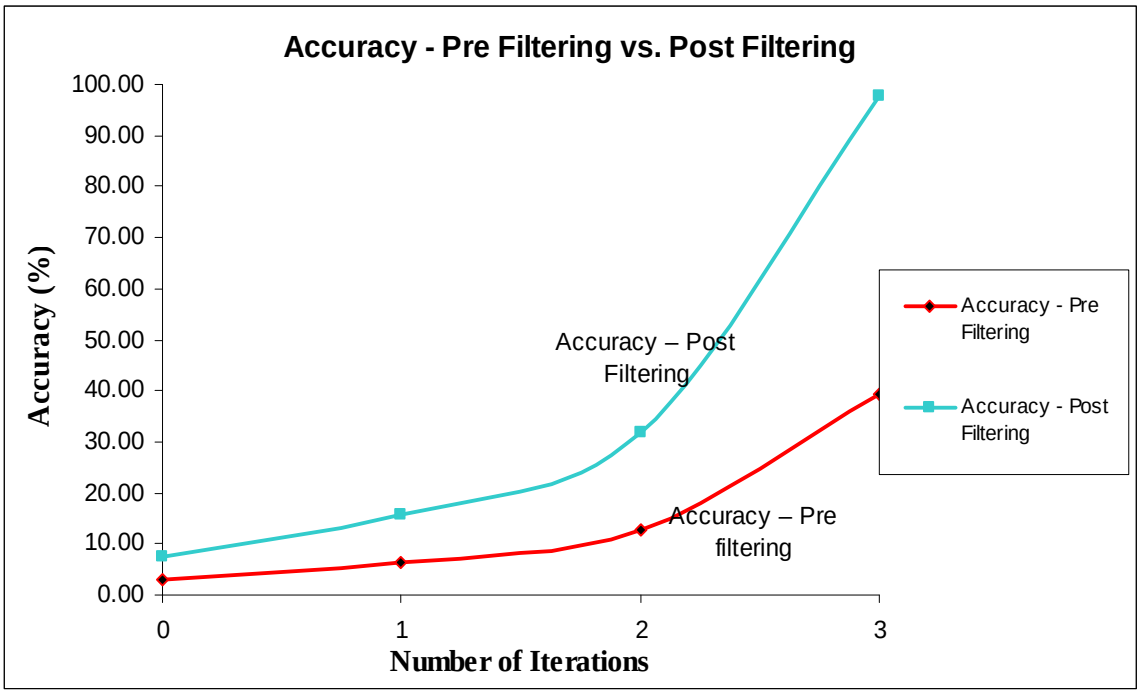


Figure 12: Performance Comparison – Accuracy Pre-Filtering vs. Post-Filtering

Discussion: Tables 3, 4 and 5 show a comparison of the performance measures averaged over all the 7 queries for each of the iterations. From these values, it can be concluded that the process of filtering removes a larger amount of irrelevant information when compared to the amount of relevant information lost. From the results shown in figure 11, it is evident that the precision values are higher in the case of post filtering than the precision values before filtering over all the iterations. The accuracy values, as shown in Figure 12, are higher in the case of post-filtering. Thus, filtering the documents during the process of crawling reduced the number of non-taxonomic documents to a large extent and increased the number of taxonomic documents retrieved.

4.3.2 Experiment 2b: The quality of pages collected will peak after some number of iterations

Procedure: The same procedure which was followed in hypothesis 1 is used here. The query which was considered to verify this hypothesis is "*Pteropus chiroptera phylogeny*", which is combination of the query term specified by the user, general expansion and a taxonomic expansion (order).

Results: The table below shows the evaluation of the precision, recall and accuracy values for each of the iterations 0, 1, 2, and 3 for the documents collected by the spider.

Metric	Iteration 0	Iteration 1	Iteration 2	Iteration 3
Precision (%)	**66.67**	**20.51**	**2.99**	**1.11**
Recall* (%)	**60**	**80**	**100**	**100**
Accuracy (%)	**7.55**	**15.65**	**31.85**	**97.67**

Table 6: Evaluation Metrics on Iterations 0 though 3

Discussion: From the plot shown in Figure 13, we see that the precision values tend to approach 0 and recall approaches 1 as the number of iterations increases. Informally, as the number of relevant documents retrieved increases, the amount of irrelevant information retrieved also increases. The results show that, after iteration 2,

there are no relevant documents retrieved and all the documents found are irrelevant. Thus, the crawler should stop after two iterations.

