

A Novel Hybrid Focused Crawling Algorithm to Build Domain-Specific Collections

Yuxin Chen

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Advisory Committee:

Edward A. Fox, Chair
Weiguo Fan
Chang-Tien Lu
Naren Ramakrishnan
Ricardo da Silva Torres

February 5, 2007
Blacksburg, Virginia

Keywords: digital libraries, focused crawler, classification, meta-search.

Copyright© 2007 by Yuxin Chen
ALL RIGHTS RESERVED

A Novel Hybrid Focused Crawling Algorithm to Build Domain-Specific Collections

Yuxin Chen

Abstract

The Web, containing a large amount of useful information and resources, is expanding rapidly. Collecting domain-specific documents/information from the Web is one of the most important methods to build digital libraries for the scientific community. Focused Crawlers can selectively retrieve Web documents relevant to a specific domain to build collections for domain-specific search engines or digital libraries. Traditional focused crawlers normally adopting the simple Vector Space Model and local Web search algorithms typically only find relevant Web pages with low precision. Recall also often is low, since they explore a limited sub-graph of the Web that surrounds the starting URL set, and will ignore relevant pages outside this sub-graph. In this work, we investigated how to apply an inductive machine learning algorithm and meta-search technique, to the traditional focused crawling process, to overcome the above mentioned problems and to improve performance. We proposed a novel hybrid focused crawling framework based on Genetic Programming (GP) and meta-search. We showed that our novel hybrid framework can be applied to traditional focused crawlers to accurately find more relevant Web documents for the use of digital libraries and domain-specific search engines. The framework is validated through experiments performed on test documents from the Open Directory Project [22]. Our studies have shown that improvement can be achieved relative to the traditional focused crawler if genetic programming and meta-search methods are introduced into the focused crawling process.

Acknowledgements

First and foremost, I give special thanks to my advisor, Professor Edward A. Fox, for his guidance, encouragement, and generous support during my PhD study. This dissertation could not be finished without his constant help and warm support.

I also give heartfelt thanks to my other committee members: (in alphabetical order) Dr. Weiguo Fan, Dr. Chang-Tien Lu, Dr. Naren Ramakrishnan, Dr. Ricardo Torres, for their thoughtful feedback and comments and careful review of this dissertation.

In addition, I thank members of the Digital Library Research Laboratory for their useful comments and suggestions. Special thanks go to Ryan Richardson and Aaron Krowne for their great help with this research work. I also appreciate a key collaborator in this work, Marco Cristo, for his huge help and support with SVM.

Finally, I would like to thank my parents and my lovely wife, Baoping Zhang. Their continuous support is a key factor to my success.

Contents

1 Introduction	1
1.1 Problem and Motivation	1
1.2 Objectives, Broad Impacts, and Contributions	4
1.2.1 Objectives.....	5
1.2.2 Broad Impacts and Contributions.....	5
1.3 Organization of this Work	7
2 Related Work	8
2.1 Focused Crawler	8
2.1.1 Web Analysis Algorithms	8
2.1.2 Web Search Algorithms.....	9
2.2 Support Vector Machines in Text Classification	11
2.3 GA/GP in Information Retrieval.....	12
2.4 Meta-search.....	14
2.5 CITIDEL.....	15
2.6 NDLTD.....	17
3 Research Problems	20
3.1 Limitations of Current Web Analysis Algorithms.....	20
3.2 Shortcomings of Local Search Algorithms.....	21
4 Approach	27
4.1 Similarity Measures	27
4.2 Combining Evidence.....	28
4.2.1 Definition of GP-based Classification Problem	29
4.3 Inductive learning Method - GP	30
4.3.1 Machine Learning: Genetic Programming.....	30
4.3.2 GP System Configuration	35
4.3.3 Fitness Function.....	36
4.3.4 Classification Framework.....	36
4.3.5 Properties of Classification Framework.....	38
4.4 Meta-search.....	38

4.4.1	<i>Construction of Query Terms</i>	39
4.4.2	<i>Choice of Search Engines</i>	40
4.5	Rationale of using GP and Meta-search.....	40
4.6	Focused Crawling Framework.....	41
5	Experiments	46
5.1	Test Data Collection.....	46
5.2	Experimental Data Set Design.....	48
5.3	Baselines.....	49
5.4	Experimental Set Up.....	50
5.5	Starting URL Set.....	50
5.6	Experimental Results.....	51
5.7	Computation Time and Space Analysis.....	60
6	Conclusions and Future Work	63
6.1	Conclusions.....	63
6.2	Future Work.....	64
	Bibliography	67

List of Figures

Figure 1: Differences between standard crawling and focused crawling	3
Figure 2: Support Vector Machines (SVMs) find the <i>hyperplane</i> h , which separates the <i>positive</i> and <i>negative</i> training examples with maximum distance. The examples closest to the hyperplane are called Support Vectors (marked with <i>circles</i>).....	12
Figure 3: A snapshot of MetaCrawler.....	15
Figure 4: A snapshot of CITIDEL	17
Figure 5: A snapshot of NDLTD	19
Figure 6: Relevant Web pages with co-citation relationships missed by focused crawlers	23
Figure 7: Relevant Web communities separated by non-relevant Web pages.....	24
Figure 8: A focused crawler trapped within the initial Web community	25
Figure 9: A sample tree representation.	31
Figure 10: A graphical illustration of crossover operation [101]	33
Figure 11: A graphical illustration of mutation operation	34
Figure 12: Architecture of proposed meta-search framework	39
Figure 13: Design of proposed focused crawling framework.....	43
Figure 14: Sample size vs. training time.....	61
Figure 15: Sample size vs. training matrices size	62

List of Tables

Table 1: Types of evidence	29
Table 2: Essential GP components	35
Table 3: Configurations of GP used for similarity function discovery.....	35
Table 4: Number of documents under each category of ODP	47
Table 5: Number of documents in data sets.....	49
Table 6: GP system experimental settings	50
Table 7: Content of starting URL set.....	51
Table 8: Macro F1 comparison between GP, content-based SVM, and combination-based SVM on the 10% sample	53
Table 9: Macro F1 comparison between GP, content-based SVM, and combination-based SVM on the 20% sample	53
Table 10: Macro F1 comparison between GP, content-based SVM, and combination-based SVM on the 30% sample	54
Table 11: Macro F1 gains of GP over content-based SVM and combination-based SVM	54
Table 12: Macro F1 comparison between GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 10% sample.....	56
Table 13: Macro F1 comparison between GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 20% sample.....	56
Table 14: Macro F1 comparison between GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 30% sample.....	57
Table 15: Macro F1 gains of GP over content-based SVM and combination-based SVM on 500 crawled Web pages	57
Table 16: Macro F1 comparison between GP + meta-search, GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 10% sample	58
Table 17: Macro F1 comparison between GP + meta-search, GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 20% sample	58

Table 18: Macro F1 comparison between GP + meta-search, GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 30% sample	59
Table 19: Macro F1 gains of GP + meta-search over GP, content-based SVM, and combination-based SVM on 500 crawled Web pages	59
Table 20: Computation time and space requirement	61

Chapter 1

1 Introduction

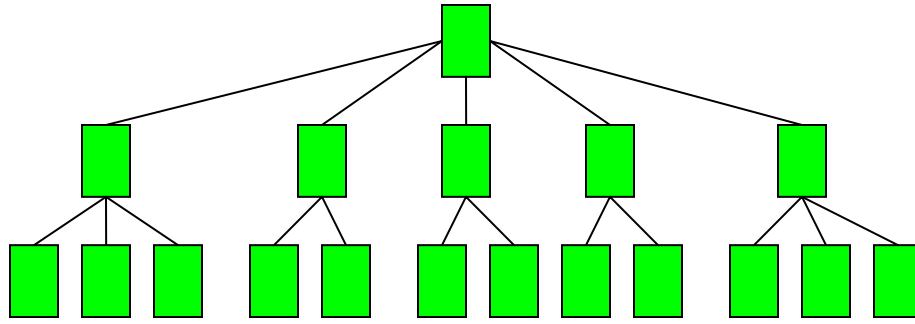
A focused crawler (also known as a topical crawler) is a program or automated script which attempts to download Web pages that are similar to each other. In recent years, automated focused crawling has attracted considerable interest in industry and academia, due to the increasing emergence of digital libraries and domain-specific search engines.

1.1 Problem and Motivation

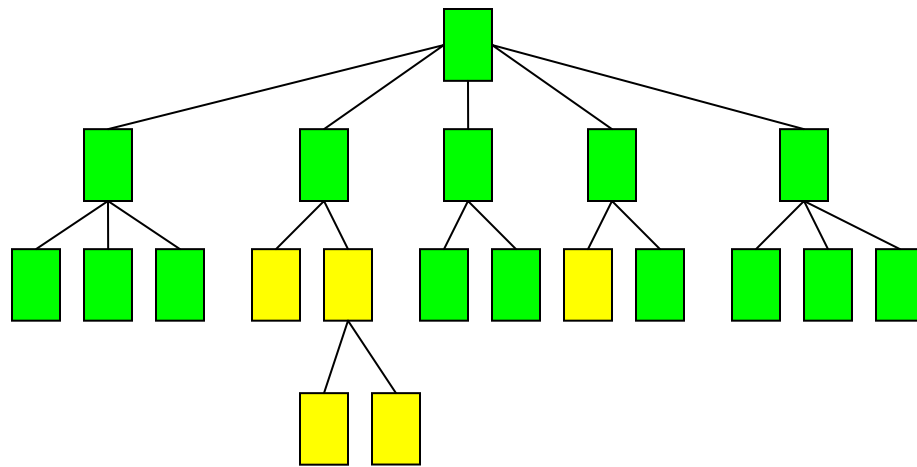
Being already the largest information source in the world, the World Wide Web (WWW) is still expanding rapidly. Nowadays, millions of people are seeking information on it, and search engines play a very important role during this process. A Web crawler is a key component inside a search engine. It can traverse (a portion of) the Web space by following Web pages' hyperlinks and storing the downloaded Web documents in local repositories that will later be indexed and used to respond to the users' queries efficiently. However, with the huge size and explosive growth of the Web, it becomes more and more difficult for search engines to provide effective services to end-users. Moreover, such a large collection often returns thousands of result documents in response to a single query. Browsing many documents to find the relevant ones is time-consuming and tedious.

The indexable Web has more than 11.5 billion pages. Even Google, the largest search engine, has only 76.16% coverage [46]. About 7 million new pages go online each day. It is impossible for major search engines to update their collections to meet such rapid growth. As a result, end-users often find the information provided by major search engines not comprehensive or out-of date.

To address the above problems, domain-specific search engines were introduced, which keep their Web collections for one or several related domains. Focused crawlers were used by the domain-specific search engines to selectively retrieve Web pages relevant to particular domains to build special Web sub-collections, which have smaller size and provide search results with high precision. The general Web crawlers behind major search engines download any reachable Web pages in *breadth-first* order. This may cause heavy network traffic. On the other hand, focused crawlers follow Web links that point to the relevant and high-quality Web pages. Those Web links that point to non-relevant and low-quality Web pages will never be visited. The above-mentioned major differences between general Web crawlers and focused crawlers are summarized in Figure 1.



(a) Standard Crawling



(b) Focused Crawling



Relevant page



Non-relevant page

Figure 1: Differences between standard crawling and focused crawling

In Figure 1(a), a standard crawler follows each link, typically applying a breadth-first search strategy. If the crawler starts from a document which is i steps from a target document, all the documents that are up to $i-1$ steps from the starting document must be downloaded before the crawler hits the target. In Figure 1(b), a focused crawler tries to identify the most promising links, and ignores off-topic documents. If the crawler starts from a document which is i steps from a target document, it downloads a small subset of

all the documents that are up to $i-1$ steps from the starting document. If the search strategy is optimal, the crawler takes only i steps to discover the target.

Most focused crawlers use simple adaptations of the vector space model to judge the relevance of Web pages, and local search algorithms such as *best-first* (see Section 2.1.2) to determine the collecting order in which the target links are visited. Some problems caused by traditional focused crawlers arise as more and more knowledge about Web structure has been revealed through Web structural studies. These problems could result in low-quality domain-specific collections, which would degrade the performance of domain-specific search engines.

Another motivation of this work is to enhance teaching and learning in the computing field, by enriching the CITIDEL system [15], part of the National Science Digital Library (NSDL) [72]. High-quality Web collections retrieved by a novel hybrid focused crawler could be loaded into the CITIDEL system [15] to increase the number of online computing-related resources.

The work presented here aims to overcome the problems mentioned above with a novel hybrid framework including a machine learning algorithm and meta-search technique. In particular, we investigate an inductive learning method – Genetic Programming – for the discovery of better similarity functions to be used in the focused crawler, and explore how this framework can be used to improve focused crawling performance.

1.2 Objectives, Broad Impacts, and Contributions

The proposed work combines an inductive machine learning algorithm – through Genetic Programming (GP) – and meta-search technology, to improve focused crawling performance, overcoming the problems discussed above. The evidence used by GP comes from the structural content information of Web pages. The purpose of this research is to investigate whether and how the proposed hybrid focused crawling framework can

produce better results than other traditional content-based best-first search focused crawlers.

1.2.1 Objectives

Our proposed work tries to apply a hybrid framework to the focused crawling process to find more relevant Web resources in order to overcome the problems faced by traditional focused crawlers. Our goal is to create a solution which is extensible, flexible [39], integrated, efficient, and effective. Our objectives include:

1. Discovering the most effective Web analysis algorithm that solves difficult but common problems in Web page classification, confirming that GP is the best solution through extensive experiments.
2. Discovering the most effective Web search algorithm by introducing meta-search technology to find more relevant Web resources, overcoming the problems that plague traditional best-first search algorithms.
3. Enhancing the effectiveness of CITIDEL [15], thence of NSDL [72], and of other systems like NDLTD [70], by importing the computing-specific collection retrieved by the hybrid focused crawler.
4. Enhancing the effectiveness of domain-specific search engines that are based on domain-specific collections, like CITIDEL [15] or NDLTD [70].

1.2.2 Broad Impacts and Contributions

This research work produces basic findings concerning the effectiveness and efficiency of the proposed hybrid focused crawling framework which aims to solve the problems faced by traditional focused crawlers. The algorithms developed for this research should be generally understandable, and capable of being implemented in a wide variety of information applications. This research would have big implications on several

information retrieval areas including focused crawling, searching, and teaching and learning, as described in more detail below:

1. Theory: We produce a hybrid focused crawling framework, and apply it to the traditional focused crawling process. Focused crawlers built upon this framework help build high-quality domain-specific Web collections, better than those from traditional focused crawlers.
2. Crawling: Our approach can help focused crawlers find more relevant Web pages to satisfy the diverse information needs of information seekers, including to have access to comprehensive collections.
3. Searching: By building high-quality domain-specific collections, our work can help end-users get correct and comprehensive results from domain-specific search engines.
4. Categorization: Our research also contributes to the data categorization field. The hybrid focused crawling algorithm implemented in our research can be used to automatically retrieve records related to specific academic majors like physics or computing, e.g., from NDLTD [70] collections.
5. Teaching and Learning: Our research can enhance teaching and learning in the computing-related fields. A computing-specific collection generated by the hybrid focused crawler can be imported into the CITIDEL system [15] in order to provide more comprehensive online computing resources to teachers and students.

The techniques we develop should be applicable to other content domains where high-quality searching and browsing services are essential to the end-users. Published results of our work can have beneficial impact on the teaching and learning community, as well as academic institutions and commercial entities. With the fast growing Web, it has become increasingly important to build high-quality domain-specific search engines, and our research will give assistance to this goal via building high-quality domain-specific collections. Delivery of open source code and educational materials, as well as testbeds, can be imported into the resource bases for our Open Digital Library project [90],

OCKHAM project [68], and ETANA project [79, 80] – to stimulate teaching, learning, and research in this area.

1.3 Organization of this Work

In this chapter, we gave some background on focused crawlers and introduced the problems and motivation of this research work. Chapter 2 discusses research related to our work. Chapter 3 describes problems associated with traditional focused crawlers and some potential solutions.

In Chapter 4 we discuss the approach to implement our novel hybrid focused crawling framework including classification framework and meta-search technique. Chapter 5 presents experimental designs and results. It discusses our testbed, experimental data set design, and baselines. It also presents the experimental results on the testbed and the comparison between our approach and other technologies.

Finally, in Chapter 6 we give conclusions and suggestions regarding future research activities to be carried out to solve the problems left open by this dissertation.

