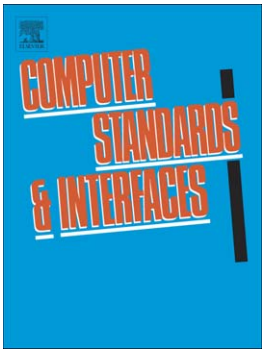
Accepted Manuscript



A Focused Crawler Combinatory Link and Content Model Based on T-Graph Principles

Ali Seyfi, Ahmed Patel

PII: S0920-5489(15)00073-2
DOI: doi: 10.1016/j.csi.2015.07.001
Reference: CSI 3043

To appear in: *Computer Standards & Interfaces*

Received date: 24 August 2014
Revised date: 5 July 2015
Accepted date: 9 July 2015

Please cite this article as: Ali Seyfi, Ahmed Patel, A Focused Crawler Combinatory Link and Content Model Based on T-Graph Principles, *Computer Standards & Interfaces* (2015), doi: 10.1016/j.csi.2015.07.001

A Focused Crawler Combinatory Link and Content Model Based on T-Graph Principles

Ali Seyfi [a,*] and Ahmed Patel [b,c]

[a] Department of Computer Science, The George Washington University, Washington DC, United States
[b] Faculty of Computer Science and Information Systems, Jazan University, Saudi Arabia
[c] Visiting Professor, Faculty of Science, Engineering and Computing, Kingston University, United Kingdom

Abstract

The two significant tasks of a focused Web crawler are finding relevant topic-specific documents on the Web and analytically prioritizing them for later effective and reliable download. For the first task, we propose a sophisticated custom algorithm to fetch and analyze the most effective HTML structural elements of the page as well as the topical boundary and anchor text of each unvisited link, based on which the topical focus of an unvisited page can be predicted and elicited with a high accuracy. Thus, our novel method uniquely combines both link-based and content-based approaches. For the second task, we propose a scoring function of the relevant URLs through the use of T-Graph (Treasure Graph) to assist in prioritizing the unvisited links that will later be put into the fetching queue. Our Web search system is called the *Treasure-Crawler*. This research paper embodies the architectural design of the Treasure-Crawler system which satisfies the principle requirements of a focused Web crawler, and asserts the correctness of the system structure including all its modules through illustrations and by the test results. The Treasure-Crawler is evaluated with both precision and recall values close to 0.5, which indicate the significance of the proposed architecture.

Keywords: Focused Web Crawler, Robot, T-Graph, HTML Data, Information Retrieval, Search Engine

1. Introduction

The openly indexable World Wide Web has empirically passed 60 trillion documents, with more than 50 billion pages indexed daily [1-3] and hitherto, growth does not appear to level off. Considering the exponentially increasing amount of dynamic content on the Web, such as news, schedules, social networks and individual data, it is evidently asserted that relying on search engines is inevitable for the people to approach their desired information. Whereof these concerns make searching the Web a profound task, experts apply machine learning algorithms to accomplish several phases of this job such as ranking retrieved Web pages based on their estimated relevance to the user query. The main goal is to make up the best weighting algorithm that could represent queries and pages as in a vector space. This way, the closeness in such a space would convey the semantic relevance.

A Web crawler systematically collects information about documents from the Web to create an index of the data it is searching and it maintains an updated index through subsequent crawls. As an automatic indexer, the crawler operates in the context of listing the documents relevant to a subject or topic which one would expect in a typical user search query. Traditional general purpose Web crawlers are not easily scalable since the Web is not under the control of one operator or proprietary. They also may not be set to target specific *topics* for accurate indexing, and lag behind in time and updates to manage updated indexes/indices of the whole Web to stay current because of the distribution, subject and volume involved. To overcome these shortcomings, focused crawlers are intended to rely on the linked structure of the Web in order to identify and harvest *topically relevant* pages to increase their performance in terms of accuracy, currency and speed. A significant benefit in using focused crawlers is the possibility of decentralizing the resources and storage indexes.