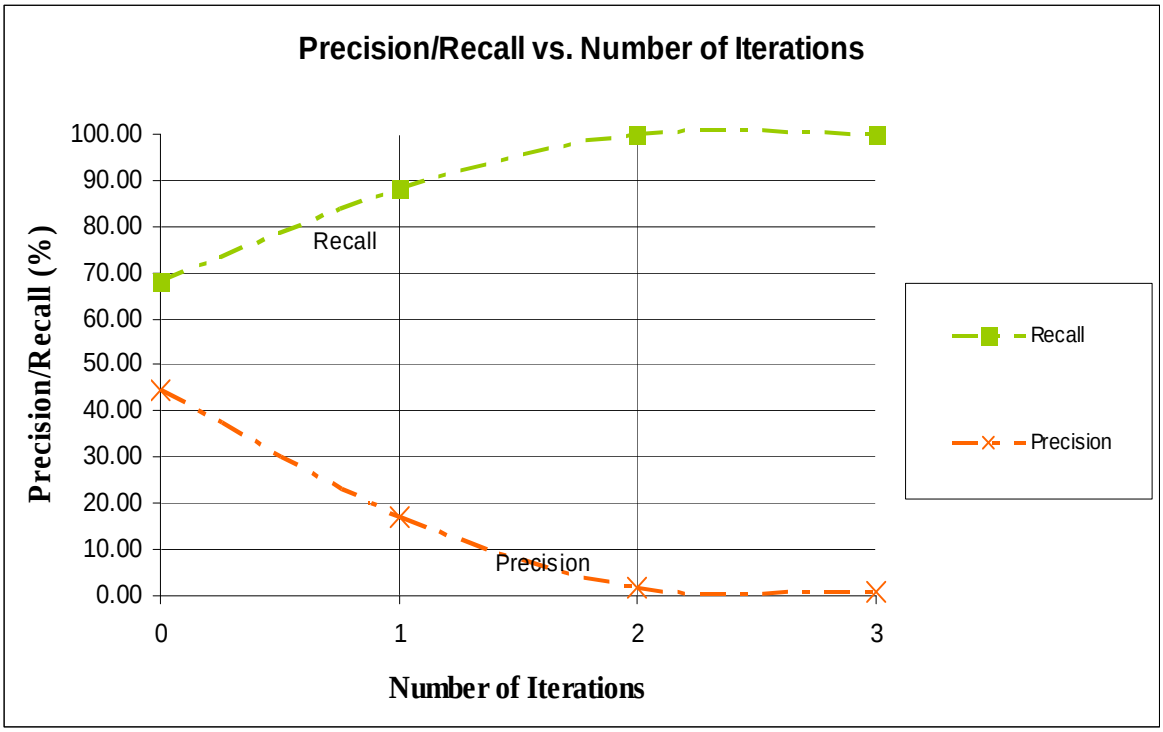


Figure 13: Precision/Recall* vs. Number of Iterations

4.3.3 Experiment 2c: Generically expanded queries retrieve more number of relevant documents than taxonomically expanded queries

Procedure: The spider program was run on two types of queries – generically expanded queries and taxonomically expanded queries, the latter one being a combination of the species name and genus name. The queries were run over three iterations and the document statistics for each of the queries are specified in the appendix. The following are the taxonomically and the generically expanded queries

considered and table 6 shows the average precision and recall values over all the queries for each of the query expansion techniques considered.

Taxonomically Expanded

Pteropus + "phylogenetic"

Pteropus + chiroptera phylogenetic

Pteropus Vampyrus

Pteropus vampyrus phylogenetic

Generically Expanded

Pteropus + "new genus"

Pteropus + "relationship species"

Bats+ "new species"

Results: The table below shows the evaluation of the precision and recall values for the total number of documents collected by the spider over all the iterations for each of the queries.

Measure	Generically expanded	Taxonomically Expanded
Precision (%)	**1.11**	**0.51**
Recall* (%)	**52.63**	**30.77**

Table 7: Performance Measures – General expanded queries vs. Taxonomically expanded queries

Discussion: The table above shows that the precision and recall values in the case of generically expanded queries are higher than those of the taxonomically expanded ones; the reason for this being that as more taxonomic terms are added to the query the search narrows down to a very domain-specific region. However, in the case of generically expanded queries, we are generalizing the query to a certain extent so that the search can be expanded and considering that all the irrelevant information can be filtered out by the filtering technique, this expansion technique works better. From these results it can hence be proved that generically expanded queries perform better when compared to the taxonomically expanded queries.

4.2.4 Rank Order Queries

The table 8 below shows each of the query types considered and they are ordered according to their precision values. A genus name coupled with a general term, "new genus" achieved the highest precision when compared to the others. But the number of documents retrieved for this query is less when compared to some of the other queries considered. This query however, tends to retrieve more relevant, top-ranked

documents when compared to the others based on its precision value. As the query tends to become more specific the precision values decreased.

Query name	Precision (%)	# Taxonomic documents
Pteropus* + *"new genus"	**1.75**	**5**
Pteropus* + *"phylogenetic"	**1.33**	**12**
Pteropus* + *chiroptera phylogenetic	**1.11**	**10**
Pteropus vampyrus phylogenetic	**0.51**	**4**
Bats*+ *"new species"	**0.48**	**6**
***Pteropus vampyrus* (1-iteration only)**	**50**	**3**
AVERAGE	**7.88**	**6**

Table 8: Queries rank ordered by precision values

Query name	Precision (%)	# Taxonomic documents
Pteropus vampyrus (1-iteration only)	**50**	**3**
Pteropus vampyrus phylogenetic	**0.51**	**4**
Pteropus + "new genus"	**1.75**	**5**
Bats+ "new species"	**0.48**	**6**
Pteropus + chiroptera phylogenetic	**1.11**	**10**
Pteropus + "phylogenetic"	**1.33**	**12**

Table 9: Queries rank ordered by number of taxonomic documents retrieved

Measure	**Generically expanded**	**Taxonomically Expanded**
Precision (%)	21.36	13.94
Recall* (%)	71.42	59.35

Table 10: Generically expanded vs. Taxonomically expanded

Table 9 shows the queries rank ordered by the number of taxonomic documents which passed through the filter. The query which is a combination of a genus name and a species name alone retrieved lesser number of documents than the same query concatenated with a general term like “phylogenetic”, which retrieved 4 documents. The difference between each of these queries is that the latter retrieved an extra document. However, the genus name alone with the term “phylogenetic” retrieved 12 documents which is the most returned by the queries considered. Table 10 shows the average recall and the precision values of all the generically and taxonomically expanded queries considered, and the precision values in the case of taxonomically expanded queries is higher when compared to the taxonomically expanded queries, which implies that the amount of relevant information obtained using generically expanded queries is more when compared to the latter.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this project, we have implemented the idea of using a topic specific spider to collect taxonomic treatments and evaluated the performance of the system developed. Also we considered two query expansion techniques, of which the generally expanded queries performed better than the taxonomically expanded queries. Also, we confirmed our hypothesis that filtering documents helped reduce the amount of non-taxonomic information which otherwise existed.

From the experiments conducted, it can be concluded that *post-filtering* the precision, recall, and accuracy values of the spider are higher when compared to *pre-filtering*, i.e., the process of text classification and spidering gave more relevant information. The optimal number of iterations at which the quality of pages is the highest was found to be two. As mentioned above, query expansion has been implemented in the system so as to enable us to consider more information, and it has been found that generically expanded queries performed better than taxonomically expanded queries.

5.2 Future Work

Although this thesis focused on the development of the topic specific spider for taxonomic treatments, there are many avenues which can be considered to further improve the performance of this system i.e., to increase the number of taxonomic documents and reduce the amount of non-taxonomic information. To cite a few

- ✓ Development of a user interactive system through which a taxonomist can obtain taxonomic treatments by submitting a query to the system.
- ✓ Consider other sources of seed URL's, not just one search engine as implemented in this project. The other possible sources can be BioOne, JSTOR, Yahoo search or other science portals.
- ✓ Decrease the number of documents that the end user needs to go through to identify taxonomic documents by increasing the performance of the filter by adding new features to the classifiers.
- ✓ Even before downloading a document calculate the similarity score between the query term and the summary.
- ✓ A way to access documents which are in the form of images and to convert them to text format for indexing
- ✓ Many web pages have only abstracts available for direct access and in order to view the full document we need a complete access. Such documents cannot be processed further i.e. in the next iterations the crawler just collects unnecessary information. Subscribe to more sources in order to access full text documents.