Chapter 2

2 Related Work

In this chapter, we review works related to focused crawlers, text classification, and meta-search. In Section 2.1 we review traditional focused crawlers, Web analysis algorithms, and Web search algorithms. Support Vector Machines have been extensively used for classification, and serve as one evaluation baseline to compare against our approach, so we describe SVMs in Section 2.2. Since we propose to use Genetic Programming as the Web analysis algorithm of the hybrid focused crawler, important works related to the use of Genetic Programming in the information retrieval field are reviewed in Section 2.3. Works on Genetic Algorithms are also discussed in Section 2.3 because GA is closely related to GP. Section 2.4 presents research works related to meta-search. Finally, Sections 2.5 and 2.6 briefly describe CITIDEL and NDLTD, likely beneficiaries of our research.

2.1 Focused Crawler

One of the first focused Web crawlers is discussed in [6]. Experiences with a focused crawler implementation were described by Chakrabarti in 1999 [8]. Focused crawlers contain two types of algorithms to keep the crawling scope within the desired domain: (1) Web analysis algorithms are used to judge the relevance and quality of the Web pages pointed to by target URLs; and (2) Web search algorithms determine the optimal order in which the target URLs are visited [76].

2.1.1 Web Analysis Algorithms

During previous studies, many different Web analysis algorithms have been proposed regarding focused crawling and related activities. Generally, they can be classified into

two categories: (1) content-based Web analysis algorithms and (2) link-based Web analysis algorithms. Content-based analysis algorithms analyze the actual HTML content of a Web page to get the relevance information about the page itself. For example, with the help of document indexing techniques, keywords or phrases can be extracted from the body text of a Web page to determine whether the page is relevant to a specified domain. Another common way is to compare Web pages to *Standard Documents* which are already known to be relevant to the target domain, using the *Vector Space Model* [55, 83]. The Vector Space Model has been widely deployed in many existing focused crawlers [3, 19, 54].

Previous related studies have shown that the link structure of the Web represents a considerable amount of latent human annotation. Thus it can provide some important information when analyzing the relevance and quality of Web pages [11]. For example, when there is a direct link from page A to page B, it often means that the author of page A suggests page B because it consists of the same or similar content. In addition, Web pages with more incoming links are often considered to be more important than those with fewer incoming links. The co-citation concept has been employed in link-based analysis algorithms. Web pages are co-cited when they have incoming links from the same set of parent Web pages. Heavily co-cited Web pages are often relevant to the same or similar topics. Co-citation is particularly helpful in finding relevant pages in some domains where pages having similar content avoid pointing to each other (see Section 3.2). Two of the most popular link-based Web analysis algorithms are PageRank [5] and HITS [13].

2.1.2 Web Search Algorithms

The purpose of Web search algorithms is to determine an optimal order in which the target URLs are visited. Like Web analysis algorithms, many different Web search algorithms have been proposed and tested in the focused crawling field. *Breadth-first Search* and *Best-first Search* are two of the most popular search algorithms used in focused crawlers.

Breadth-first search is one of the simplest search algorithms used in Web crawlers. All URLs in the current level will be visited in the order they are discovered before URLs in the next level are visited. The major feature of breadth-first search is that it does not differentiate Web pages of different quality or different topics, which makes it a good choice to build collections for general-purpose search engines. Also, recent studies [69] have shown that breadth-first search could be employed to build domain-specific collections. The proposed assumption is that if the URLs in the current level are relevant to the target domain, it is very likely that Web pages in the next level are also relevant to the target domain. Previous results from [69] have shown that focused crawlers using breadth-first search algorithms could build domain-specific collections with reasonable quality. However, fetching Web pages in a breadth-first order could result in small collections. After a large number of Web pages have been fetched, a focused crawler using a breadth-first search algorithm begins to lose its focus and introduces a lot of noise (non-relevant pages) into the final collection. Some researchers have tried to combine a breadth-first algorithm and a Web analysis algorithm in the focused crawling process [32]. Their proposed approach contains two main steps: Web pages are first downloaded in a breadth-first order, and then non-relevant pages are filtered out from the collection with the help of a Web analysis algorithm. The benefit gained from this combined method is that it can generate much larger collections with much less noise. But this method suffers from low efficiency because it has to process and filter out a lot of non-relevant Web pages during the crawling process.

Best-first search is currently the most popular search algorithm used in focused crawlers [3, 8, 32, 56, 66]. Compared to breadth-first search, a focused crawler adopting best-first search will not simply visit URLs in the order they are discovered. Instead, some heuristics, such as results from Web analysis algorithms, are adopted to rank the URLs in the crawling queue. URLs with high ranks, which are usually considered more promising to link to relevant pages, are visited first. Non-promising URLs are put to the back of the queue, and rarely or never get a chance to be visited. It is quite obvious that best-first search has advantages over breadth-first search because it searches only in directions

where relevant Web pages occur, and avoids downloading non-relevant pages. However, best-first search also has problems. Since best-first search is a *Local Search Algorithm*, it only can traverse the Web space by probing neighbors of the nodes previously visited. Bergmark [4] has shown that focused crawlers using best-first search could miss many relevant pages and build the final collections with low quality.

Some other more advanced search algorithms also were introduced into the focused crawling domain. For example, a parallel search technique called the *Spreading Activation Algorithm* was proposed by Chau and Chen [11] to build domain-specific collections. The algorithm effectively combined content-based and link-based Web analysis algorithms together, which successfully avoids many shortcomings of using either one of them alone, but as the spreading activation algorithm is still a local search algorithm, it shares the limitations of other local search algorithms.

2.2 Support Vector Machines in Text Classification

Support Vector Machines (SVMs) have been extensively evaluated for text classification on reference collections. Often, SVM has been reported the best classifier for text [50]. A content-based SVM classifier was first used in text classification by Joachims [50]. It works over a vector space, where the problem is to find a hyperplane with the maximal margin of separation between two classes. This hyperplane can be uniquely constructed by solving a constrained quadratic optimization problem, by means of quadratic programming techniques. The optimal separating hyperplane will minimize the risk of mis-classifying the training samples and unseen test samples. This idea is illustrated in Figure 2 [101].

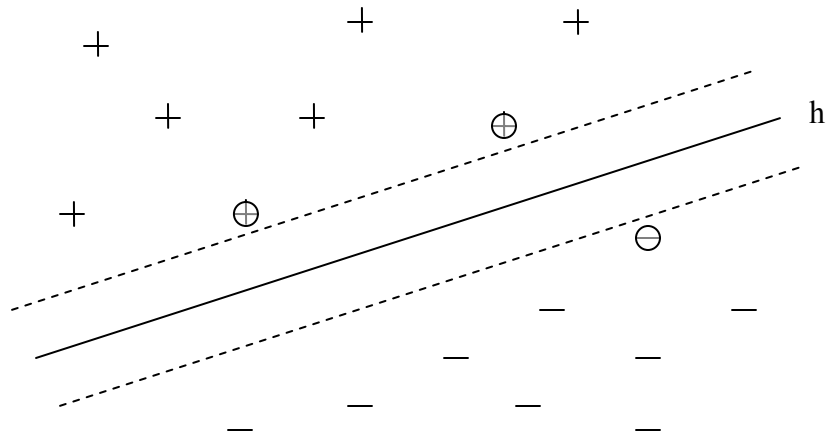


Figure 2: Support Vector Machines (SVMs) find the *hyperplane* h , which separates the *positive* and *negative* training examples with maximum distance. The examples closest to the hyperplane are called Support Vectors (marked with *circles*)

2.3 GA/GP in Information Retrieval

Genetic algorithms have been used to support intelligent information retrieval for quite a long time. A GA-based approach used to index documents was proposed by Gordon [42]. In his research, competing document descriptions (sets of keywords) were associated with a document and changed over time to improve retrieval. Following a similar approach, Gordon [43] also studied the problem of document clustering. His experimental results have shown that, as document descriptions were changed, those relevant to similar queries began to form clusters. A similar research study was carried out by Raghavan and Agarwal [78], who tried to directly build clusters of documents, without trying to adjust the descriptions of individual documents. In Petry [75], a weighted Boolean query was modified by a genetic algorithm in order to improve recall and precision. Yang [97, 98] proposed to adjust query terms' weights based on user feedback. He also reported the effect of adopting genetic algorithms in large scale databases. Chen [14] attempted to learn a user's information need by combining query-

by-selected samples and machine learning techniques. Experimental results indicated that a machine learning system based on a genetic algorithm consistently outperforms other approaches. Chen [13] compared the performance of Web spiders governed by either a genetic algorithm or a best-first search algorithm. When given the same initial set of Web pages provided by a user, a GA-based Web spider could discover more new, relevant documents than the Web spider using a best-first search algorithm. A similar Internet agent/spider system based on a genetic algorithm was developed by Martin-Bautista [65]. Fan et al. [26, 27, 29-31] have successfully applied Genetic Programming algorithms to discover personalized ranking functions by effective combination of different weighting features.

As an extension of Genetic Algorithms, Genetic Programming has been applied to data classification as well. In [12], Cheang proposed a Genetic Parallel Programming classifier to evolve parallel programs for data classification problems. Eggermont [25] proposed several methods using techniques from the machine learning domain to refine and reduce the size of the search space for evolving decision trees for data classification. Experimental results showed how classification performance improves when shrinking the search space in which a tree-based Genetic Programming algorithm searches. Kishore [52] proposed a methodology for GP-based n-class pattern classification. A given n-class problem was modeled as n two-class problems. Furthermore, a GP classifier expression was evolved as a discriminant function for each class. In [53], Kishore integrated the GP classifier with feature space partitioning for localized learning to improve pattern classification. Genetic Programming also has been applied to text classification through the use of a parse-tree. In [16], Clark used GP to route inbound documents to a central classifier which autonomously sent documents to interested research group within a large organization. The central classifier used a parse tree to match the aspects of a document to nodes of the tree, which ultimately leads to a single numerical value—the classification or “confidence value”—during evaluation. Castillo [7] developed a multi-strategy classifier system for document classification. Different types of classifiers (e.g., Naïve Bayes, Decision Trees) were applied to different parts of the document (e.g., titles, references). Genetic algorithms were applied for feature selection as well as for

combining the output of the different classifiers. In her recent studies, Zhang [102-105] proposed a GP-based classification framework to intelligently fuse evidence from multiple sources in order to improve classification of text documents into predefined categories.

2.4 Meta-search

A meta-search engine is a special search engine that sends user queries to several other search engines and/or databases, and returns the results from them [86]. Meta-search engines do not compile a physical database or catalogue of the web. Instead, they take a user's request, pass it to several other heterogeneous databases, and then compile the results in a homogeneous manner based on a specific algorithm. The primary advantages of a meta-search engine over a single search engine are increased coverage and a consistent interface [85]. DogPile [23] submits a user's query to a set of search engines and returns the results in the order determined by the search engines. Glover and Birmingham [36] demonstrated the use of decision theory as a mean of re-ordering results from a single search engine to capture more of a user's information need than a text query alone. Nguyen and Haddawy [71] also showed the use of decision theory as a means of making result ordering decisions. In order to increase the precision of the meta-search results, some meta-search engines such as ProFusion [34], SavvySearch [48], or MetaSEEK [2] do not always send the user's query to the same search engines. ProFusion considers the performance, the predicted subject of the query, and the user's explicit search engine preferences. MetaSEEK considers past results and the user's keywords for source selection, while SavvySearch allows users to specify a "category" to determine the sources to be searched. Glover et al. [37, 38, 63] developed a meta-search tool - Inquirus - at NEC Research Institute, which focuses on making certain user preferences explicit. These preferences define a search strategy that specifies source selection, query modification, and result scoring. One of the best known meta-search engines is MetaCrawler [85, 86]; a snapshot of it is shown in Figure 3.

The CITIDEL project has taken advantage of modern information technologies and digital library research to create and support rich learning environments for a wide variety of end-users. It incorporated the most recent DL technologies and information management research available when building the system. The system is based on components extending ideas of the OAI Protocol [73], allowing easy introduction of new components as needed, and replacement of others [89, 91]. The XML Log Standard for DL, developed at Virginia Tech Digital Library Research Laboratory (DLRL) [21], has been incorporated into the project, which provides a comprehensive record of user access and interaction with the system [40, 41].

One major goal of CITIDEL is to maximize the number of computing-related resources available to computer science scholars and practitioners. With the continuing growth of collections, and as interest in NSDL expands, CITIDEL should attract more and more users, especially undergraduates and high-school students, and have significant impact on the teaching and learning of those interested in computing and information technology. A snapshot of the CITIDEL system is shown in Figure 4.

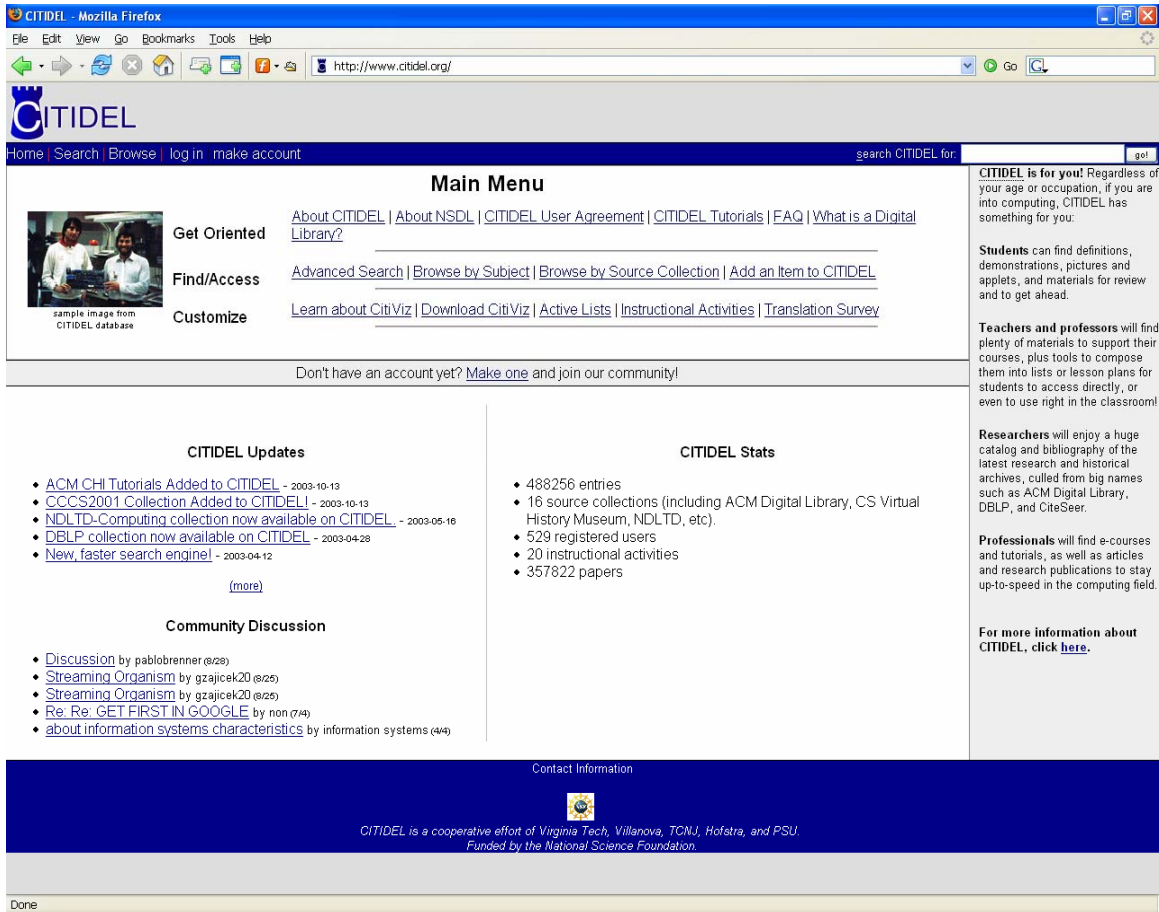


Figure 4: A snapshot of CITIDEL

2.6 NDLTD

The Networked Digital Library of Theses and Dissertations (NDLTD) [70, 87, 88] is a collaborative effort of universities around the world to promote creating, archiving, distributing, and accessing Electronic Theses and Dissertations (ETDs). Since 1996, over two hundred universities have joined the initiative, emphasizing the importance institutions place on training their graduates in the emerging forms of digital publishing and information access. This project has multiple important objectives, including:

1. to improve graduate education by allowing students to produce electronic documents, use digital libraries, and understand issues in publishing;

2. to increase the availability of student research for scholars and to preserve it electronically;
3. to lower the cost of submitting and handling theses and dissertations;
4. to empower students to convey a richer message through the use of multimedia and hypermedia technologies;
5. to empower universities to unlock their information resources; and
6. to advance digital library technology.