---

[*] Corresponding author:

E-mail addresses: seyfi@gwu.edu (A. Seyfi), whinchat2010@gmail.com (A. Patel)

There are two major open problems in focused crawling: the first problem is the prediction of the topical focus of an unvisited page before actually attempting to download the content of the page. As one of its fundamental tasks, the crawler should use specific algorithms in order to make this prediction with the highest possible accuracy. Typically, the focused crawlers download the whole content of the page, analyze it and make a decision on whether it is related to their topic of interest. Alternatively, some researches [4-6] show that the topical focus of a page can be predicted by analyzing the anchor text of its link in the parent page. Between these two extremes, in our research we take into account several HTML structural elements of the parent page, in addition to the anchor text. This will help improve the accuracy of the topical detection of the unvisited link. The second problem is prioritizing the links for later downloads. A proper priority score should be assigned to all the extracted URLs from a Web page. These URLs along with their scores will then be put in a queue to be downloaded later. This task of prioritization is very important in that some irrelevant pages should be visited and passed off on the way to reach another populated area of relevant pages. For prioritization, we employ a novel tree-like data structure called T-Graph (Treasure Graph) [7]. The traversal method in the construction of T-Graph can be both top-down or bottom-up. Also, in our proposed system which is called the Treasure-Crawler, no resources from any other search engines are required, provided the T-Graph is set to be constructed top-down.

This study is conducted to determine a novel crawler's effectiveness and viability in crawling to fetch topic specific documents. The system is designed and implemented in a way that all the components/modules have the minimum dependency on each other; hence each can be changed or replaced while requiring the least manipulation of other parts except for some interfaces. The document standards such as HTML and XML are respected for this system. Also HTTP and HTTPS communication protocol standards from IETF and W3C are considered, as well as the robot exclusion protocol known as "robots.txt".

1.1 The Methodology

One of the main objectives of this research has been to enhance the accuracy of Web page classification, which is made possible by defining a flexible interpretation of the surrounding text, in addition to specific HTML data. The Dewey Decimal Classification (DDC) system is applied as a basis to classify the text into appropriate topic/subject boundaries. The other significant objective of this research has been to reach target documents in the shortest possible time. This is accomplished by reaching the T-Graph most matching node(s) with the document, and then calculating the distance of such nodes to the target level based on the number of documents to download [7]. As a result, due to a better determination of the topic boundary and significant decrease of the volume of downloaded documents in text format, this strategy helps the crawler update its indexes more pragmatically, accurately and rapidly. Based on these assumptions, we designed a new algorithm and built a prototype to exercise, test and validate it against the functional and non-functional requirements. The summary results are only presented in this paper, while the actual detailed evaluation results are reported in a follow-up paper with the title of "Analysis and Evaluation of the Link and Content Based Focused Treasure-Crawler" [8].

In the rest of the current paper, we consecutively review some major Web crawlers and their reported experimental results, detail the requirements and evaluation criteria of a focused Web crawler, present the architectural design of the Treasure-Crawler, elaborate the employed methods and algorithms, describe the outcome of the carried-out tests and validations, and conclude with a summarized list of results and conclusions.

2. Background

Topical and focused crawling are two slightly different concepts, first introduced by Menczer [9] and by Chakrabarti [10] respectively. A focused crawler selects and seeks out pages that are relevant to (a) pre-defined topic(s). It systematically analyzes its crawl frontiers and tries to detect the pages that are most likely to be on the designated topic(s) of the crawl, while it tries to avoid off-topic Web regions. As a result, this process brings considerable savings in different resources, such as network, computing, and storage, hence, the crawl becomes more up-to-date. Usually, the topics of interest are optionally defined by keywords, categorized/classified standard lexicon entries, or by a set of exemplary documents. The major challenge of a focused Web crawler is the capability of predicting the relevance of a given page before actually crawling it. Achieving this goal requires particular intelligence for the crawler. Focused Web crawlers use this kind of intelligence to avoid irrelevant regions of the Web in order to make the task manageable. Additionally, a focused Web crawler should also pay attention to the ability to discover relevant regions which are separated by groups of irrelevant Web regions in order to achieve desirable Web coverage. A well-designed focused Web crawler should be able to stay in pre-defined topics as long as possible, while covering the Web as much as possible.

To index the Web, the first generation of crawlers relied on basic graph traversal algorithms, namely the breadth-first or depth-first. An initial set of URLs is used as a seed set, and the algorithm recursively follows hyperlinks down to other documents. Document content is of less importance since the ultimate goal is to cover

the whole Web. A focused crawler, on the other hand, explores the documents about a specific (set of) topic(s) and guides the searching process based on both the content and link structure of the Web. The main strategy is to associate a score with each unvisited link within the downloaded pages.

Best-first is basically an optimization to the Breadth-first algorithm. When the unvisited links are extracted, an estimator tries to prioritize them. After being associated with a priority score, these links are inserted into a priority queue. These links are then fetched from the queue according to their assigned priority.

Panorama (1996) is one of the first systems that established a digital library by using the Web and made CiteSeer, the most popular scientific literature digital library and search engine, focusing on information technology and computer science. Panorama traverses the Web for on-topic documents in PDF and Postscript formats in the computer science field. To construct its topic of interest, Panorama submits relevant papers' main titles and the titles of references within the papers as separate exact phrase queries to Google Web APIs. Thus, the returned URLs form a positive example set, and examples from unrelated papers form a negative example set. Both of the two sets train a Naïve Bayes classifier, which guides the crawling process [11].