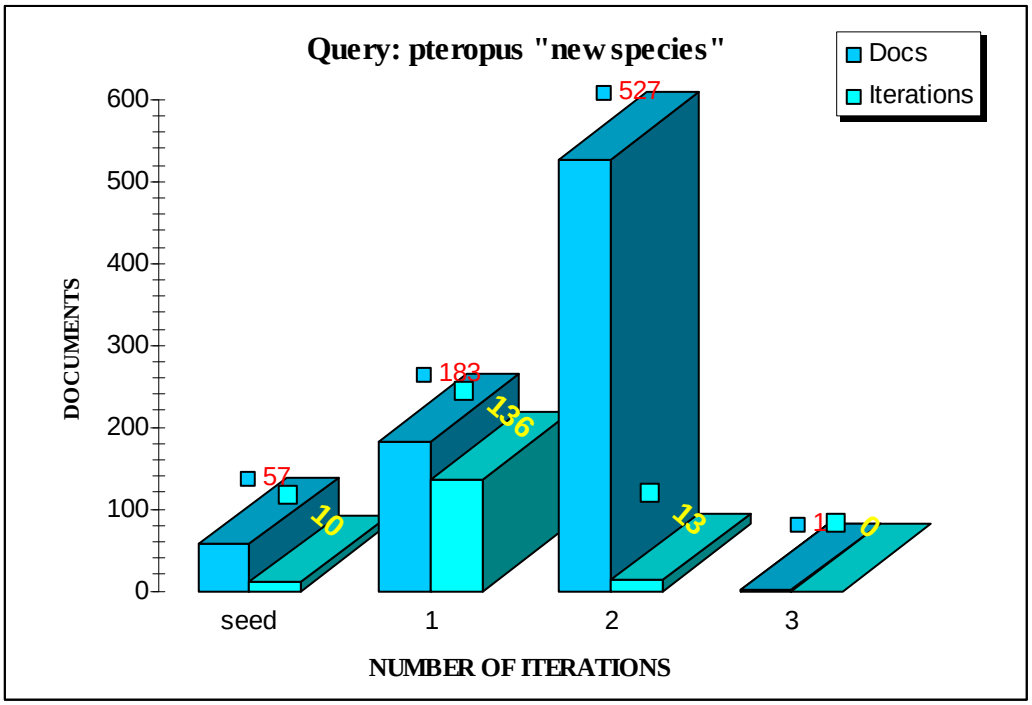
- ✓ Many journal documents are available in PDF formats which contain more information than normal html pages, a way to access them directly
- ✓ Some web pages are dynamic in nature and a technique should be developed to work around such pages
- ✓ Some pages contain the same journal document in each of them, thus the title of the document can be queued in order to avoid repetition of the same document.
- ✓ Exploit the html page structure and extract all the necessary information from it instead of downloading the entire document.

APPENDIX

MANUALLY GENERATED QUERIES

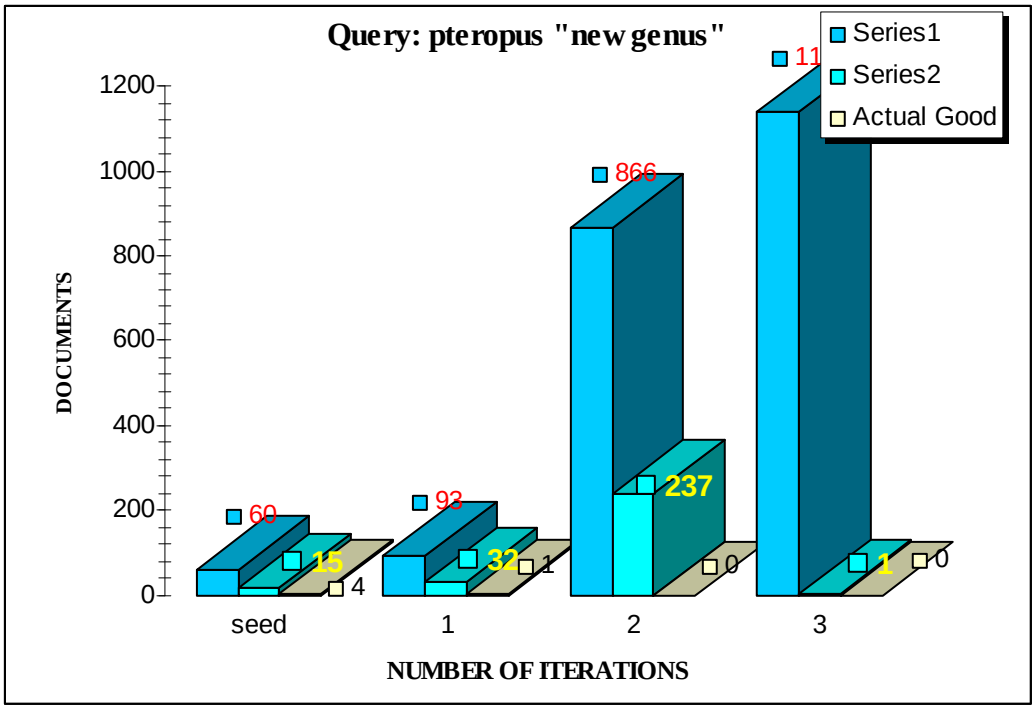
1) pteropus “new species”

ITERATIONS	NUMBER OF DOCUMENTS	
	Total	Taxonomic (Filtered by classifier)
seed	57	10
1	183	136
2	527	13
3	1	0



2) pteropus “new genus”

ITERATIONS	NUMBER OF DOCUMENTS		
	Total	Taxonomic (Filtered by classifier)	Actual Good (Filtered Manually)
seed	60	15	4
1	93	32	1
2	866	237	0
3	1141	1	0