The production and archiving of electronic theses and dissertations is fast becoming an accepted part of the normal operation of universities in the new electronic age. NDLTD is dedicated to supporting this trend with tools, standards, and services that empower individual institutions to set up and maintain their own collections of ETDs. At the same time NDLTD promotes the use of these ETDs through institutional websites as well as portal-type websites that aggregate the individual sites and create seamless views of the NDLTD collection. A snapshot of the NDLTD home page is shown in Figure 5.

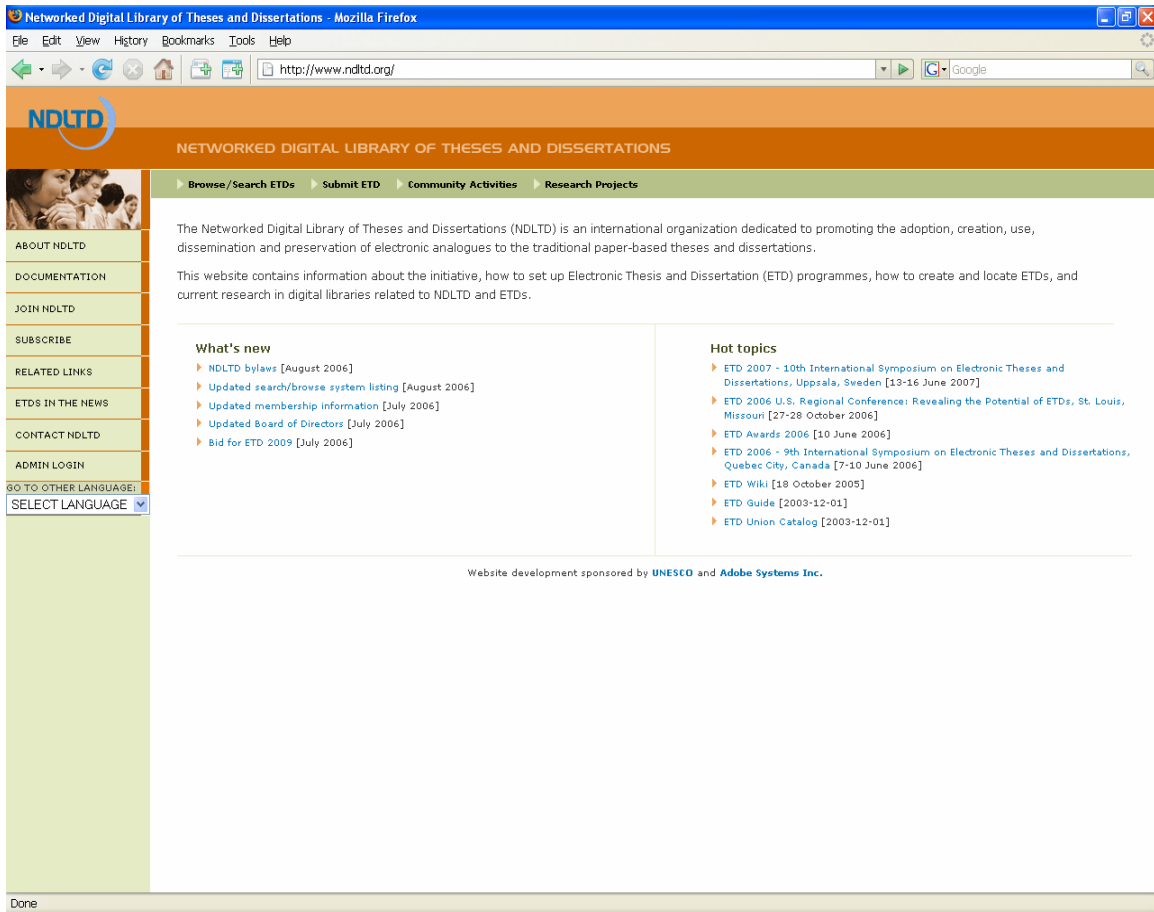


Figure 5: A snapshot of NDLTD

Chapter 3

3 Research Problems

Focused crawlers are used to download topic-specific content from the Internet. They employ two types of algorithms: a Web analysis algorithm and a Web search algorithm. In this chapter, we discuss problems associated with traditional focused crawlers, especially problems with the two algorithms. We also describe some potential solutions, and their advantages and disadvantages.

3.1 Limitations of Current Web Analysis Algorithms

A Web analysis algorithm is a very crucial part of a focused crawler. If the relevant Web pages judged by a Web analysis algorithm are not accurate, they will badly affect the Web search algorithm and result in low-quality Web collections. Implementing a highly effective Web analysis algorithm is the first necessary step leading to a successful focused crawler.

Previous research suggests that content-based Web analysis algorithms perform poorly on the Web [9, 44]. Web documents are usually noisy and with little text, containing images, scripts, and other types of data unusable by simple text classifiers. Furthermore, they can be created by many different authors, with no coherence in style, language, or structure. For this reason, previous research studies with the Vector Space Model, using very simple similarity functions like tf-idf [28] or pivoted tf-idf [28], had limited success on Web page classification.

One possible solution to overcome the limitation of the Vector Space Model is to apply combining evidence from different similarity measures to the content of Web documents. Zhang [102-105] has successfully demonstrated that intelligently fused evidence from

multiple sources can greatly improve classification of text documents into predefined categories.

Global search schemes like the Genetic Algorithm are also promising potential solutions. In Qin [76], researchers proposed to integrate the Genetic Algorithm framework with Web analysis algorithms to build domain-specific collections. Experimental results have shown that the approach mentioned above also could become a potential solution because it is able to build large final collections with less noisy pages, as well as find more relevant Web communities than traditional focused crawlers.

3.2 Shortcomings of Local Search Algorithms

Most existing focused crawlers use local search algorithms to decide the order in which the target URLs are visited, which can result in low-quality domain-specific collections. Most research in the focused crawling field has aimed to improve the performance of different Web analysis algorithms. On the other hand, the problems caused by local search algorithms have been ignored by most researchers.

One property of local search algorithms is that they traverse the Web space by visiting the neighbors of previously visited Web nodes. One major problem with local search algorithms is that a focused crawler will miss a relevant Web page if there is not a chain of links that connects some starting page to that relevant page. Furthermore, the focused crawler will give up crawling in a direction before it reaches the final target pages if there are non-relevant Web pages existing between the start pages and the target pages. Due to this limitation, focused crawlers using local search algorithms are only able to find relevant pages within a limited sub-graph of the Web that surrounds the starting URL set, and miss any relevant Web pages outside this sub-graph. A formal name for this problem is *being trapped with local optima* [76, 77].

The existence of *Web communities* [19, 35, 55, 95], which reinforce the shortcomings of using local search algorithms in focused crawlers, was revealed by recent studies on Web

structure. Researchers found that online Web pages are naturally organized into different groups by specific hyperlink structures. The member pages inside a group are all related to the same or similar topics of interest. Such Web groups are referred to as Web communities. The goal of performing focused crawling is to retrieve all and only those Web pages that belong to relevant Web communities. Unfortunately, three major structural properties of Web communities make local search algorithms not suitable for focused crawlers.

First, researchers found that, instead of directly linking to each other, many Web pages in the same Web community relate to each other through co-citation relationships [19, 95]. This is quite a normal situation existing in the commercial domains where competition is involved. For example, Yahoo news, MSN news, and Google news all provide the same or similar types of news information, but they never contain Web links pointing to each other. Under this circumstance, a focused crawler would probably miss some relevant Web pages even though they belong to the same relevant Web community as the starting URLs.

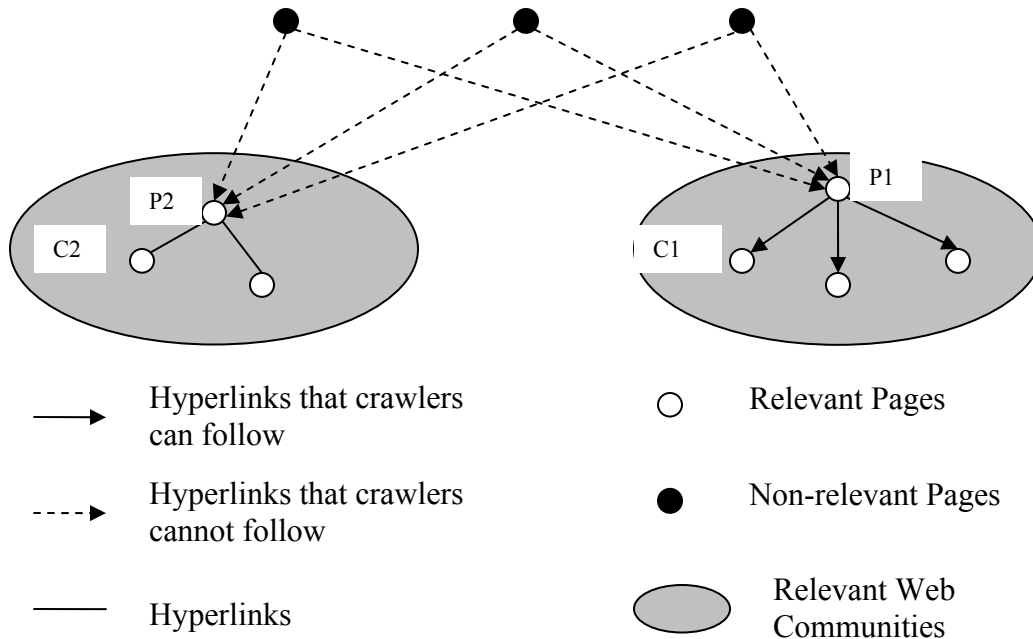


Figure 6: Relevant Web pages with co-citation relationships missed by focused crawlers

Figure 6 illustrates the problem described above. As shown in the figure, C1 and C2 are relevant Web communities, and relevant Web page P2 is related to P1 through co-citation relationships. If a focused crawler starts crawling from page P1, it only can fetch all the relevant pages in Web community C1. The relevant page P2, and all the relevant pages linked to it in Web community C2, will be missed by the focused crawler.

Second, Web pages related to the same domain could be separated into different Web communities by non-relevant Web pages. In [4], Bergmark found that, after studying 500,000 Web pages, most of the pages related to the same domain are separated by at least 1, to up to 12, non-relevant pages, and the average number of non-relevant pages between two relevant ones is 5. In other related research, Kumar [59, 60] reported that, from a large snapshot of the Web, they successfully identified more than 100,000 distinct Web communities. Many of them are relevant to the same or similar topics of interest. Since focused crawlers using local search algorithms will give up downloading when they encounter non-relevant pages in a crawling direction, they will not be able to find

any relevant Web communities outside the initial Web communities containing the starting URLs.

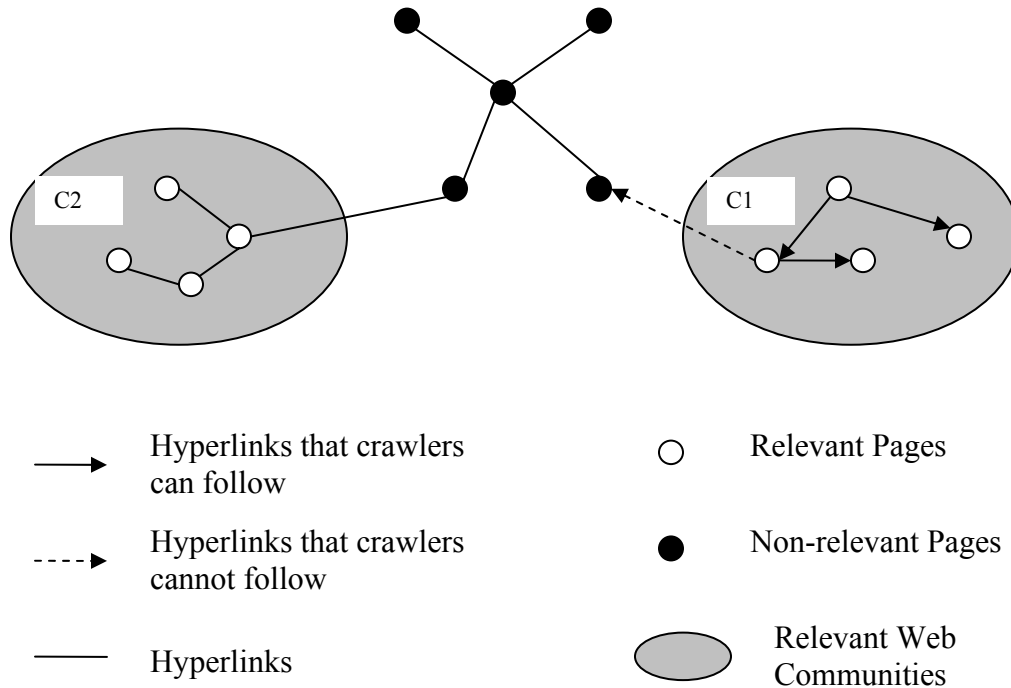


Figure 7: Relevant Web communities separated by non-relevant Web pages

Figure 7 illustrates the problem described above. As shown in the figure, C1 and C2 are two relevant Web communities, which are separated by some non-relevant pages. In this scenario, a focused crawler that starts the crawling process in C1 would miss all the relevant Web pages in C2 because it cannot follow a path with non-relevant Web pages.

Third, researchers found that sometimes some Web links do exist between Web pages which belong to two relevant Web communities, but all of the links have the same direction. They all point from the pages of one community to those of the other, with none of them pointing in the reverse direction [95]. For example, let us consider two Web communities, A and B, which are relevant to a famous pop singer. Community A contains the singer’s official Web pages, and community B contains the singer’s fan club Web pages. Intuitively, Web pages in B will contain URLs linking to pages both in A and

B, but Web pages in A may only contain URLs linking to other pages in A, with no links pointing to pages in B. In this situation, if a focused crawler using a local search algorithm starts to fetch Web pages from community B, the relevant pages in A can still be retrieved. But if the starting URL is in community A, the relevant pages in B will be missed.

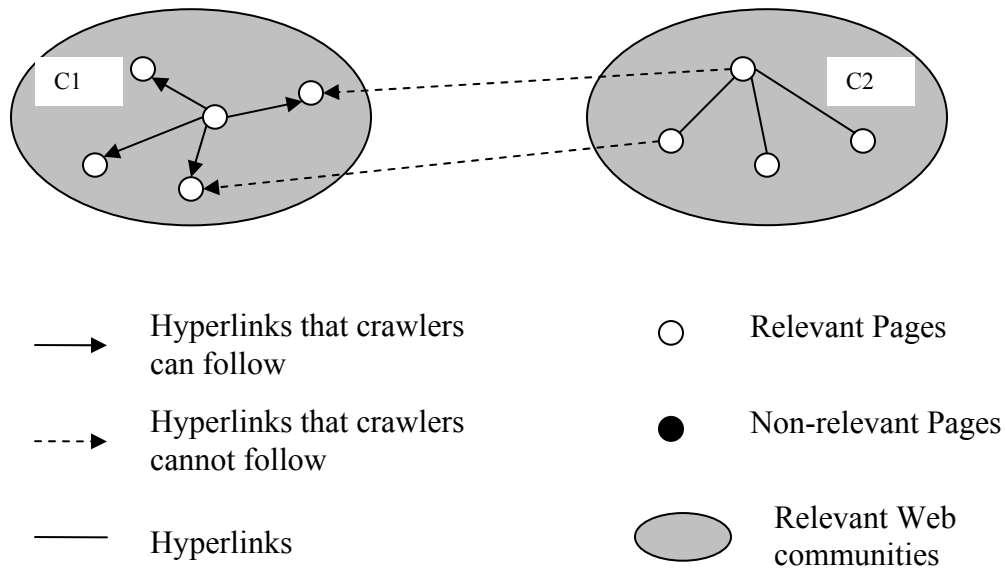


Figure 8: A focused crawler trapped within the initial Web community

Figure 8 illustrates the problem described above. In this figure, C1 and C2 are two relevant Web communities. There are some hyperlinks pointing from C2 to C1, with no hyperlinks in the opposite direction. If a focused crawler starts its searching in C1, it would miss all the relevant Web pages in C2 because even though C2 and C1 are connected, there are no hyperlinks pointing from C1 to C2.

One of the simplest solutions is to provide more starting URLs to the focused crawler. The more starting URLs provided to the focused crawler, the more comprehensive a final collection could be generated. However, it is a very expensive and time-consuming task for domain experts to compile a complete list of high-quality starting URLs. Moreover,

considering the tremendous size and extremely fast growth of the Web, the solution of increasing the number of starting URLs is obviously a dead end.

Another solution is to use the Tunneling technique proposed by Bergmark [4] to address the problems existing in the local search algorithms. Tunneling is a heuristic-based method that solves simple global optimization problems. In the focused crawling field, a focused crawler using Tunneling technology will not give up searching a direction when it encounters a non-relevant Web page. It will continue searching in that direction for a pre-set number of steps. A focused crawler based on Tunneling technology can travel from one relevant Web community to another relevant community if the gap (number of non-relevant pages) between them is within the pre-set limit. Experimental results proved that focused crawlers using Tunneling can find more relevant Web pages than those without Tunneling. However, introducing Tunneling technology into the focused crawling process still cannot solve the problem completely because it doesn't change the local search nature of focused crawling. At the same time, noise (non-relevant pages) may be introduced into the collection, which will downgrade the collection's quality when the focused crawler is forced to visit non-relevant pages.

Some research outside the focused crawling area has provided helpful insights into solving the problems caused by local search algorithms. In their study of the Web size, Lawrence and Giles [64] found that overlap between the search indexes of major search engines is actually very small, so the combined top results from multiple search engines have quite high coverage over the Web. They strongly suggested that any information seeker looking for information about a topic should meta-search multiple search engines and combine top results to fulfill his comprehensive and diverse information need. This meta-search technique has been incorporated and tested in [76, 77], and showed to be a potential solution to the problems caused by local search algorithms.

Chapter 4

4 Approach

In this research, we propose a hybrid focused crawling framework which consists of two main parts: a machine learning algorithm will be used as a Web analysis algorithm, and meta-search will be introduced into the Web search process. Particularly, we investigate an inductive learning method – Genetic Programming – for the discovery of better similarity functions to be used in Web classifiers. We also explore how such combination functions can be used to improve classification performance. The new similarity functions, discovered by our GP system, will be used by *kNN* classifiers [99] to get the final classification results.

4.1 Similarity Measures

Three different content-based similarity measures applied to the content of Web documents will be used: bag-of-words, cosine, and Okapi. These three similarity measures have been widely used in scientific research activities especially in the text classification field [101-105]. In order to compute these similarity measures, the documents are required to be represented as vectors, as in the Vector Space Model [83]. Suppose we have a collection with t distinct index terms t_j . A document d_i can be represented as follows: $d_i = (w_{i1}, w_{i2}, \dots, w_{it})$, where w_{ij} represents the weight assigned to term t_j in document d_i .

For the bag-of-words measure, the similarity between two documents d_1 and d_2 can be calculated as:

$$\text{bag-of-words}(d_1, d_2) = \frac{|\{d_1\} \cap \{d_2\}|}{|d_1|}$$

where $\{d_i\}$ corresponds to the set of terms occurring in document d_i .

For the cosine measure, the similarity between two documents can be calculated as [84]:

$$\text{cosine}(d_1, d_2) = \frac{\sum_{i=1}^t w_{1i} * w_{2i}}{\sqrt{(\sum_{i=1}^t w_{1i}^2) * (\sum_{i=1}^t w_{2i}^2)}}$$

For the Okapi measure, the similarity between two documents can be calculated as:

$$\text{okapi}(d_1, d_2) = \sum_{t \in d_1 \cap d_2} \frac{3 + tf_{d_2}}{0.5 + 1.5 * \frac{\text{len}_{d_2}}{\text{len}_{avg}} + tf_{d_2}} * \log \frac{N - df + 0.5}{df + 0.5} * tf_{d_1}$$

where tf is the term frequency in a document, df is the document frequency of the term in the whole collection, N is the number of documents in the whole collection, len is the length of a document, and len_{avg} is the average length of all documents in the collection.

From these equations, we can find that the cosine similarity measure is symmetric, while the bag-of-words and Okapi similarity measures are not.

4.2 Combining Evidence

Table 1 lists each type of evidence mentioned above. We applied each type of evidence to the title and body fields of Web documents.

Table 1: Types of evidence

Content-based evidence	Ttitle_BOW	Bag of words similarity measure using title as content
	Title_Cos	Cosine similarity measure using title as content
	Title_Okapi	Okapi similarity measure using title as content
	Content_BOW	Bag of words similarity measure using body as content
	Content_Cos	Cosine similarity measure using body as content
	Content_Okapi	Okapi similarity measure using body as content

4.2.1 Definition of GP-based Classification Problem

Each type of evidence shown in the previous section is represented as a document \times document matrix, and serves as the input matrix to the GP-based classification framework. The matrix is defined as:

$$M_k = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

In this matrix, a_{ij} is the similarity value between the two documents d_i and d_j based on one type of similarity measure mentioned in the previous section. GP will try to find a best non-linear function f to combine the matrices M_1, M_2, \dots, M_n , where n is the number of types of evidence. The computational output of the combination through such a non-linear function f is an output matrix defined as M_{GP} :

$$M_{GP} = f(M_1, M_2, \dots, M_n)$$

M_{GP} is a matrix of similarities between pairs of documents. In order to take advantage of information represented in M_{GP} to predict the class label for a document in the classification process, we introduced a method based on a nearest neighbor classifier –

kNN [99]. More detailed information about kNN will be discussed in Section 4.3.4. Macro FI (see Section 5.2) will be used to measure the effectiveness of the GP discovered non-linear function f . When compared with M_k , M_{GP} is denser, more accurate, and can produce better classification results.

4.3 Inductive learning Method - GP

In our proposed approach, the machine learning techniques we have investigated are described below.

4.3.1 Machine Learning: Genetic Programming

Based on the principle of biological inheritance and evolution, Genetic Programming (GP) [58], an extension of Genetic Algorithms (GAs), is a set of artificial intelligence search algorithms which has strong ability to traverse a very large search space efficiently and find approximate global optimal solutions instead of local optimal solutions. Genetic Programming has been widely used and proved to be effective in solving optimization problems, such as financial forecasting, engineering design, data mining, and operations management [58]. GP is capable of solving complex problems for which conventional methods cannot find an answer easily.

The difference between GA and GP is the internal representation (data structure) of the individual. In GA, each individual is commonly (though not always) represented by a fixed-length bit string (like 10010110...) or a fixed-length sequence of real numbers (like 1.2, 3.2, 4.6, 2, ...). In GP, more complex data structures, such as trees (see Figure 9), linked lists, or stacks, are used [61]. Furthermore, the length or size of the data structure is not fixed, although it may be constrained by an actual implementation to within a certain size limit. GA is often used to solve difficult optimization problems, while GP is typically used to approximate complex, nonlinear functional relationships [58].

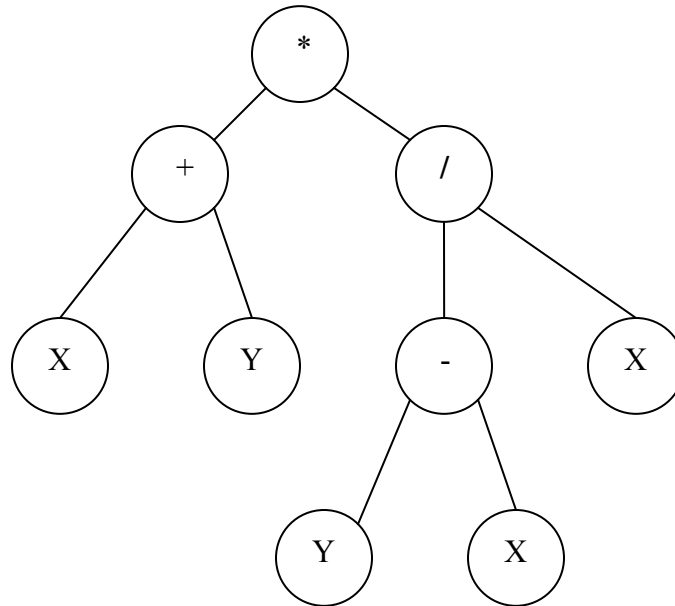


Figure 9: A sample tree representation.

In GP, a population (i.e., a large number of individuals), is maintained at each generation. An individual is a potential solution (formula) for the target problem. All these solutions form a space, say, Σ . An individual can be stored using complex data structures like a tree, a linked list, or a stack. A fitness function ($f(\cdot): \Sigma \rightarrow \mathbb{R}$) also is needed in Genetic Programming. A fitness function takes the solution space, Σ , as its domain, and returns a real number for each individual in the space. Hence, tentative solutions, represented by individuals, can be evaluated and ordered according to their fitness values. The return value of a fitness function must appropriately measure how well an individual, which represents a solution, can solve the target problem.

GP searches for an “optimal” solution by evolving the population, generation after generation. It works by iteratively applying genetic transformations, such as reproduction, crossover, and mutation, to a population of individuals, to create more diverse and better performing individuals in subsequent generations. The reproduction operator directly copies or, using a more appropriate term, clones some individuals into the next generation. The probability for an individual to be selected for reproduction should be proportional to its fitness. Therefore the better a solution solves the problem, the higher

the probability it has to enter the next generation. While reproduction keeps the best individuals in the population, crossover and mutation introduce transformations, and provide variations to enter into the new generation. The crossover operator (see Figure 10) randomly picks two groups of individuals, selects the best (according to the fitness) individual in each of the two groups as parent, exchanges a randomly selected gene fragment of each parent, and produces two “children”. Thus, a “child” may obtain the best fragments of its excellent parents and so may surpass them, providing a better solution to the problem. Since parents are selected from a “competition”, good individuals are more likely to be used to generate offspring. The mutation operator (see Figure 11) randomly changes a gene code of an individual. Using these genetic operators, subsequent generations keep individuals with the best fitness in the last generation, and take in “fresher air”, providing creative solutions to the target problem. Better solutions are obtained either by inheriting and reorganizing old ones or by lucky mutation, simulating Darwinian Evolution.

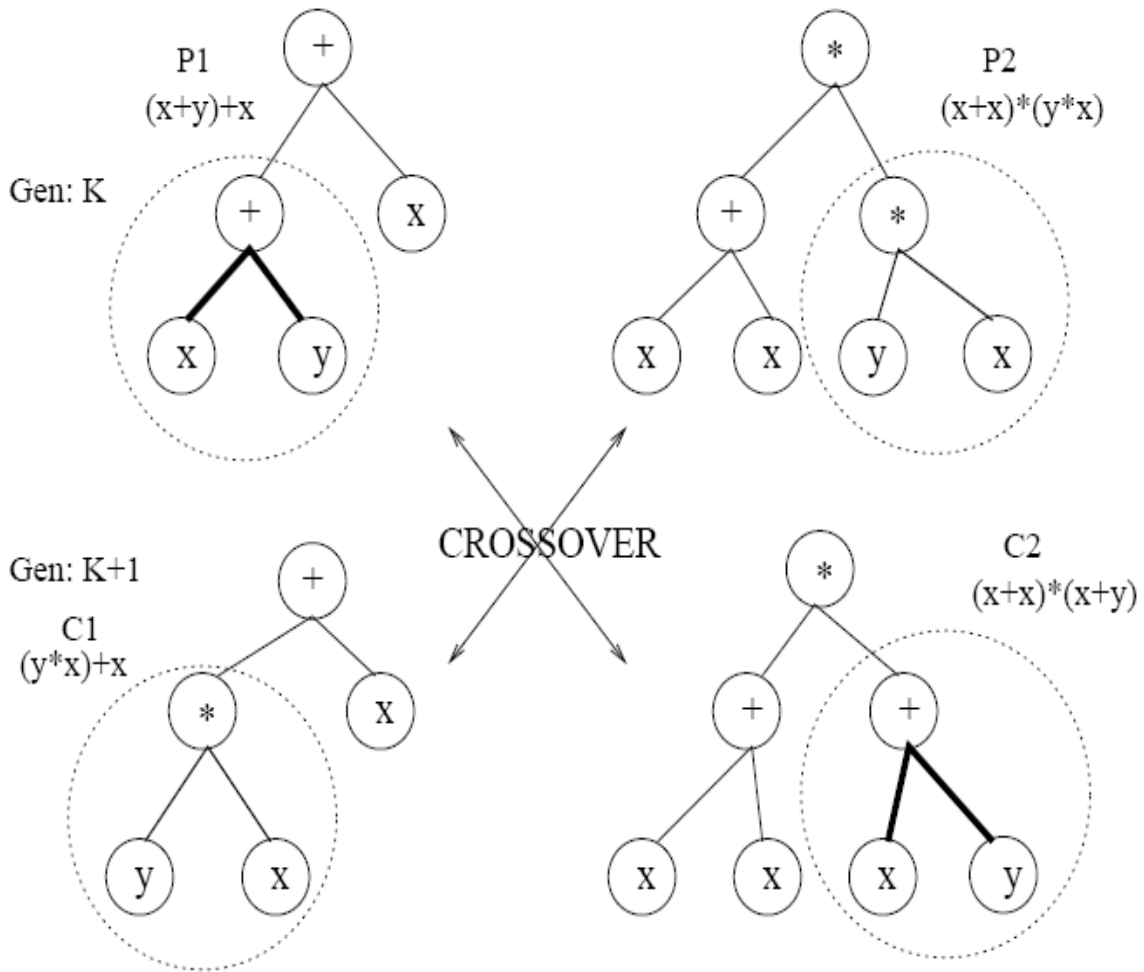


Figure 10: A graphical illustration of crossover operation [105]

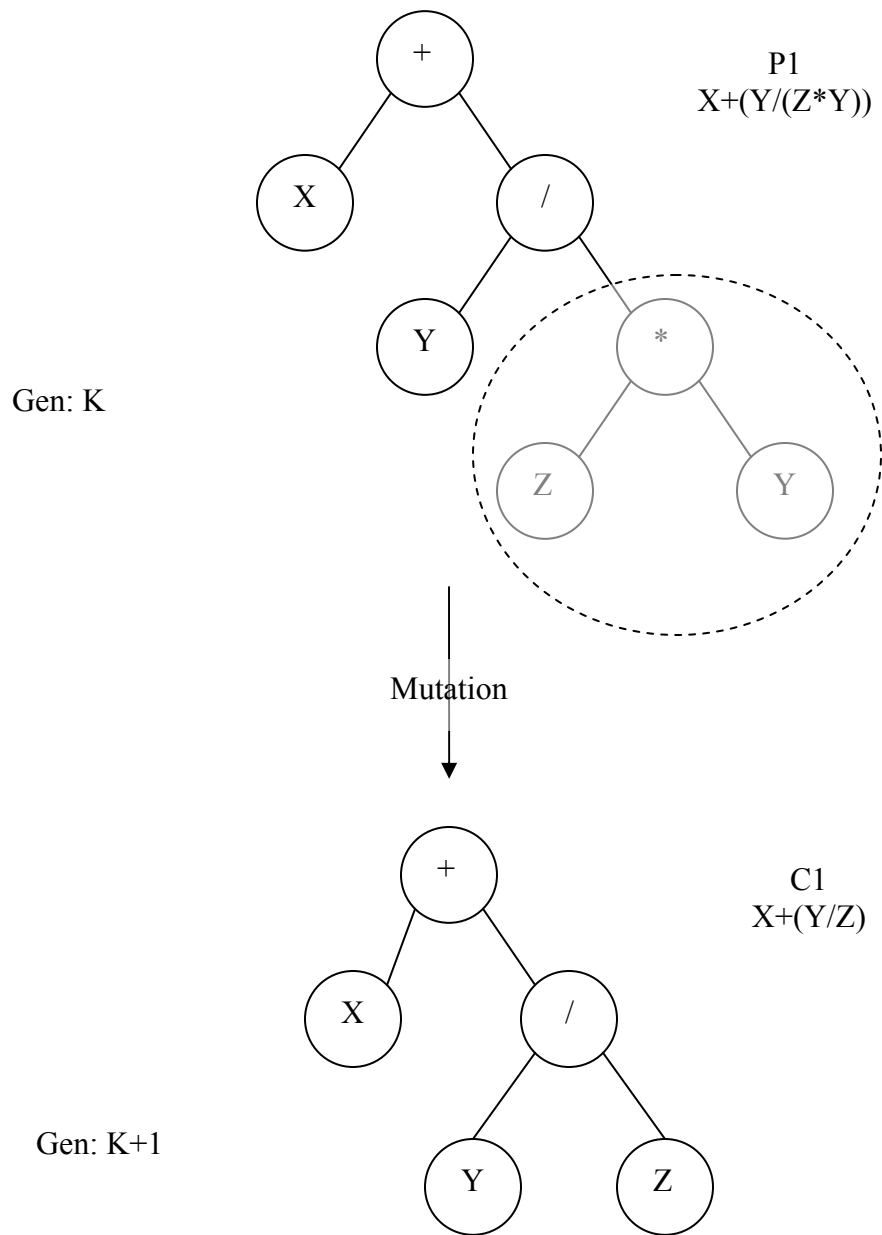


Figure 11: A graphical illustration of mutation operation

In order to apply GP to the Web classification problem, several required key components of a GP system need to be defined. Table 2 lists these essential components along with their descriptions.