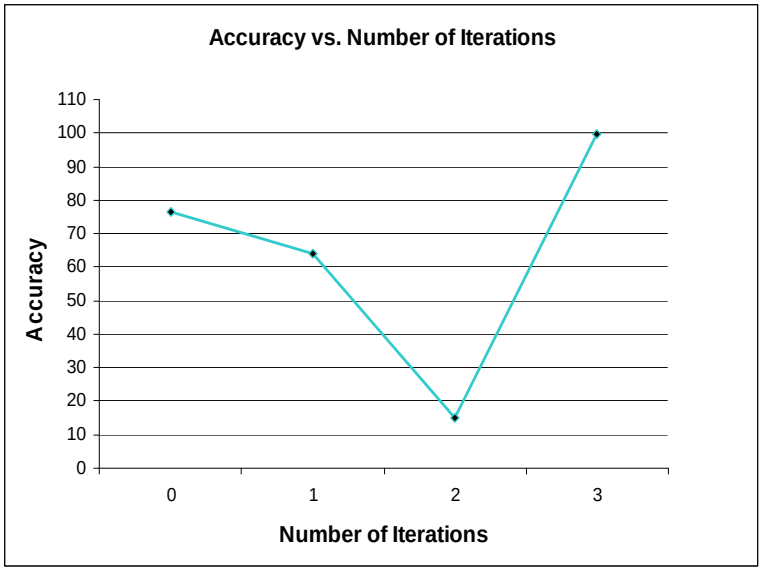
seed		Predicted	
		Negative	Positive
Actual	Negative	41	11
	Positive	3	4

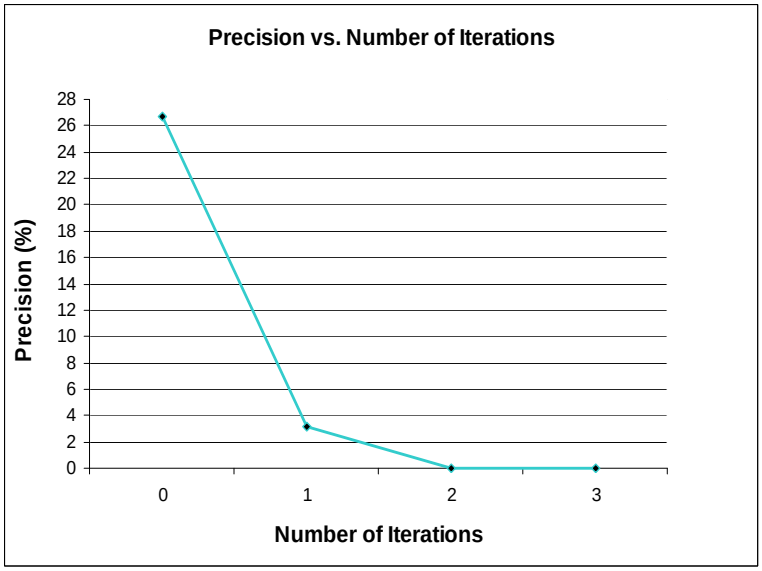
Iteration 1		Predicted	
		Negative	Positive
Actual	Negative	54	31
	Positive	0	1

Iteration 2		Predicted	
		Negative	Positive
Actual	Negative	42	237
	Positive	0	0

Iteration 3		Predicted	
		Negative	Positive
Actual	Negative	986	1
	Positive	0	0

Iteration	Accuracy	Precision	Recall
0	76.27119	26.66667	5.333333
1	63.95349	3.125	1
2	15.05376	0	NA
3	99.89868	0	NA





2) pteropus phylogenetic

ITERATIONS	NUMBER OF DOCUMENTS		
	Total	Documents Passing Filter	Taxonomic Documents
seed	87	9	6
1	118	30	2
2	494	296	2
3	1531	566	2

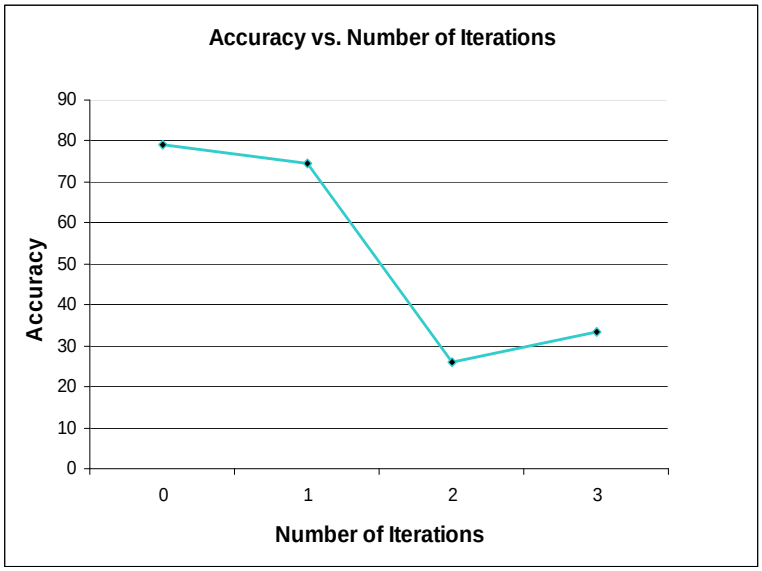
seed		Predicted	
		Negative	Positive
Actual	Negative	58	4
	Positive	13	7

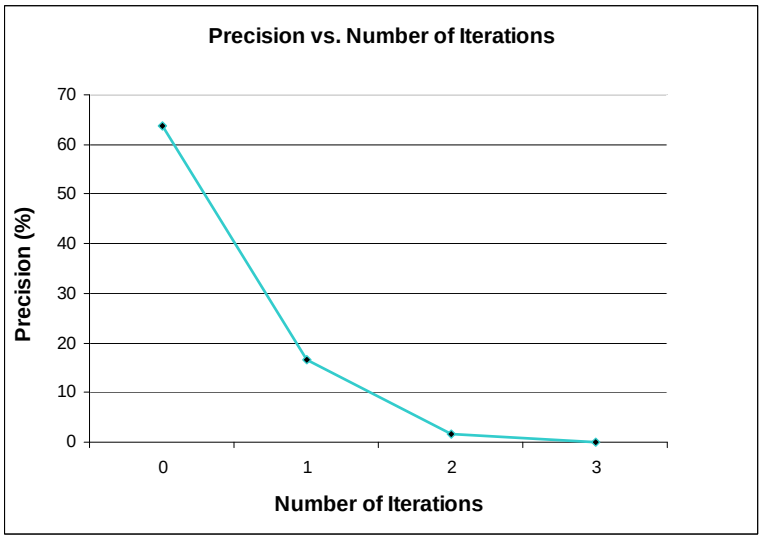
Iteration 1		Predicted	
		Negative	Positive
Actual	Negative	54	20
	Positive	0	4

Iteration 2		Predicted	
		Negative	Positive
Actual	Negative	108	323
	Positive	1	5