Table 2: Essential GP components

Components	Meaning
Terminals	Leaf nodes in the tree structure. i.e., x, y as in Figure 9.
Functions	Non-leaf nodes used to combine the leaf nodes. Commonly, numerical operations: +, -, *, /, log.
Fitness Function	The objective function GP aims to optimize.
Reproduction	A genetic operator that copies the individuals with the best fitness values directly into the population of the next generation without going through the crossover operation.
Crossover	A genetic operator that exchanges sub-trees from two parents to form two new children. Its aim is to improve the diversity as well as the genetic fitness of the population (see Figure 10).
Mutation	A genetic operator that replaces a selected individual's sub-tree whose root is a picked mutation point with a randomly generated sub-tree (see Figure 11).

4.3.2 GP System Configuration

The configurations of the GP system used for similarity function discovery are shown in Table 3.

Table 3: Configurations of GP used for similarity function discovery

Terminals	We use features discussed in Section 4.1 as terminals.
Functions	+, *, /, sqrt
Fitness Function	Macro F1 (discussed in detail in Section 5.2)
Genetic Operators	Reproduction, Crossover, Mutation

4.3.3 Fitness Function

The choice of fitness function can have a huge impact on the final classification results [26]. A good similarity function is defined as a similarity function with a high value. When it is applied to a document y of class C , it ranks documents from class C in such a way that those documents with greater similarities to y are top-ranked. The higher the fitness value, the better the fitness function. The fitness function we select to use is Macro F1. This algorithm uses kNN to predict the category label for a document. Its detailed procedure is shown below:

Let $R = 0, P = 0, T = 0$

for each document y in test collection **do**

 Find the k documents most similar to y

 Predict a category for y according to the kNN algorithm (Section 4.3.4) using the k documents as the k nearest neighbors

if this is a correct prediction **then**

$R = R + 1, P = P + 1$

end if

$T = T + 1$

end for

Let $p = P/T, r = R/|C|$

$$F = \frac{2pr}{p+r} \quad (F \text{ stands for Macro F1})$$

4.3.4 Classification Framework

Along with the configuration settings discussed in the previous sections, the overall classification framework has 4 steps, as described below:

1. For the training collection, generate an initial population of random trees, each tree representing a similarity function.

2. Perform the following sub-steps on training documents for N_{gen} generations.
 - (a) Calculate the fitness value of each similarity tree.
 - (b) Record the top N_{top} similarity trees.
 - (c) Create a new population by: reproduction, crossover, and mutation.
3. Apply the recorded ($N_{gen} * N_{top}$) candidate similarity trees to a set of validation documents, and choose the best performing tree T_{bst} as the unique best discovered similarity tree.
4. Use T_{bst} as the similarity function in a kNN classifier (see below), and apply this classifier to a set of testing documents to get the final category labels.

Steps 1, 2, and 3 concentrate on the training process within GP which aims to discover the best similarity function for the training set, but the discovered function only can be used to calculate the similarities between pairs of documents. In order to evaluate the performance of the best similarity function in the classification task, we adopted a strategy which is based on a nearest neighbor classifier – kNN [99]. The reason to choose the kNN algorithm is its simplicity and its making direct use of similarity information. kNN assigns a category label to a test document, based on the categories attributed to the k most similar documents in the training set.

In the kNN algorithm, to a given test document d is assigned a relevance score $S_{c,d}$ associating d with each candidate category c . This score is defined as:

$$S_{c,d} = \sum_{d' \in N_k(d) \wedge class(d')=c} similarity(d, d')$$

where $N_k(d)$ are the k nearest neighbors (the most similar documents) of d in the training set. In step 4 of our classification framework, the generic similarity function of kNN is replaced by the similarity function discovered by GP.

4.3.5 Properties of Classification Framework

The classification framework suffers from a common scalability problem with GP. It takes a relatively long time for GP to find a good solution during the training process. As a matter of fact, the time complexity of an experiment based on the above framework is $O(N_{gen} * N_{ind} * T_{eval})$, where N_{gen} is the number of generations for evolution, N_{ind} is the number of individuals in a population pool, and T_{eval} is the fitness evaluation time for an individual. Since T_{eval} is determined by the complexity of an individual ($Size$) and the number of training samples ($N_{samples}$), then the total time complexity of an experiment is $O(N_{gen} * N_{ind} * Size * N_{samples})$ [26, 30]. In our research, we mainly focus on the effectiveness of the web classifiers, not the scalability and computing efficiency issue. This problem can be alleviated by improving the speed of the training process through parallel computing as well as through optimization [101]. In addition, in a practical environment, the GP training process only occurs occasionally; the frequent operation will be the actual classification of incoming new Web documents.

4.4 Meta-search

A 1999 study about Web information by Lawrence and Giles [62] estimated the size of the Web at about 800 million indexable pages. This same study also concluded that no single search engine covered more than about sixteen percent of the total. By searching multiple search engines simultaneously through a meta-search technique, the result coverage increases dramatically over searching over one engine. This theory is introduced into our focused crawling framework for the purpose of finding more domain-specific Web documents. Figure 12 illustrates the architecture of the proposed meta-search framework.

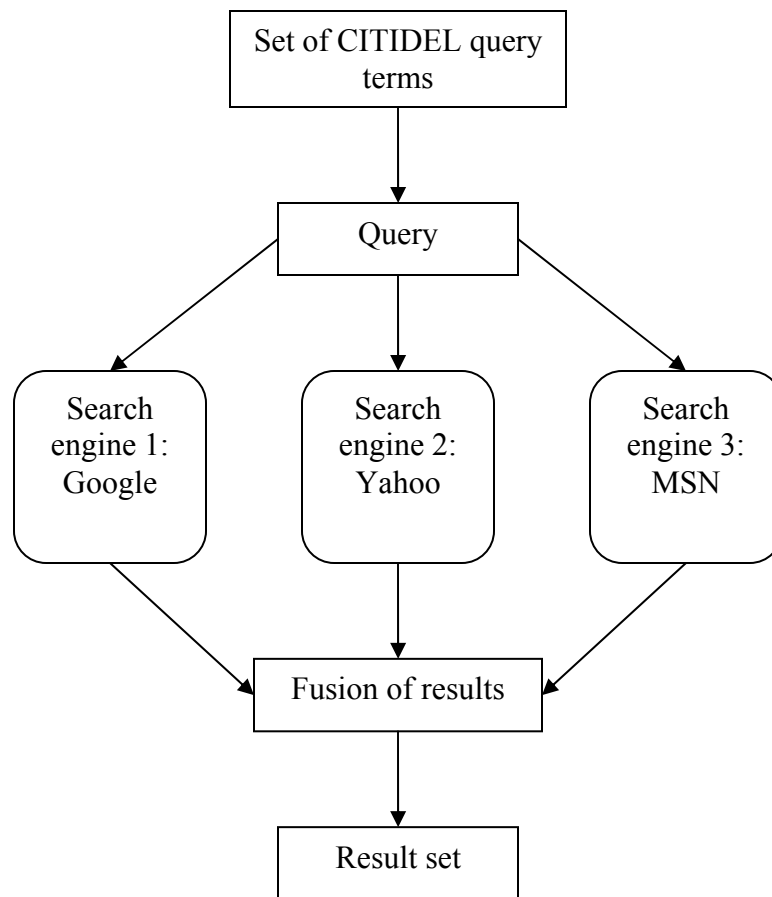


Figure 12: Architecture of proposed meta-search framework

4.4.1 Construction of Query Terms

We use the CITIDEL query logs to construct the query terms used to meta-search multiple search engines. Since CITIDEL only contains computing resources, all queries to the search engine should be relevant to the computing domain. The dates of query log files are from 3/2003 to 12/2003. The total number of queries in the log files is 9080. After removing stop words and duplicate words, we have a set of 3487 different query terms. In order to alleviate the effects of number of query terms over the search results, our meta-search algorithm will randomly pick up a certain number (1-3) of query terms to form a query and use it to meta-search three major search engines (see next section).

4.4.2 Choice of Search Engines

We choose three major search engines as the meta-search target search engines. They are Google (<http://www.google.com>), Yahoo (<http://www.yahoo.com>), and MSN (<http://www.msn.com>). The reason to select these search engines is that they are the top 3 ones which have the largest coverage over the Web [46]. Queries constructed as described in Section 4.4.1 will be sent to these search engines. Top 10 results from each of the search engines will be combined after removing the duplicate Web pages. The result set will then be used by the focused crawling framework (see Section 4.6).

4.5 Rationale of using GP and Meta-search

As reviewed earlier in Section 2.3, both GA and GP have been applied to the information retrieval field [26, 29, 30, 42], as well as for data classification. The decision to choose GP and meta-search for the focused crawling to build high-quality domain-specific collections was motivated by four main factors:

1. The large size of the search space:

With the Web including 11.5 billion pages and growing at a rate of 7 million new pages per day, traditional focused crawlers cannot function well to handle this huge Web space. Genetic Programming has long been known for its strong ability to efficiently traverse very large search spaces to accurately find relevant Web pages.

2. The limitations of local search algorithms:

Given the fact that the overlap between the search indexes of major search engines is actually very small and the combined top results from multiple search engines have very high coverage over the Web [64], meta-search operation can add comprehensive and diverse URLs from many different relevant Web communities [19, 35, 55, 95] during the crawling process.

3. Previous success on the application of GP in the IR field as well as in text classification:

Fan previously reported success in applying GP to discover ranking functions for both individual queries (information routing tasks) [27, 29] and multiple queries (ad-hoc retrieval tasks) [30]. The success of combining several types of evidence through GP to improve text classification [102-105], also motivated us to use GP to accurately classify the incoming Web pages during the focused crawling process.

4. Little prior work on applying GP and meta-search to the focused crawling domain: We propose to fill the near-void that exists in this area. Our research approach appears to have promising broad impact on the focused crawling domain.

4.6 Focused Crawling Framework

A high-level design sketch of the proposed framework adopted for focused crawling process is presented below:

- 1. Discovery of best similarity function.** A data collection with both computing and non-computing documents obtained from DMOZ (see Section 5.1) [22] will be used as the training and validation collections (see Section 5.2). Contents of these Web pages will be analyzed, and similarities based on different measures such as bag-of-Words, cosine [84], and Okapi [82] will be calculated. GP will be used to discover the best similarity function which is a combination of these similarity measures. This newly discovered similarity function can represent the similarity relationship among these Web pages more accurately. The discovered best similarity function will be used in Step 3.
- 2. Initialization.** In this step, the Web pages pointed to by the starting URLs (see Section 5.5) are fetched by the crawler to form the base set.

- 3. Classification.** For each fetched Web page, the GP discovered best similarity function will be used by a kNN classifier to decide if this is a computing-related Web page. If yes, this Web page will survive and be put into the collection. Otherwise, this Web page will be discarded.
- 4. Breadth-first search.** The breadth-first search algorithm will be used to fetch new Web pages. The outgoing links of the surviving relevant Web pages will be collected and put into the crawling queue. The reason to choose breadth-first search is that it is not a local search algorithm (like best-first search), and it does not share the natural limitations of the local search algorithms. Although breadth-first search may increase the crawling time, it is still considered as a good method to solve the problems caused by local search, since crawling time is not a crucial factor when building a domain-specific collection.
- 5. Meta-search.** Meta-searching will be introduced into the crawling process. A random number (selected in the range 1-3) of computing-specific query terms are generated from the query logs of the CITIDEL search engine. These query terms will form a new query to meta-search three major search engines (Google, Yahoo and MSN). Top 10 results from the search engines are combined and put into the crawling queue. The meta-search step will try to obtain diverse relevant URLs globally from the whole search space, and it will not be limited by the boundaries between relevant Web communities because it does not have to follow hyperlinks to find relevant pages.
- 6. Termination.** Steps 3 - 5 are repeated until the number of Web pages in a local collection repository reaches a pre-set limit (see Section 5.6).

Figure 13 summarizes the high-level design of the proposed focused crawling framework.

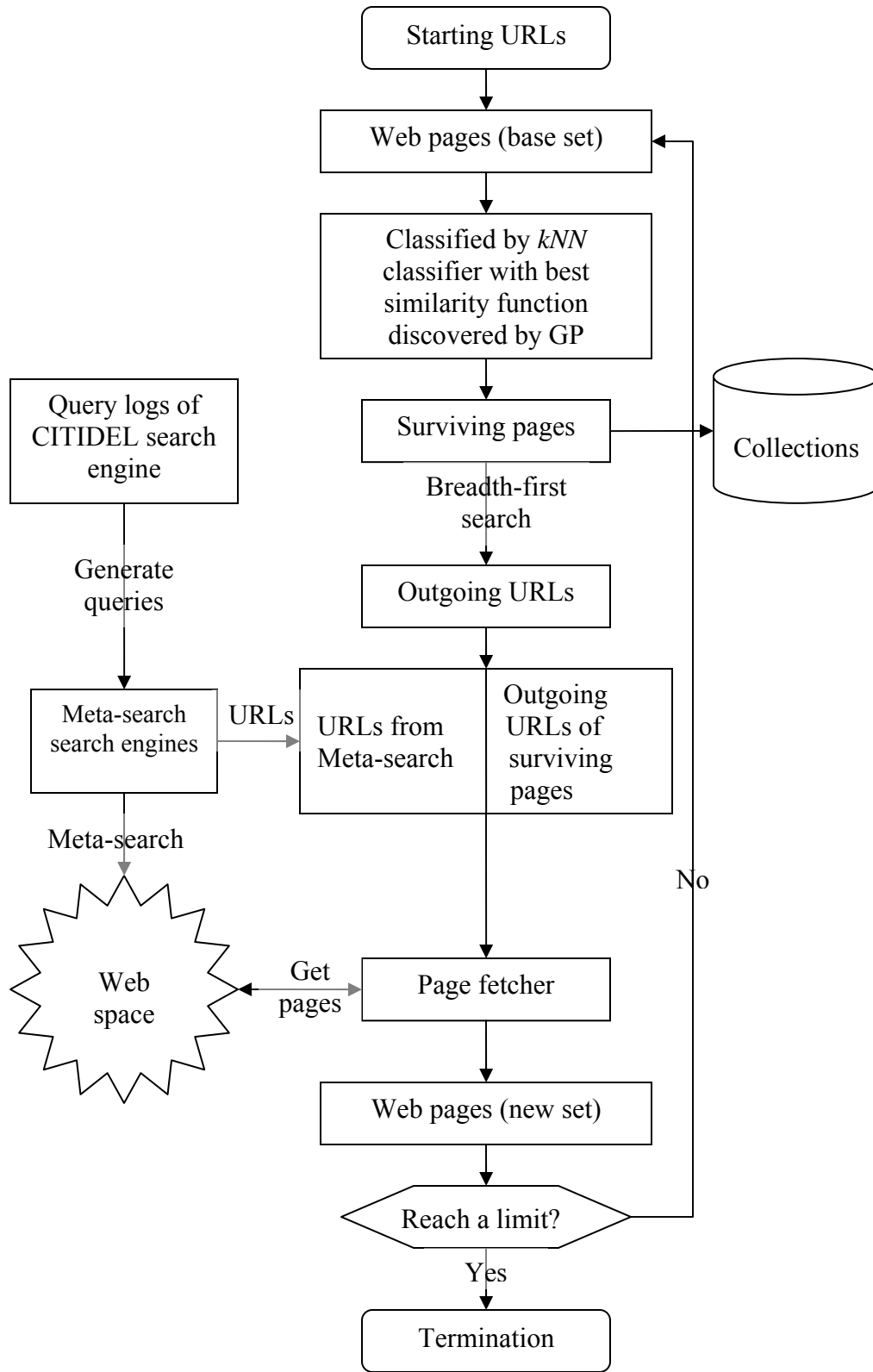


Figure 13: Design of proposed focused crawling framework

Unlike the traditional focused crawlers using the Vector Space Model and local search algorithm, the proposed approach has three design features to ensure traversing the Web search space globally to precisely find more relevant Web documents. The first feature is Genetic Programming's strong ability to efficiently traverse very large search space to accurately find topic-related Web pages. The second feature is the selection of breadth-first search to overcome the problems caused by local search algorithms. The last feature is that the meta-search function could add diverse URLs from many different relevant Web communities due to the fact that the search indexes of different major search engines have little overlap, and their combination covers a very large portion of the Web space.

The proposed hybrid focused crawler has advantages over the previously suggested approaches like using more starting URLs. The meta-search step will introduce more diverse domain-specific URLs along with the crawling process. It is much easier than manually composing a complete list of high-quality starting URLs. This approach will not only make the collection building process easier but also establish a comprehensive and diverse final collection to address the user's information needs more effectively.

The proposed approach also has advantages when compared to the Tunneling technology. Although Tunneling is capable of extending the reach limit of focused crawlers, it doesn't change their local search nature. Moreover, Tunneling introduces noise into the final collection by forcing the focused crawlers to visit non-relevant pages. In contrast, the proposed approach allows the focused crawlers to find new relevant Web communities without any distance limit or any additional noise in the final collection.

One flaw of the proposed focused crawling framework comes from the nature of the breadth-first search. Although breadth-first search is not a local search algorithm, its time and space complexity requirements may make it unsuitable for large Web spaces. Since all tree nodes discovered so far have to be saved, the space complexity of breadth-first search is $O(|V| + |E|)$ where $|V|$ is the number of nodes and $|E|$ the number of edges in the tree graph. Another way of saying this is that it is $O(B^M)$ where B is the maximum

branching factor and M is the maximum path length of the search tree. This immense demand for space is the reason why breadth-first search may be impractical for larger problems. Since in the worst case breadth-first search has to consider all paths to all possible nodes, the time complexity of breadth-first search is $O(|V| + |E|)$ where $|V|$ is the number of nodes and $|E|$ the number of edges in the tree graph [17].

Two alternative search algorithms might be used to replace breadth-first search in the proposed framework. One alternative search algorithm is A^* . A^* (pronounced “A star”) is a graph search algorithm that finds a path from a given initial node to a given goal node. It employs a “heuristic estimate” $h(x)$ that ranks each node x by an estimate of the best route that goes through that node. It visits the nodes in order of this heuristic estimate. The A^* algorithm is therefore an example of best-first search [20]. The time complexity of A^* depends on the heuristic. In the worst case, the number of nodes expanded is exponential in the length of the solution (the shortest path). More problematic than its time complexity is A^* ’s memory usage. In the worst case, it also must remember an exponential number of nodes. Since the A^* algorithm is an example of best-first search, it is actually not a good choice for our proposed framework.

The other alternative search algorithm is the Iterative deepening depth-first search (IDDFS), a state space search strategy in which a depth-limited search is run repeatedly, increasing the depth limit with each iteration until it reaches D , the depth of the shallowest goal state. On each iteration, the IDDFS visits the nodes in the search tree in the same order as depth-first search, but the cumulative order in which nodes are first visited is effectively breadth-first. Iterative deepening depth-first search combines depth-first search’s space-efficiency and both breadth-first search’s completeness (when the branching factor is finite) and optimality (when the path cost is a non-decreasing function of the depth of the node) [17]. The space complexity of IDDFS is $O(B \cdot D)$, where B is the branching factor, and the time complexity of iterative deepening is still $O(B^D)$. In general, iterative deepening may be the preferred search method when there is a large search space and the depth of the solution is not known. More investigations on IDDFS will be carried out as a future research activity.

Chapter 5

5 Experiments

In this chapter, we discuss the data set that is used to test the hypothesis that GP is able to find the best similarity function, which is used by the classifier to categorize incoming Web pages. We also describe the design of the experiments that validate the effectiveness of the proposed GP-based Web analysis algorithm. Finally, we report detailed experimental results in this chapter.

5.1 Test Data Collection

We use the Web collection as our test data collection. DMOZ (<http://www.dmoz.org>), an Open Directory Project as well as a Web directory is chosen as our testing Web document collection. The reason of using this collection as our test data is the generality and wide applicability of the data in industry as well as academia. Experiments with this collection have been used in several research studies, and are published in works such as [74]. Scholars can use the DMOZ data at no cost for their research activities. DMOZ provides its data for download in the Resource Description Framework (RDF) format which is an application of XML. It contains huge data belonging to a wide variety of categories. The categories of Web pages have been judged by human editors of the ODP (Open Directory Project) project. Since one of our research objectives is to enhance the computing resources of CITIDEL, we use the data under the Computers and non-Computers categories for our experiments. The category data we select to build our test data collection contains: *Arts*, *Games*, *Kids_and_Teens*, *Reference*, *Shopping*, *Business*, *Health*, *News*, *Society*, *Computers*, *Home*, *Recreation*, *Science*, and *Sports*. Data under *Regional* and *World* categories are removed because they consist of a large number of non-English Web pages. Table 4 lists the number of documents under each category. The total number of computing documents is 139908, and the total number of non-computing

documents is 1758033, which is the sum of the numbers of documents under all categories except *Computers*.

Table 4: Number of documents under each category of ODP

Category name	Number of documents
Arts	289696
Games	61167
Kids_and_Teens	40261
Reference	65711
Shopping	114142
Business	254564
Health	64728
News	235423
Society	267587
Computers	139908
Home	32535
Recreation	119246
Science	105403
Sports	107570

A common problem with GP is scalability. Each terminal or feature described in Section 4.3.1 represents a similarity matrix which contains the similarity between each pair of documents. Using half or more of the whole collection as our training data set, the required computing resources – in CPU time and amount of memory – would be enormous. Also, the time required to discover a proper classification framework would be significant [101]. In order to reduce the high cost of computing resources and at the same time improve efficiency, a systematic sampling strategy [47] is deployed to evenly select

records and generate the data collection with a moderate size. So, after sampling, the total number of computing records is 13762, and the total number of non-computing records is 12627. The size of the computing category is almost the same as that of the non-computing category.

5.2 Experimental Data Set Design

We adopt a three data-sets design [27, 28, 67] in our experiments. We evenly split the data set into training, validation, and test parts. The purpose of the validation data set is to help alleviate the problem of overfitting of GP on the training data set and select the best similarity function. Overfitting is a common problem occurring in many machine learning and data mining techniques. It happens when the learned or evolved model fits the particulars of the training data overly well and consequently does not generalize to new unseen data [101].

We generate three sets of training samples for the DMOZ Web collection. The first set uses a 10% sample of the collection, the second set uses a 20% sample, and the third set uses a 30% sample. For the 10% training sample, a 7% sample of the collection is used for validation, and the rest of the samples are used for testing. Similarly, for the 20% training sample, a 15% sample of the collection is used for validation, and the rest of the samples are used for testing. For the 30% training sample, a 25% sample of the collection is used for validation, and the rest of the samples are used for testing. All experimental approaches reported in later sections use the same training sets. Results will be compared and evaluated based on the test data sets. In the remainder, we will use 10% to refer to the first sample set, 20% to refer to the second sample set, and 30% to refer to the third sample set, respectively. Table 5 shows the number of documents in these data sets. There has been recent research activities investigating the use of hierarchies for classification [18, 24, 45, 57, 96]. But we only use the content of Web documents for our experiments to verify that GP-based classification works on the Web collection. Hierarchical classification is not the research focus of this work.

Table 5: Number of documents in data sets

		Training	Validation	Test
10%	Computing	1377	983	11402
	Non-computing	1263	972	10392
20%	Computing	2753	1966	9043
	Non-computing	2526	1804	8297
30%	Computing	4200	3400	6162
	Non-computing	3800	3100	5727

5.3 Baselines

In order to test the hypothesis that GP is able to provide better classification results, we need to compare it with the classification statistics of other classification frameworks (baselines). We use the commonly used *F1* as the comparison standard, which is a combination of precision and recall; the *F1* measure was first introduced by van Rijsbergen [81]. Precision p is defined as the proportion of correctly categorized documents to all the documents assigned to the target category. Recall r is defined as the proportion of correctly categorized documents to all the documents having the target

category. $F1 = \frac{2pr}{p+r}$. The reason we choose *F1* is that *F1* is a balanced combination of

precision and recall. It avoids the errors that you can get perfect precision by always assigning zero categories, or perfect recall by always assigning every category, maximizing precision and recall at the same time, thereby maximizing *F1*. Macro *F1* (*F1* results are computed on a per-category basis, then averaged over all categories) was adopted to obtain a single performance value over all categories because it gives equal weight to each category [100].

5.4 Experimental Set Up

Normally, a larger population size and different rate for the genetic transformation operations like crossover, reproduction, and mutation produce better classification results. On the other hand, they have a huge effect on the training time. After exploration in some preliminary experiments [101], we use the settings shown in Table 6 as our GP system experimental settings. We only report performance of the best tree in the validation sets applied to the test sets.

Table 6: GP system experimental settings

Population size	400
Crossover rate	0.65
Mutation rate	0.05
Reproduction rate	0.30
Generations	30

5.5 Starting URL Set

Compiling a high-quality starting URL set is the first key factor leading to a successful focused crawler. In order to test the effectiveness of the proposed focused crawling framework, we carefully generate a starting URL set for the focused crawler. It contains URLs pointing to the homepages of thirty US Computer Science departments. All of them are relevant to the computing domain, and they also belong to different Web communities. The content of the starting URL set is shown below:

Table 7: Content of starting URL set

http://www.cs.berkeley.edu	http://www.cs.ucla.edu
http://www.cs.brown.edu	http://www.cs.uga.edu
http://www.cs.bu.edu	http://www.cs.umass.edu
http://www.cs.cmu.edu	http://www.cs.umd.edu
http://www.cs.colostate.edu	http://www.cs.unc.edu
http://www.cs.duke.edu	http://www.cs.uwp.edu
http://www.cs.haverford.edu	http://www.cs.virginia.edu
http://www.cs.lsu.edu	http://www.cs.vt.edu
http://www.cs.montana.edu	http://www.cs.wisc.edu
http://www.cs.northwestern.edu	http://www.cs.wmich.edu
http://www.cs.pitt.edu	http://www.csc.ncsu.edu
http://www.cs.rice.edu	http://www.cse.sc.edu
http://www.cs.rochester.edu	http://www.eecs.umich.edu
http://www.cs.rutgers.edu	http://www.mcs.kent.edu
http://www.cs.stanford.edu	

5.6 Experimental Results

We perform experiments on the test bed collection described in previous sections. Experimental results are reported in this section. The same training sets are used for different approaches under comparison.

We demonstrate the effectiveness of our proposed classification framework by comparing our experimental results with the results achieved through a content-based SVM classifier and a combination-based SVM classifier. The SVM classifier has been extensively evaluated for classification tasks on data collections, thus offering a strong baseline for the performance comparison. The difference between content-based SVM and combination-based SVM lies in the kernel combination. Joachims et al. [51] have

indicated how to combine different similarity measures by means of composite kernels in their research work. Their approach contains a combination of simple and well-understood kernels by a series of “kernel preserving” operations, to construct an increasingly matching feature space S . In order to apply their method to our application domain, we started by finding that each one of our types of evidence can be represented as positive document \times document matrices, as described in Section 4.2. Therefore, we can represent each type of evidence as a kernel matrix $K_{evidence}$ with elements $K_{evidence}(d_i, d_j)$, where d_i and d_j are document vectors. The kernel matrix for our final feature space S is obtained by means of a linear combination of our initial kernel matrices, as

$$K_{combined}(d_i, d_j) = \frac{1}{N} \sum_{\forall evidence} K_{evidence}(d_i, d_j)$$

where N is the number of distinct kinds of evidence used. Finally, the classification decisions for the combined kinds of evidence are obtained by applying a linear SVM classifier in our final feature space S .

The SVM classifier we use is LIBSVM - A Library for Support Vector Machines [10]. LIBSVM is an integrated software package for support vector classification (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR), and distribution estimation (one-class SVM). It supports multi-class classification. The reasons we choose it are that it is easy to use and has powerful computing abilities. Main features of LIBSVM include:

- Different SVM formulations
- Efficient multi-class classification
- Cross validation for model selection
- Probability estimates
- Weighted SVM for unbalanced data
- Automatic model selection which can generate a contour of cross validation accuracy

- Both C++ and Java sources

The SVM type we select to use is C-SVM and the kernel function type is RBF (Radial Basis Function). The results reported in this section are based on the Macro F1 fitness function. Tables 7, 8, and 9 demonstrate macro F1 comparison between GP, content-based SVM, and combination-based SVM on the 10%, 20%, and 30% samples.

Table 8: Macro F1 comparison between GP, content-based SVM, and combination-based SVM on the 10% sample

Class	GP	Content-based SVM	Combination-based SVM
Computing	80.61	76.34	76.05
Non-computing	66.94	50.50	49.98
Avg. Macro F1	73.78	63.42	63.02

Table 9: Macro F1 comparison between GP, content-based SVM, and combination-based SVM on the 20% sample

Class	GP	Content-based SVM	Combination-based SVM
Computing	81.35	78.29	76.67
Non-computing	68.25	52.22	50.22
Avg. Macro F1	74.80	65.25	63.45

Table 10: Macro F1 comparison between GP, content-based SVM, and combination-based SVM on the 30% sample

Class	GP	Content-based SVM	Combination-based SVM
Computing	82.37	78.49	77.55
Non-computing	69.24	53.60	51.12
Avg. Macro F1	75.80	66.05	64.34

With average macro F1 as the comparison criteria, when comparing GP against the content-based SVM classifier, it is clear that GP achieves better performance: When comparing with the content-based SVM, GP obtains a gain of 16.34% in the 10% sample, 14.64% in the 20% sample, and 14.76% in the 30% sample. When GP is compared with the combination-based SVM classifier, the gains are even higher. We have a gain of 17.07% in the 10% sample, 17.89% in the 20% sample, and 17.81% in the 30% sample, respectively. Table 10 displays the macro F1 gains of GP over content-based SVM and combination-based SVM.

Table 11: Macro F1 gains of GP over content-based SVM and combination-based SVM

	10% sample	20% sample	30% sample
Content-based SVM	16.34%	14.64%	14.76%
Combination-based SVM	17.07%	17.89%	17.81%

One purpose of this research work is to compare GP with other approaches to determine whether GP produces measurably better results than the others. The most common solution to this problems is to apply the pair-wise t-test to these differences [49]. The t-test compares the magnitude of the difference between the two approaches under comparison to the variation among the differences. If the average difference is large compared to its standard error, then the approaches are significantly different. According to [49], the t-test is a valid and suitable test measure for our experimental results. We did a pair-wise t-test comparing GP with content-based SVM and combination-based SVM in Tables 7, 8, and 9, respectively. GP is statistically significantly different from both of the SVM classifiers, with $p < 0.05$. Consequently we can conclude that on the average, GP performs better than other approaches.

In order to test our proposed framework, we also implement three focused crawlers. The first one uses a *kNN* classifier as the Web classifier. The second one uses the content-based SVM classifier as the Web classifier. The third one uses the combination-based SVM classifier as the Web classifier. Each of them uses the same starting URL set (see Section 5.5), and is pre-set to download the same number of Web pages (500). The computing relevance of the downloaded Web pages is judged by human experts. The number of Web pages to be downloaded is set to be 500 so the human experts can finish the categorization task within a reasonable time period. All of the downloaded Web pages are classified based on the 10%, 20%, and 30% samples.

Table 12: Macro F1 comparison between GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 10% sample

Class	GP	Content-based SVM	Combination-based SVM
Computing	80.60	77.83	77.27
Non-computing	64.62	52.08	50.78
Avg. Macro F1	72.61	64.95	64.03

Table 13: Macro F1 comparison between GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 20% sample

Class	GP	Content-based SVM	Combination-based SVM
Computing	81.20	77.98	77.01
Non-computing	64.62	52.65	50.77
Avg. Macro F1	72.91	65.31	63.89

Table 14: Macro F1 comparison between GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 30% sample

Class	GP	Content-based SVM	Combination-based SVM
Computing	82.71	78.62	77.97
Non-computing	66.67	53.94	52.33
Avg. Macro F1	74.69	66.28	65.15

From Tables 11, 12, and 13, we can see that GP still outperforms the content-based SVM classifier at a gain of 11.79% in the 10% sample, 11.64% in the 20% sample, and 12.69% in the 30% sample. GP also produces better performance than the combination-based SVM classifier at a gain of 13.40% in the 10% sample, 14.12% in the 20% sample, and 14.64% in the 30% sample. Table 14 shows the macro F1 gains of GP over content-based SVM and combination-based SVM. Again, we did a pair-wise t-test comparing GP with content-based SVM and combination-based SVM in Tables 11, 12, and 13, respectively in order to strengthen the conclusion that GP produces better results than the others. GP is statistically significantly different from both SVM classifiers, with $p < 0.05$.

Table 15: Macro F1 gains of GP over content-based SVM and combination-based SVM on 500 crawled Web pages

	10% sample	20% sample	30% sample
Content-based SVM	11.79%	11.64%	12.69%
Combination-based SVM	13.40%	14.12%	14.64%

In order to prove the effectiveness of our hybrid focused crawling framework, we implement the fourth focused crawler which uses a *kNN* classifier as the Web classifier, and meta-search algorithm as described in Section 4.4. This hybrid focused crawler uses the same starting URL set (see Section 5.5) as three other focused crawlers, and also is pre-set to download the same number of Web pages (500). The computing relevance of the downloaded Web pages is judged by human experts. All of the downloaded Web pages are classified based on the 10%, 20%, and 30% samples.