Iteration 3		Predicted	
		Negative	Positive
Actual	Negative	591	1181
	Positive	0	0

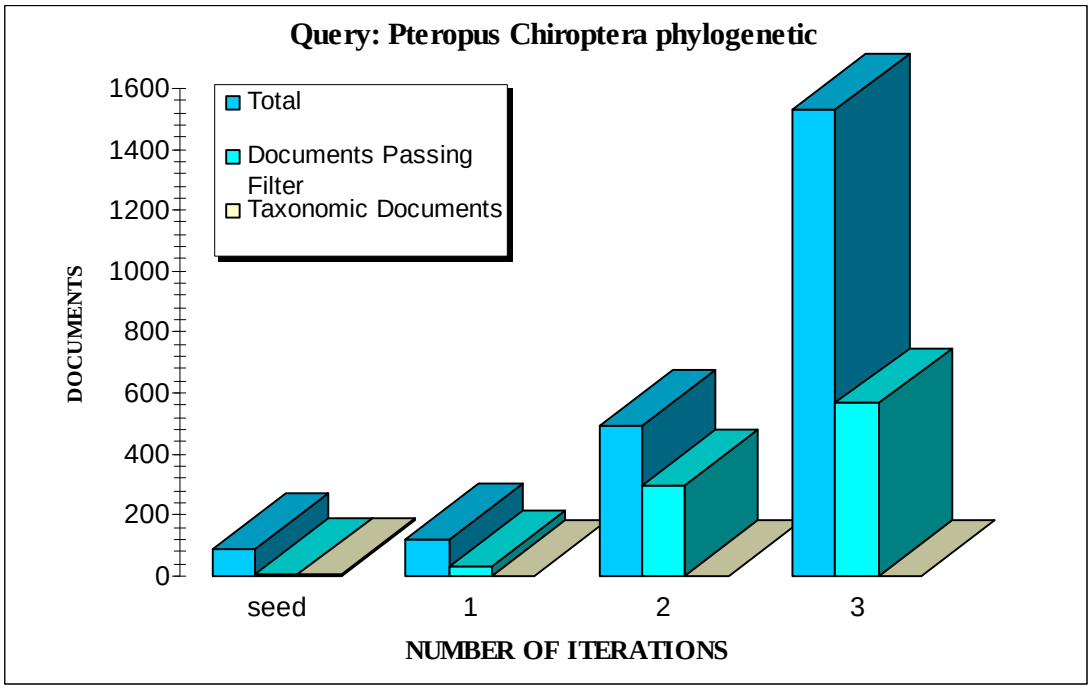
Iteration	Accuracy	Precision	Recall
0	79.26829	63.63636	7.538462
1	74.35897	16.66667	1
2	25.85812	1.52439	0.833333
3	33.35214	0	NA





3) Pteropus +Chiroptera phylogenetic

ITERATIONS	NUMBER OF DOCUMENTS		
	Total	Documents Passing Filter	Taxonomic Documents
seed	87	9	6
1	118	30	2
2	494	296	2
3	1531	566	0



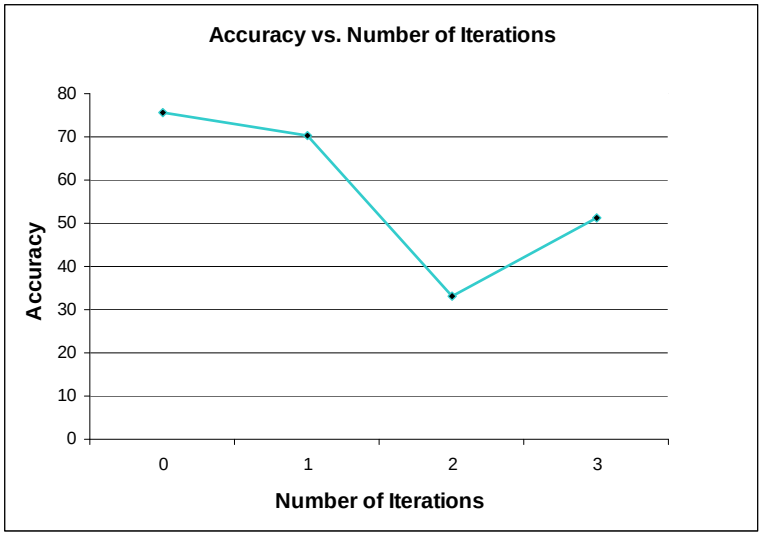
seed		Predicted	
		Negative	Positive
Actual	Negative	62	3
	Positive	19	6

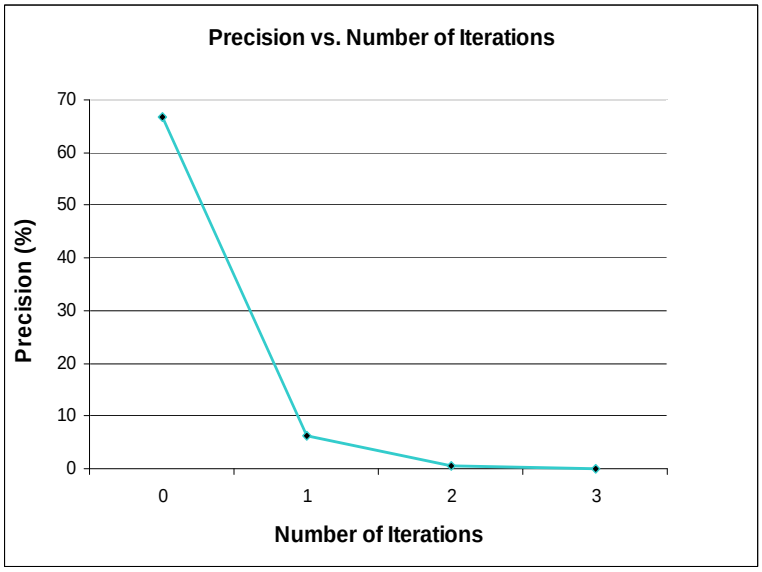
Iteration 1		Predicted	
		Negative	Positive
Actual	Negative	71	30
	Positive	1	2

Iteration 2		Predicted	
		Negative	Positive
Actual	Negative	144	296
	Positive	0	2

Iteration 3		Predicted	
		Negative	Positive
Actual	Negative	593	566
	Positive	0	0