Table 16: Macro F1 comparison between GP + meta-search, GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 10% sample

Class	GP + meta-search	GP	Content-based SVM	Combination-based SVM
Computing	85.66	80.60	77.83	77.27
Non-computing	66.58	64.62	52.08	50.78
Avg. Macro F1	76.12	72.61	64.95	64.03

Table 17: Macro F1 comparison between GP + meta-search, GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 20% sample

Class	GP + meta-search	GP	Content-based SVM	Combination-based SVM
Computing	87.02	81.20	77.98	77.01
Non-computing	65.88	64.62	52.65	50.77
Avg. Macro F1	76.45	72.91	65.31	63.89

Table 18: Macro F1 comparison between GP + meta-search, GP, content-based SVM, and combination-based SVM of 500 crawled Web pages on the 30% sample

Class	GP + meta-search	GP	Content-based SVM	Combination-based SVM
Computing	87.34	82.71	78.62	77.97
Non-computing	68.42	66.67	53.94	52.33
Avg. Macro F1	77.88	74.69	66.28	65.15

From Table 15, 16, and 17, we conclude that GP with meta-search produces better performance than GP at a gain of 4.83% in the 10% sample, 4.86% in the 20% sample, and 4.27% in the 30% sample. When comparing with content-based SVM and combination-based SVM, GP with meta-search outperforms both of them at big gains of 17.20% and 18.88% in the 10% sample, 17.06% and 19.66% in the 20% sample, 17.50% and 19.54% in the 30% sample, respectively. Table 18 shows the macro F1 gains of GP with meta-search over GP, content-based SVM, and combination-based SVM. Results of pair-wise t-tests show that GP with meta-search is statistically significantly different from the others, with $p < 0.05$.

Table 19: Macro F1 gains of GP + meta-search over GP, content-based SVM, and combination-based SVM on 500 crawled Web pages

	10% sample	20% sample	30% sample
GP	4.83%	4.86%	4.27%
Content-based SVM	17.20%	17.06%	17.50%
Combination-based SVM	18.88%	19.66%	19.54%

There are two design flaws associated with the above experiments. The first one is that the category labels of 500 downloaded Web pages were judged by the author himself, which may introduce evaluation bias into the experimental results. This problem can be solved by having third-party experts to classify the downloaded Web pages to erase the evaluation bias. The second one is that the testing focused crawlers are set to download 500 pages only due to the time limitation of our research. More experiments with large numbers of downloaded pages should be carried out to reinforce the effectiveness of the proposed crawling framework. Improvement of the experimental design will be achieved as part of future work.

5.7 Computation Time and Space Analysis

As discussed earlier in Section 4.3.5, the classification framework suffers from the scalability problem. Table 19 shows the space requirement and computation time for discovering the best similarity functions for the three sample data sets. Figure 14 shows a graph of sample size vs. training time, and Figure 15 shows a graph of sample size vs. training matrices size. Given a large data collection, the storage space requirement is very huge in order to store the similarity values between each pair of documents. But this problem can be alleviated through the approach of sparse representation of the similarity matrices and/or fast price drop of hard drives. We consider the computation time shown in Table 19 to be acceptable to our experiments since our research mainly focuses on the effectiveness of the framework. The scalability issue can be addressed through matrix optimization [101] and/or parallel processing [1, 33].

Table 20: Computation time and space requirement

Machine	Sample	Sample size	Training time (Hours)	Training matrices size (MBytes)
CPU: 2 × Intel Xeon 3.60 GHz Memory: 4GB Hard Drive: 138GB	10%	2640	5.64	385
	20%	5279	17.12	1382
	30%	8000	45.24	2147

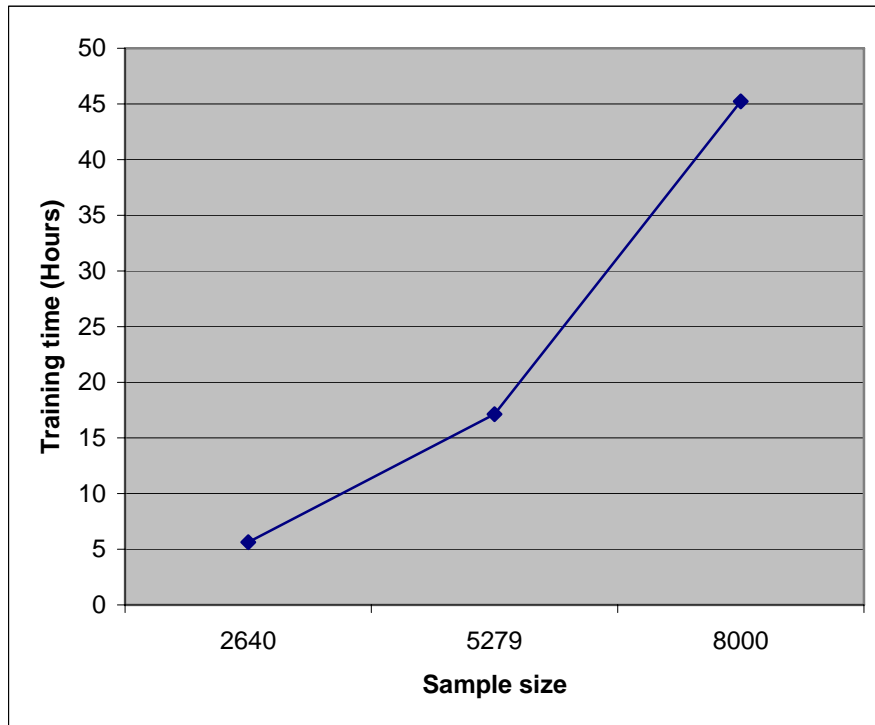


Figure 14: Sample size vs. training time

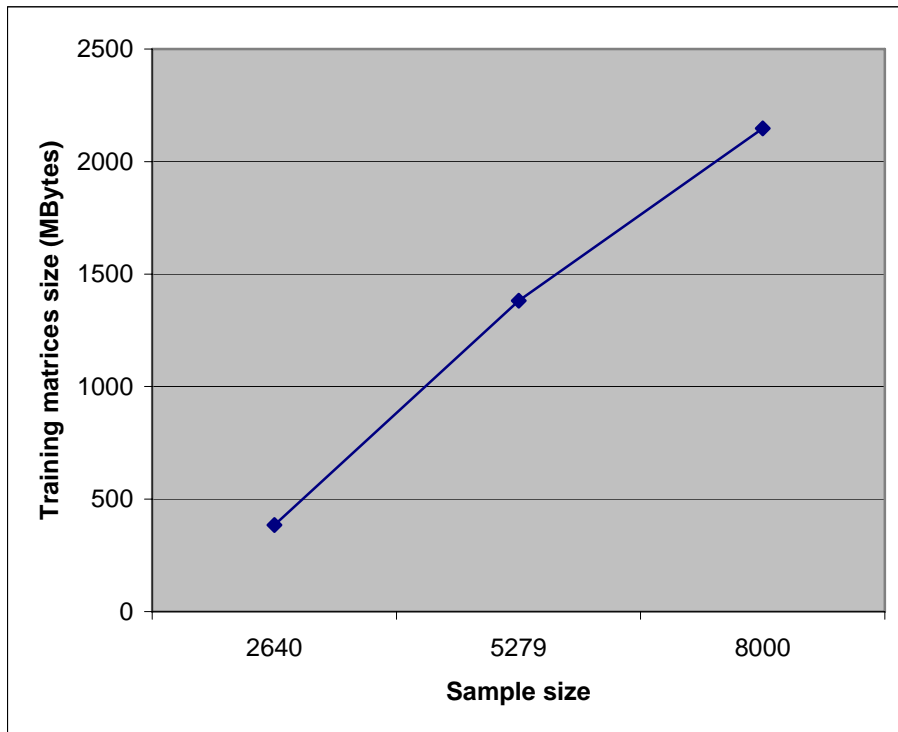


Figure 15: Sample size vs. training matrices size

Chapter 6

6 Conclusions and Future Work

In this chapter, we summarize the achievements of this research work, and discuss some possible future research directions.

6.1 Conclusions

In this work, we considered the problems of traditional focused crawlers which use the simple Vector Space Model and local search algorithm. A framework for addressing such problems, based on Genetic Programming and meta-search, has been proposed and tested. Through the use of Genetic Programming, different sources of evidence based on the content of Web documents were combined, and the effects of such combination were investigated. Through the use of meta-search, query terms generated from CITIDEL query logs were sent to three major search engines, and the effects of such a process were explored.

We studied the effectiveness of the proposed framework with the testbed collection from DMOZ. Our experiments on this collection show that improvement can be achieved relative to other machine learning approaches if Genetic Programming methods are combined with a kNN classifier.

Our experimental results have demonstrated that the GP framework can be used to discover better similarity functions, which can be used by a kNN classifier to achieve better results than both traditional content-based and combination-based SVM classifiers. Our experiments also showed that the meta-search technique can improve the efficiency of Web search by combining the results of multiple search engines. Experimental results coming from the focused crawler with both GP and meta-search showed the effectiveness of this hybrid framework.

6.2 Future Work

In this section, we give a list of suggestions for the future work of this dissertation. The list discusses open questions left by the research, and new ideas found during the course of the work.

Analysis of Experimental Results

There are two interesting phenomena shown in the experimental results. One is that the meta-search algorithm only improves Macro F1 performance by 4-5%. The other is that the *kNN* classifier always achieves much better results on computing related Web documents than non-computing related documents. More extensive analysis is needed to explain these results. Special attention should be given to work to reveal the internal structural characteristics of Web collections.

Parallel Computing

The effectiveness of the GP classification framework comes with a flaw of high computational cost. It requires a long training time when calculating best fitness functions. It is quite possible to solve this problem by parallelization [1]. With the help from the Virginia Tech supercomputer System X [92], which can theoretically handle 17 trillion operations per second, the scalability problem could be greatly alleviated. It can promote the proposed framework into practical application.

Multi-class Classification

In this dissertation, the Web documents used for all the experiments belong to one category only. In the real world, it is very common that Web documents fall into more than one category. This is known as multi-class classification. Multi-class classification is a very important problem in the machine learning area because applications that require multi-class categorization constantly exist. Our classification framework can be extended to remove the constraint that the category of a document is unique.

Hierarchical Classification

For the test data collection, our experiments treat each category (class) separately, which discards the hierarchical structure information existing in the collection. The hierarchical structure can be used to decompose the classification problem into a set of smaller problems corresponding to hierarchical splits in the category tree. The idea is that after learning to distinguish among classes at the top level, lower level distinctions are learned only within the appropriate top level of the category tree. Each of the sub-problems can be solved much more efficiently, and hopefully more accurately, as well [24].

Crawling Update Problem

The crawling update problem contains two aspects: one is the update of the classification framework, the other is the update of search engine databases. With the fast growing speed of online Web documents, it is very possible that documents of a new category appear after some time. So it is a common problem to update the classification model periodically due to the fact that running training tasks takes quite a large amount of time. Also, centralized search engines databases are always out of date. There is a time lag between the time when new information is available online and the time that it is indexed. This problem will introduce out-of-date search results from the meta-search process. Making the update tasks more time efficient will be a tough issue and also a key factor to the successful application of the crawling framework into practice.

Increase Meta-search Coverage

We choose three major search engines in our meta-search step in order to increase the limited coverage of single search engine. But the approach we adopted can't guarantee a full coverage of the Web. One way to solve this issue is to increase the number of search engines used in the meta-search step to get as much Web coverage as possible. The choice of search engines should include general search engines as well as domain-specific ones.

Support Multiple File Formats

The proposed focused crawling framework can only handle HTML Web pages. Much online information exists in the formats of image, PDF, or PS files, etc. Our proposed focused crawling framework can be extended to be capable of handling multiple file formats [93, 94] so that it will find more online Web information and enrich the final data collections.

Bibliography

1. Andre, D. and Koza, J.R., *A parallel implementation of genetic programming that achieves super-linear performance.* in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications.* p. 1163-1174. 1996. Sunnyvale, CA, USA.
2. Benitez, A.B., Beigi, M., and Chang, S.-F., *Using Relevance Feedback in Content-Based Image Metasearch.* IEEE Internet Computing, 1998. 2(4): p. 59-69.
3. Bergmark, D., *Collection Synthesis.* in *Proceedings of the 2nd ACM/IEEE-CS joint conference on digital libraries* p. 253-262. 2002. Portland, Oregon, USA.
4. Bergmark, D., Lagoze, C., and Sbityakov, A., *Focused Crawls, Tunneling, and Digital Libraries.* in *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries.* p. 91-106. 2002. Rome, Italy.
5. Boley, D., Gini, M., Gross, R., Han, S., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., and Moore, J., *Partitioning-Based Clustering for Web Document Categorization.* Decision Support Systems, 1999. 27(3): p. 329-341.
6. Bra, P.D., Houben, G., Kornatzky, Y., and Post, R., *Information Retrieval in Distributed Hypertexts.* in *Proceedings of the 4th RIAO Conference.* p. 481-491. 1994. New York.
7. Castillo, M.D.D. and Serrano, J.I., *A multistrategy approach for digital text categorization from imbalanced documents.* SIGKDD, 2004. 6(1): p. 70-79.
8. Chakrabarti, S., Berg, M.V.D., and Dom, B., *Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery.* in *Proceedings of the 8th International WWW Conference.* p. 545-562. 1999. Toronto, Canada.
9. Chakrabarti, S., Dom, B., and Indyk, P., *Enhanced hypertext categorization using hyperlinks.* in *Proceedings of the ACM SIGMOD International Conference on Management of Data.* p. 307-318. 1998. Seattle, Washington.
10. Chang, C.-C. and Lin, C.-J., *LIBSVM: a library for support vector machines, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.* in 2001.
11. Chau, M. and Chen, H., *Comparison of Three Vertical Search Spiders.* IEEE Computer, 2003. 36(5): p. 56-62.

12. Cheang, S.M., Lee, K.H., and Leung, K.S., *Data Classification Using Genetic Parallel Programming*. in *Genetic and Evolutionary Computation - GECCO-03*. p. 1918-1919. 2003. Chicago, USA.
13. Chen, H., Chung, Y., Ramsey, M., and Yang, C., *A Smart Itsy-Bitsy Spider for the Web*. *Journal of the American Society for Information Science*, 1998. 49(7): p. 604-618.
14. Chen, H., Shankaranarayanan, G., She, L., and Iyer, A., *A machine learning approach to inductive query by examples: an experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing*. *Journal of the American Society for Information Science*, 1998. 49(8): p. 639-705.
15. CITIDEL, *Computing and Information Technology Interactive Digital Educational Library*, www.citidel.org. 2004.
16. Clack, C., Farrington, J., Lidwell, P., and Yu, T., *Autonomous document classification for business*. in *The First International Conference on Autonomous Agents - AGENTS-97*. p. 201-208. 1997. Marina del Rey, California, USA.
17. Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C., *Introduction to Algorithms*. 2001, Cambridge, MA, USA: MIT Press and McGraw-Hill.
18. D'Alessio, S., Murray, K., Schiaffino, R., and Kershenbaum, A., *Category levels in hierarchical text categorization*. in *Proceedings of 3rd Conference on Empirical Methods in Natural Language Processing - EMNLP-98*. 1998. Granada, ES.
19. Dean, J. and Henzinger, M.R., *Finding Related Pages in the World Wide Web*. in *Proceedings of the 8th International WWW Conference*. p. 1467-1479. 1999. Toronto, Canada.
20. Dechter, R. and Pearl, J., *Generalized best-first search strategies and the optimality of A**. *Journal of the ACM (JACM)*, 1985. 32(3): p. 505-536.
21. DLRL, *Virginia Tech Digital Library Research Laboratory*, <http://www.dlib.vt.edu>.
22. DMOZ, *Directory Mozilla*, <http://www.dmoz.org>.
23. DogPile, *Meta-search engine*. <http://www.dogpile.com/>.

24. Dumais, S. and Chen, H., *Hierarchical classification of Web content*. in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 00* p. 256-263. 2000. Athens, Greece.
25. Eggermont, J., Kok, J.N., and Kusters, W.A., *Genetic Programming for Data Classification: Refining the Search Space*. in *Proceedings of the 15th Belgium/Netherlands Conference on Artificial Intelligence - BNAIC 03*. p. 123-130. 2003. Nijmegen, Netherlands.
26. Fan, W., Fox, E.A., Pathak, P., and Wu, H., *The effects of fitness functions on genetic programming-based ranking discovery for web search*. JASIST, 2004. 55(7): p. 628-636.
27. Fan, W., Gordon, M.D., and Pathak, P., *Discovery of context-specific ranking functions for effective information retrieval using genetic programming*. TKDE-04, 2004. 16(4): p. 523-527.
28. Fan, W., Gordon, M.D., and Pathak, P., *A generic ranking function discovery framework by genetic programming for information retrieval*. Information Processing and Management, 2004. 40(4): p. 587-602.
29. Fan, W., Gordon, M.D., and Pathak, P., *Personalization of Search Engine Services for Effective Retrieval and Knowledge Management*. in *Proceedings of the International Conference on Information Systems 2000*. p. 20-34. 2000. Brisbane, Australia.
30. Fan, W., Gordon, M.D., Pathak, P., Xi, W., and Fox, E.A., *Ranking Function Optimization For Effective Web Search By Genetic Programming: An Empirical Study*. in *Proceedings of 37th Hawaii International Conference on System Sciences*. p. 105-112. 2004. Hawaii, USA.
31. Fan, W., Luo, M., Wang, L., Xi, W., and Fox, E.A., *Tuning before feedback: combining ranking function discovery and blind feedback for robust retrieval*. in *Proceedings of the 27th Annual International ACM SIGIR Conference*. p. 138-145. 2004. Sheffield, U.K.
32. Flake, G.W., Lawrence, S., and Giles, C.L., *Efficient Identification of Web Communities*. in *Proceedings of the sixth ACM SIGKDD international conference*

- on knowledge discovery and data mining*. p. 150-160. 2000. Boston, Massachusetts, USA.
33. Gagné, C., Parizeau, M., and Dubreuil, M., *The Master-Slave Architecture for Evolutionary Computations Revisited*, in *Lecture Notes in Computer Science*. 2003, Springer / Heidelberg: Berlin. 2724: p. 1578-1579.
 34. Gauch, S., Wang, G., and Gomez, M., *ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines*. *Journal of Universal Computer Science*, 1996. 2(9): p. 637-649.
 35. Gibson, D., Kleinberg, J., and Raghavan, P., *Inferring Web Communities from Link Topology*. in *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*. p. 225-234. 1998. Pittsburgh, Pennsylvania, USA.
 36. Glover, E.J., Gordon, M.D., and Birmingham, W.P., *Improving web search using utility theory*. in *Proceedings of Web Information and Data Management - WIDM 98*. p. 5-8. 1998. Bethesda, MD, USA.
 37. Glover, E.J., Lawrence, S., Birmingham, W.P., and Giles, C.L., *Architecture of a Metasearch Engine that Supports User Information Needs*. in *Eighth International Conference on Information and Knowledge Management - CIKM 99*. p. 210-216. 1999. Kansas City, MO, USA.
 38. Glover, E.J., Lawrence, S., Gordon, M.D., Birmingham, W.P., and Giles, C.L., *Web Search -- Your Way*. *Communications of the ACM*, 2001. 44(12): p. 97-102.
 39. Gonçalves, M.A., France, R.K., and Fox, E.A., *MARIAN: Flexible Interoperability for Federated Digital Libraries*, in *Lecture Notes in Computer Science*. 2001, Springer / Heidelberg: Berlin. 2163: p. 173-186.
 40. Goncalves, M.A., Luo, M., Shen, R., Ali, M.F., and Fox, E.A., *An XML log standard and tool for digital library logging analysis*. in *Research and Advanced Technology for Digital Libraries, 6th European Conference - ECDL 02*. p. 129-134. 2002. Rome, Italy.
 41. Goncalves, M.A., Panchanathan, G., Ravindranathan, U., Krowne, A., Fox, E.A., Jagodzinski, F., and Cassel, L., *The XML log standard for digital libraries: Analysis, evolution, and deployment*. in *Proceedings of Third Joint ACM / IEEE-CS Joint Conference on Digital Libraries - JCDL 03*. p. 312-314. 2003. Houston.

42. Gordon, M., *Probabilistic and Genetic Algorithms for Document Retrieval*. CACM, 1988. 31(10): p. 1208-1218.
43. Gordon, M., *User-based document clustering by redescribing subject descriptions with a genetic algorithm*. Journal of the American Society for Information Science, 1991. 42(5): p. 311-322.
44. Govert, N., Lalmas, M., and Fuhr, N., *A probabilistic description-oriented approach for categorizing web documents*. in *Proceedings of the 8th International Conference on Information and Knowledge Management CIKM 99*. p. 475-482. 1999. Kansas City, Missouri, USA.
45. Grobelnik, M., *Feature selection for classification based on text hierarchy*. in *Proceedings of the Workshop on Learning from Text and the Web, Conference on Automated Learning and Discovery*. 1998. Pittsburgh, PA, USA.
46. Gulli, A. and Signorini, A., *The indexable web is more than 11.5 billion pages*. in *Special interest tracks and posters of the 14th international conference on World Wide Web*. p. 902 - 903. 2005. Chiba, Japan.
47. Harangsri, B., Shepherd, J., and Ngu, A., *Selectivity estimation for joins using systematic sampling*. in *Proceedings of Eighth International Workshop on Database and Expert Systems Applications*. p. 384-389. 1997. Toulouse, France.
48. Howe, A.E. and Dreilinger, D., *SavvySearch: A Metasearch Engine That Learns Which Search Engines to Query*. AI Magazine, 1997. 18(2): p. 19-25.
49. Hull, D., *Using statistical testing in the evaluation of retrieval experiments*. in *Proceedings of the 16th annual international ACM SIGIR conference on research and development in information retrieval*. p. 329-338. 1993. Pittsburgh, Pennsylvania, USA.
50. Joachims, T., *Text categorization with support vector machines: learning with many relevant features*. in *Proceedings of 10th European Conference on Machine Learning - ECML 98*. p. 137-142. 1998. Chemnitz, Germany.
51. Joachims, T., Cristianini, N., and Shawe-Taylor, J., *Composite kernels for hypertext categorisation*. in *Proceedings of 18th International Conference on Machine Learning - ICML 01*. p. 250-257. 2001. Williams College, USA.

52. Kishore, J.K., Patnaik, L.M., Mani, V., and Agrawal, V.K., *Application of genetic programming for multicategory pattern classification*. IEEE TEC-00, 2000. 4(3): p. 242-258.
53. Kishore, J.K., Patnaik, L.M., Mani, V., and Agrawal, V.K., *Genetic programming based pattern classification with feature space partitioning*. Information Sciences, 2001. 131(1-4): p. 65-86.
54. Kitsuregawa, M., Toyoda, M., and Pramudiono, I., *WEB Community Mining and WEB Log Mining: Commodity Cluster Based Execution*. in *Proceedings of the 13th Australasian Database Conference*. p. 3-10. 2002. Melbourne, Australia.
55. Kleinberg, J.M., *Authoritative sources in a hyperlinked environment*. Journal of the ACM (JACM), 1999. 46(5): p. 604-632.
56. Kluev, V., *Compiling Document Collections from the Internet*. SIGIR Forum, 2000. 34(2): p. 9-12.
57. Koller, D. and Sahami, M., *Hierarchically classifying documents using very few words*. in *Proceedings of the Fourteenth International Conference on Machine Learning - ICML 97* p. 170-178. 1997. San Francisco, CA, USA.
58. Koza, J.R., *Genetic programming: On the programming of computers by natural selection*. 1992, Cambridge, MA, USA: MIT Press.
59. Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A., *Trawling the Web for Emerging Cyber-Communities*. in *Proceedings of 8th International World Wide Web Conference*. p. 1481-1493. 1999. Toronto, Canada.
60. Kumar, R., Raghavn, P., Rajagopalan, S., and Tomkins, A., *Extracting Large-Scale Knowledge Bases from the Web*. in *Proceedings of the 25th International Conference on Very Large Data Bases Conference*. p. 639-650. 1999. Edinburgh, Scotland, UK.
61. Langdon, W.B., *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming!* 1998, Boston: Kluwer Academic Publishers.
62. Lawrence, S. and Giles, C.L., *Accessibility of information on the web*. Nature, 1999. 400(July 8): p. 107-109.

63. Lawrence, S. and Giles, C.L., *Inquirus, The NECI Meta Search Engine*. in *Proceedings of Seventh International World Wide Web Conference*. p. 95-105. 1998. Brisbane, Australia.
64. Lawrence, S. and Giles, C.L., *Searching the World Wide Web*. Science, 1998. 280(5360): p. 98.
65. Martin-Bautista, M.J., Vila, M., and Larsen, H.L., *A fuzzy genetic algorithm approach to an adaptive information retrieval agent*. *Journal of the American Society for Information Science*, 1999. 50(9): p. 760-771.
66. McCallum, A., Nigam, K., Rennie, J., and Seymore, K., *A Machine Learning Approach to Building Domain-Specific Search Engines*. in *Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI 99*. p. 662-667. 1999. Stockholm, Sweden.
67. Mitchell, T.M., *Machine learning*. 1996, New York, USA: McGraw Hill.
68. Morgan, E.L., Frumkin, J., and Fox, E.A., *The OCKHAM Initiative - Building Component-based Digital Library Services and Collections*. *D-Lib Magazine*, 2004. 10(11).
69. Najork, M. and Wiener, J.L., *Breadth-First Search Crawling Yields High-Quality Pages*. in *Proceedings of the 10th International WWW Conference*. p. 114-118. 2001. Hong Kong, China.
70. NDLTD, *Networked Digital Library of Theses and Dissertations*, <http://www.ndltd.org>.
71. Nguyen, H. and Haddawy, P., *The Decision-Theoretic Video Advisor*. in *AAAI Workshop on Recommender Systems*. p. 77-80. 1998. Menlo Park, California, USA.
72. NSDL, *National Science Digital Library*, <http://nsdl.org>.
73. OAI, *Open Archives Initiative*, <http://www.openarchives.org/>.
74. Pant, G. and Srinivasan, P., *Learning to crawl: Comparing classification schemes*. *ACM Transactions on Information Systems (TOIS)*, 2005. 23(4): p. 430-462.
75. Petry, F., Buckles, B., Prabhu, D., and Kraft, D., *Fuzzy information retrieval using genetic algorithms and relevance feedback*. in *Proceedings of the ASIS Annual Meeting*. p. 122-125. 1993. Columbus, OH, USA.

76. Qin, J. and Chen, H., *Using Genetic Algorithm in Building Domain-Specific Collections: An Experiment in the Nanotechnology Domain*. in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences - HICSS 05*. p. 102.2. 2005. Hawaii, USA.
77. Qin, J., Zhou, Y., and Chau, M., *Building domain-specific web collections for scientific digital libraries: a meta-search enhanced focused crawling method*. in *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*. p. 135-141. 2004. Tucson, AZ, USA.
78. Raghavan, V. and Agarwal, B., *Optimal determination of user-oriented clusters: an application for the reproductive plan*. in *Proceedings of the Second International Conference on Genetic Algorithms and their Application*. p. 241-246. 1987. Cambridge, Massachusetts, United States.
79. Ravindranathan, U., Shen, R., Goncalves, M.A., Fan, W., Fox, E.A., and Flanagan, J.W., *ETANA-DL: a digital library for integrated handling of heterogeneous archaeological data*. in *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital Libraries*. p. 76-77. 2004. Tucson, AZ, USA.
80. Ravindranathan, U., Shen, R., Goncalves, M.A., Fan, W., Fox, E.A., and Flanagan, J.W., *ETANA-DL: managing complex information applications -- an archaeology digital library*. in *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital Libraries*. p. 414. 2004. Tucson, AZ, USA.
81. Rijsbergen, C.J.V., *Information Retrieval*. 1979, Newton, MA, USA: Butterworth-Heinemann.
82. Robertson, S.E., Walker, S., and Beaulieu, M.M., *Okapi at TREC-4*. in *TREC-4*. p. 73-96. 1995.
83. Salton, G., *Automatic Text Processing*. 1989, Boston, Massachusetts, USA: Addison-Wesley.
84. Salton, G. and Buckley, C., *Term-weighting approaches in automatic text retrieval*. *IPM*, 1988. 24(5): p. 513-523.
85. Selberg, E. and Etzioni, O., *The MetaCrawler architecture for resource aggregation on the Web*. *IEEE Expert*, 1997. 12(1): p. 11-14.

86. Selberg, E. and Etzioni, O., *Multi-Service Search and Comparison Using the MetaCrawler*. in *Proceedings of the 4th International World-Wide Web Conference*. p. 195-208. 1995. Boston, Massachusetts, USA
87. Suleman, H., Atkins, A., Gonçalves, M.A., France, R.K., Fox, E.A., Chachra, V., Crowder, M., and Young, J., *Networked Digital Library of Theses and Dissertations Bridging the Gaps for Global Access - Part 1: Mission and Progress*. D-Lib Magazine, 2001. 7(9).
88. Suleman, H., Atkins, A., Gonçalves, M.A., France, R.K., Fox, E.A., Chachra, V., Crowder, M., and Young, J., *Networked Digital Library of Theses and Dissertations Bridging the Gaps for Global Access - Part 2: Services and Research*. D-Lib Magazine, 2001. 7(9).
89. Suleman, H. and Fox, E.A., *A Framework for Building Open Digital Libraries*. D-Lib Magazine, 2001. 7(12).
90. Suleman, H. and Fox, E.A., *The Open Archives Initiative: Realizing Simple and Effective Digital Library Interoperability*. J. Library Automation, 2002. 35: p. 125-145.
91. Suleman, H., Fox, E.A., Kelapure, R., Krowne, A., and Luo, M., *Building digital libraries from simple building blocks*. Online Information Review, 2003. 27(5): p. 301-310.
92. System X, *Virginia Tech Terascale Computing Facility: System X*, <http://www.tcf.vt.edu>. 2006.
93. Torres, R.d.S. and Falcão, A.X., *Content-Based Image Retrieval: Theory and Applications*. Revista de Informática Teórica e Aplicada, 2006. 13(2): p. 161-185.
94. Torres, R.d.S. and Falcão, A.X., *Contour Saliency Descriptors for Effective Image Retrieval and Analysis*. Image and Vision Computing, 2007. 25(1): p. 3-13.
95. Toyoda, M. and Kitsuregawa, M., *Creating a Web Community Chart for Navigating Related Communities*. in *Proceedings of ACM Conference on Hypertext and Hypermedia*. p. 103-112. 2001. Århus, Denmark.
96. Weigend, A.S., Wiener, E.D., and Pedersen, J.O., *Exploiting hierarchy in text categorization*. Information Retrieval, 1999. 1(3): p. 193-216.

97. Yang, J. and Korfhage, R.R., *Effects of query term weights modification in document retrieval: a study based on a genetic algorithm*. in *Proceedings of the Second Annual Symposium on Document Analysis and Information Retrieval*. p. 271-285. 1993. Las Vegas, NV.
98. Yang, J., Korfhage, R.R., and Rasmussen, E., *Query improvement in information retrieval using genetic algorithms: a report on the experiments of the TREC project*. in *Proceedings of the First Text Retrieval Conference - TREC 1*. p. 31-58. 1993.
99. Yang, Y., *Expert network: effective and efficient learning from human decisions in text categorization and retrieval*. in *Proceedings of 17th ACM International Conference on Research and Development in Information Retrieval - SIGIR 94*. p. 13-22. 1994. Dublin, Ireland.
100. Yang, Y. and Liu, X., *A re-examination of text categorization methods*. in *Proceedings of the 22nd Annual International ACM SIGIR Conference*. p. 42-49. 1999. Berkeley, USA.
101. Zhang, B., *Intelligent Fusion of Evidence from Multiple Sources for Text Classification*. PhD thesis, Virginia Polytechnic Institute and State University, Department of Computer Science, 2006.
102. Zhang, B., Chen, Y., Fan, W., Fox, E.A., Gonçalves, M.A., Cristo, M., and Calado, P., *Intelligent Fusion of Structural and Citation-Based Evidence for Text Classification*. in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. p. 667-668. 2005. Salvador, Brazil.
103. Zhang, B., Chen, Y., Fan, W., Fox, E.A., Gonçalves, M.A., Cristo, M., and Calado, P., *Intelligent GP Fusion from Multiple Sources for Text Classification*. in *Proceedings of the 14th Conference on Information and Knowledge Management*. p. 477-484. 2005. Bremen, Germany.
104. Zhang, B., Gonçalves, M.A., Fan, W., Chen, Y., Fox, E.A., Calado, P., and Cristo, M., *Combining Structure and Citation-Based Evidence for Text Classification*. in *Proceedings of the 13th Conference on Information and Knowledge Management*. p. 162-163. 2004. Washington D.C., USA.

105. Zhang, B., Gonçalves, M.A., Fan, W., Chen, Y., Fox, E.A., Calado, P., and Cristo, M., *A Genetic Programming Approach for Combining Structural and Citation-Based Evidence for Text Classification in Web Digital Libraries*, in *Soft Computing in Web Information Retrieval: Models and Applications*. 2006: p. 65-83.