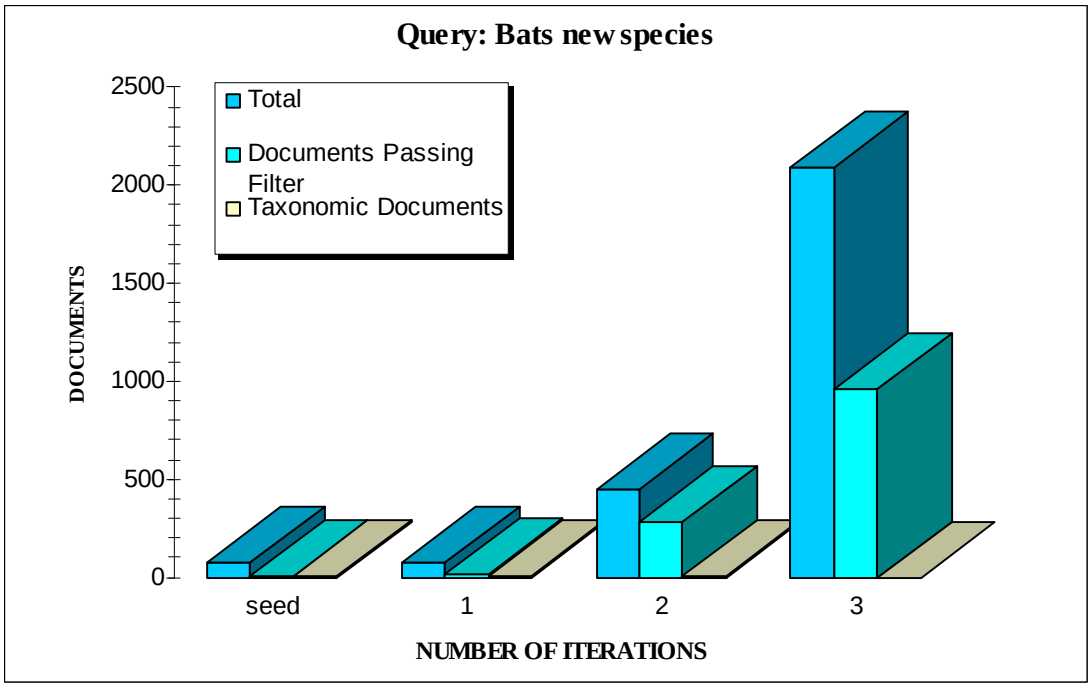
Iteration	Accuracy	Precision	Recall
0	**75.55556**	**66.66667**	**6.315789**
1	**70.19231**	**6.25**	**0.666667**
2	**33.03167**	**0.671141**	**1**
3	**51.1648**	**0**	**NA**





4) Bats + "new species"

ITERATIONS	NUMBER OF DOCUMENTS		
	Total	Documents Passing Filter	Taxonomic Documents
seed	76	9	4
1	77	12	1
2	449	282	1
3	2088	956	0



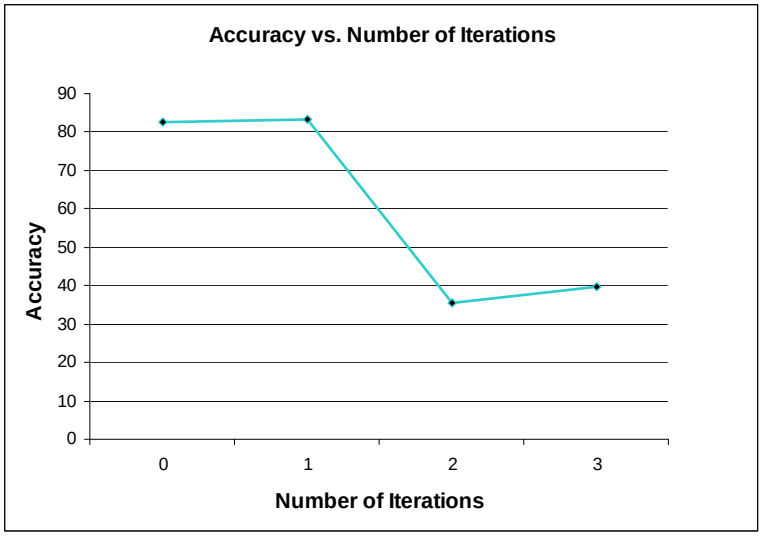
seed		Predicted	
		Negative	Positive
Actual	Negative	58	5
	Positive	8	4

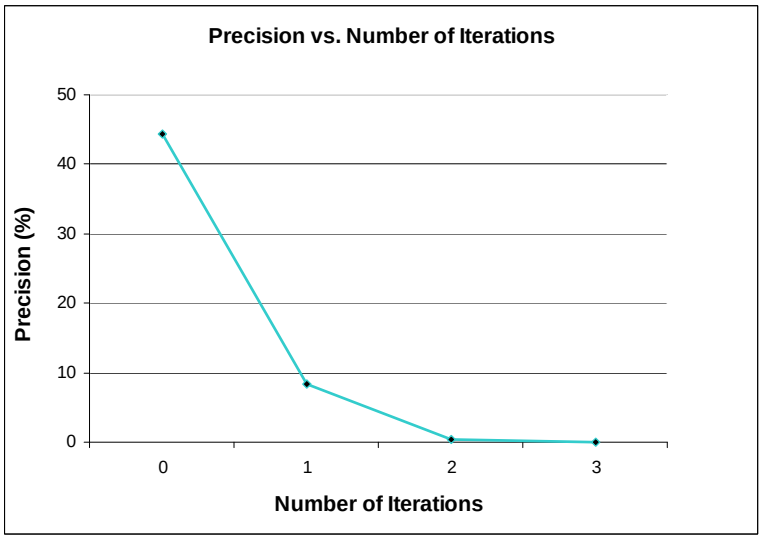
Iteration 1		Predicted	
		Negative	Positive
Actual	Negative	54	11
	Positive	0	1

Iteration 2		Predicted	
		Negative	Positive
Actual	Negative	153	281
	Positive	0	1

Iteration 3		Predicted	
		Negative	Positive
Actual	Negative	629	956
	Positive	0	0

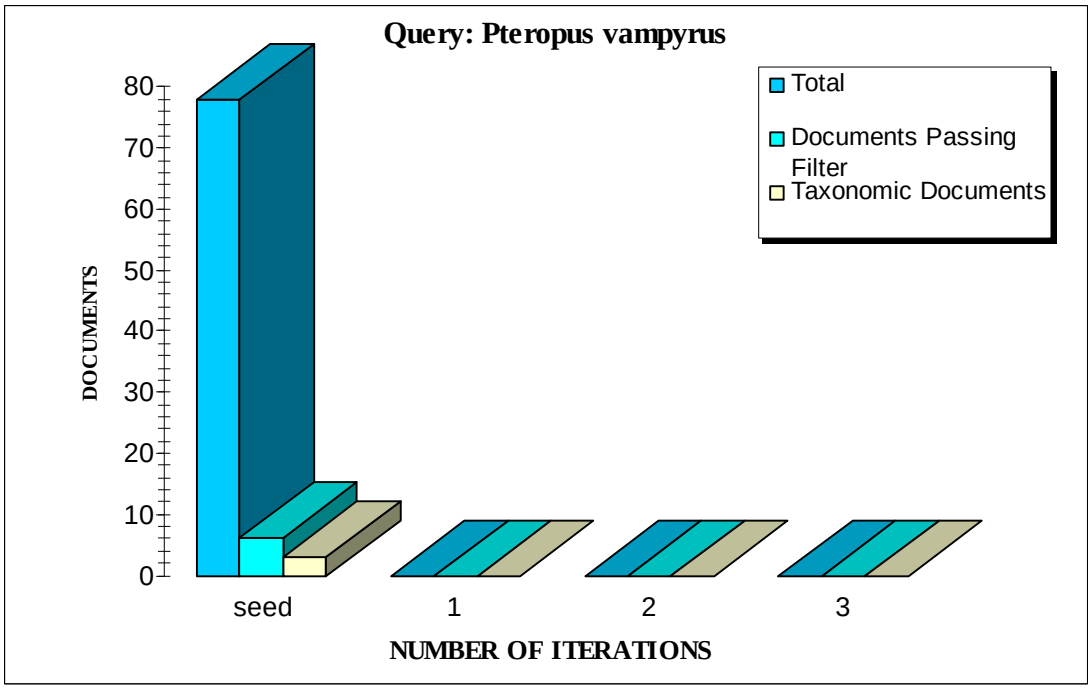
Iteration	Accuracy	Precision	Recall
0	**82.66667**	**44.44444**	**4.5**
1	**83.33333**	**8.333333**	**1**
2	**35.4023**	**0.35461**	**1**
3	**39.68454**	**0**	**NA**





5) Pteropus vampyrus

ITERATIONS	NUMBER OF DOCUMENTS		
	Total	Documents Passing Filter	Taxonomic Documents
seed	78	6	3
1	0	0	0
2	0	0	0
3	0	0	0



Seed		Predicted	
		Negative	Positive
Actual	Negative	78	6
	Positive	5	3

Iteration 1		Predicted	
		Negative	Positive
Actual	Negative	0	0
	Positive	0	0

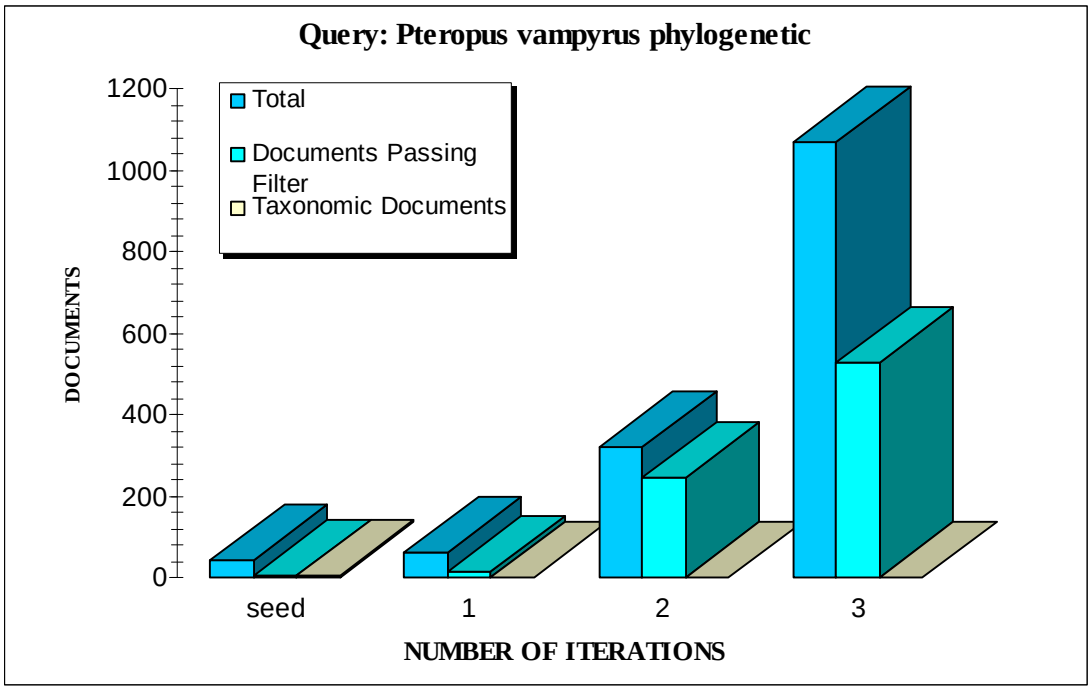
Iteration 2		Predicted	
		Negative	Positive
Actual	Negative	0	0
	Positive	0	0

Iteration 3		Predicted	
		Negative	Positive
Actual	Negative	0	0
	Positive	0	0

Iteration	Accuracy	Precision	Recall
0	88.04348	33.33333	3.6
1	#DIV/0!	#DIV/0!	#DIV/0!
2	#DIV/0!	#DIV/0!	NA
3	#DIV/0!	#DIV/0!	NA

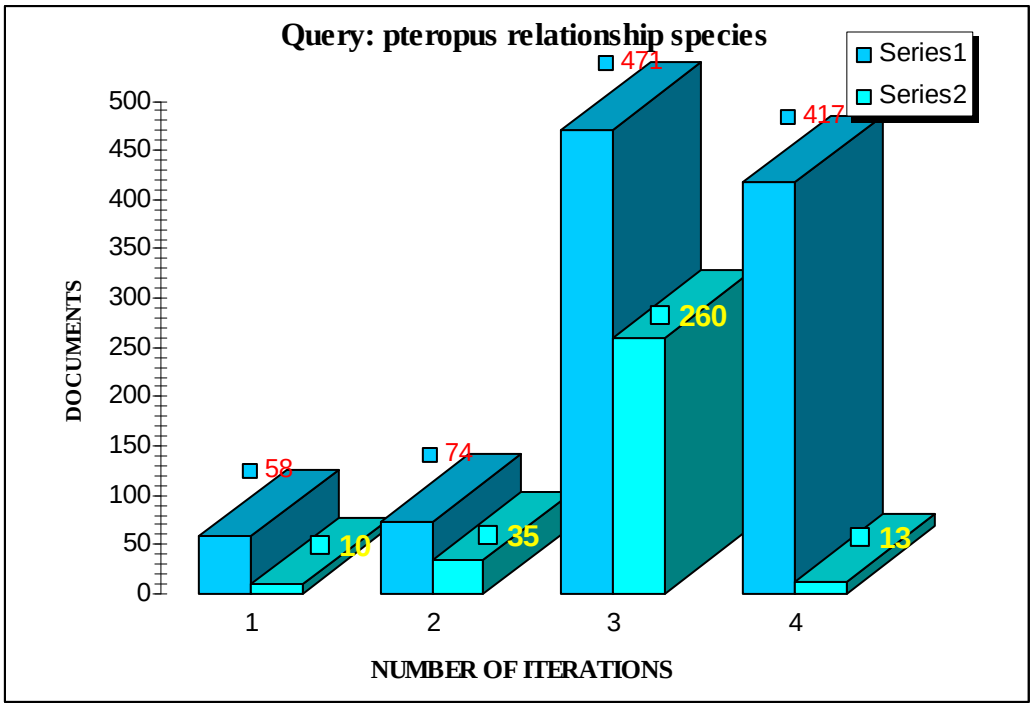
6) Pteropus vampyrus phylogenetic

ITERATIONS	NUMBER OF DOCUMENTS		
	Total	Documents Passing Filter	Taxonomic Documents
seed	43	5	3
1	63	12	1
2	322	244	0
3	1067	527	0



7) pteropus relationship species

ITERATIONS	NUMBER OF DOCUMENTS	
	Total	Taxonomic (Filtered by classifier)
seed	58	10
1	74	35
2	471	260
3	417	13



REFERENCES

[1] Haruechaiyasak, C.; Mei-Ling Shyu; Shu-Ching Chen. **Web document classification based on fuzzy association,** COMPSAC, 26th Annual International Conference, 2002, 487 – 492

[2] Guo G, Wang H, Bell D, Bi Y and Green Y (2004); **Using kNN Model for Automatic Text Categorization**, Journal of Soft Computing, Springer-Verlag Heidelberg.

[3] Li, Baoli; Yu, Shiwen; Lu, Qin, **An Improved k-Nearest Neighbor Algorithm for Text Categorization**, Proceedings of the 20th International Conference on Computer Processing of Oriental Languages, 2003

[4] A. McCallum and K. Nigam, **A comparison of event models for Naive Bayes text classification**, AAAI-98 Workshop on Learning for Text Categorization, 1998

[5] Chung Chang, and Chih-Jen Lin, **A Practical Guide to Support Vector Classification**, Department of Computer Science and Information Engineering National Taiwan University, 2000.

[6] Simon Tong and Daphne Koller, **Support Vector Machine Active Learning with Applications to Text Classification**, Proceedings of {ICML}-00, 17th International Conference on Machine Learning, 2000, 999-1006

[7] Yiming Yang, **An Evaluation of Statistical Approaches to Text Categorization**, Information Retrieval, Volume 1, 1999, 69-90

[8] Yiming Yang, Christopher G. Chute, **A Linear Least Squares Fit mapping method for information retrieval from natural language texts**, Proceedings of the 14th conference on Computational linguistics - Volume 2, Nantes, France, 1992, 447-453

[9] Hwee T. Ng and Wei B. Goh and Kok L. Low, **Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization**, Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval, 1997, 67-73

[10] G. Salton, **Introduction to Modern Information Retrieval**, MJ McGill, McGraw-Hill.

[11] Susan Gauch, Juan M. Madrid, Subhash Induri, Devanand Ravidran, and Sriram Chadalavada, **Keyconcept: A Conceptual Search Engine**, Information and

Telecommunication Technology Center, Technical Report,: ITTC-FY2004-TR-8646-37, University of Kansas.

[12] Dominic Widdows, **Geometry and Meaning - Companion website**, CSLI Publications

[13] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, **A Practical Guide to Support Vector Classification**, Department of Computer Science and Information Engineering, National Taiwan University, 2000.

[14] **Machine Learning**, SpringerLink - Computer Science and Engineering, 2002 ISSN 0885-6125

[15] Viv Cothey, **Web-crawling reliability**, Journal of the American Society for Information Science and Technology, 2004, 1228 – 1238.

[16] Wikipedia – http://www.wikipedia.org

[17] Cho, J.; Garcia-Molina, H.; Page, L., **Efficient Crawling Through URL Ordering**, Proceedings of 7th World Wide Web Conference, 1998

[18] Marc Najork, Janet L. Wiener , **Breadth-first Search Crawling Yields High-Quality Pages**, Compaq Systems Research Center, 2001

[19] Cho, J. and Garcia-Molina, H. **Synchronizing a database to improve freshness**, Proceedings of ACM International Conference on Management of Data (SIGMOD), 2000, 117-128.

[20] M. Koster, **A Standard for Robot Exclusion**, http://www.nexor.co.uk/mak/doc/robots/norobots.html, 1994.

[21] Cho, J. and Garcia-Molina, H. (2002). **Parallel crawlers**. Proceedings of the eleventh international conference on World Wide Web (ACM), pages 124–135.

[22] Filippo Menczer, Gautam Pant, Padmini Srinivasan, Miguel E. Ruiz, **Evaluating topic-driven web crawlers**, Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 24th annual international ACM SIGIR conference, 2001, 241 – 249.

[23] Yiqiao Wang, Eleni Stroulia, **Designing Efficient Topic-Driven Web Crawlers,** Computing Science Department, University of Alberta, AB T6G 2H1.

[24] Soumen Chakrabarti, Martin van den Berg and Byron Dom, **Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery**, WWW Conference, 1999.

[25] **Enabling the Science Environment for Ecological Knowledge (SEEK)**, National Science Foundation Proposal, ITR Collaborative Research, 2002

[26] Steve R. Gunn, **Support Vector Machines for Classification and Regression**, Technical Report, University of Southampton, 1998

[27] Vandana Priyadarshini Samala, **Information Extraction From Taxonomic Literature**, University of Kansas, 2007