Shark Search Crawler (1998) tries to enter the areas where a higher population of relevant pages is observed, but stops searching in the areas with no or very little number of relevant pages.

InfoSpiders (1999) constitutes a population of adaptive intelligent agents. These agents use neural networks algorithms and are very advanced in distinguishing fruitful links to follow.

Context Focused Crawler (2000) builds upon constructing a multi-level tree of sample documents called the context graph. This algorithm, first proposed by Diligenti et al. [12], has been an active research area in the past decade.

Figure 1 shows a context graph. For each target document (e.g. P) one such graph is constructed. As the target documents are supplied to the system in Level 0, another search engine is utilized to find the high rank pages that contain a link to the current target document (e.g. A and B) and they are put in Layer 1. This process is then recursively repeated for each parent page until the graph reaches a desired number of levels. By convention, there is no connection between the nodes in a common layer. Also, if two (or more) documents in layer *i* (e.g. A and B) have a common parent in layer *i+1* (e.g. C), the parent should appear two (or more) times in layer *i+1*. Note that Layer 1 Web pages are one step away from desirable documents and Layer 2 Web pages are exactly two steps away from desirable documents and so on. When the context graphs are constructed they are merged to a single graph called the merged context graph. If there are n layers in the merged context graph, n classifiers are trained to assign Web pages into its different layers. The assigned level to each page defines its priority, which is the closeness of the page to the desired topic. As a result, the context graph can learn topics that are indirectly connected to a pre-defined topic and thus increases recall.

Also, the arrangement of the nodes in layers represents any hierarchical content structure. Related content is placed close to the center, while the outer layers accommodate more general pages. The idea of using the context of a given topic to guide the crawling process could significantly increase both precision and recall.

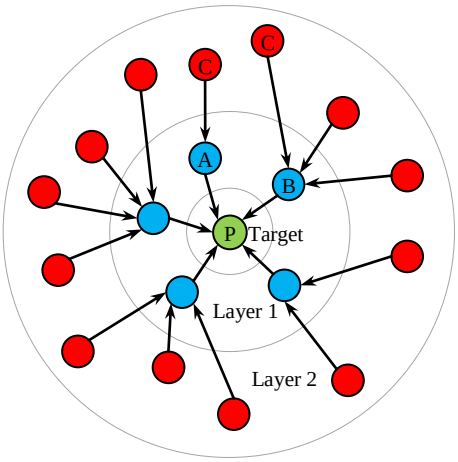


Figure 1. A context graph

Tunneling (2001) is an interesting concept where a crawler finds relevant regions (or pages) while traversing a path that does not only contain relevant pages [13]. The major task of focused Web crawlers is to unveil as many bridges among relevant regions as possible. Martin Ester et al. implemented their tunneling technique

based on taxonomies. During the crawling process, the crawler keeps an eye on the precision. If the precision goes down much faster than expected, it is necessary to broaden the focus of the crawl. For example, if a focused Web crawler is currently working on the topic "basketball" and cannot find more relevant pages, it can generalize the topic to "sport" and expect to find more bridges leading to other relevant regions. If the precision gets better than a pre-defined value after the generalization, the topic is specialized again. Taxonomy based tunneling seems an ideal and intuitive approach to increase recall.

Meta Search Crawler (2004) is another approach to build Web collections for scientific digital libraries [14]. The author addressed the drawbacks of the traditional focused Web crawlers caused by local search algorithms in the digital library construction. The meta-search crawler follows a novel approach; making queries from a domain/topic-specific dictionary/lexicon and combining the entries on the first page of the results from multiple search engines. This way, the crawler will cover diverse Web communities of documents which are presumed to have very little overlap. This approach is very effective in terms of the accuracy in remaining on the topic while showing a high Web coverage. The experimental results of the meta-search crawler show the maximum precision value of 50.23% which is almost equal to the precision value of our Treasure-Crawler. However, it is controversial if using and relying on other search engines is effective or not.

LSCrawler (2006) considers better recall as its goal and uses link semantics to guide the crawling process [15]. The LSCrawler takes into account the semantic of anchor text of the link in order to specify the relevancy of the unvisited page. Since anchor texts are often too short to provide adequate information to arrange a reasonable crawling sequence, content texts and URL patterns are optionally considered as well. However, in their paper, Yuvarani et al. do not propose a clear description of the surrounding text. LSCrawler is basically a semantic crawler as it utilizes the corresponding ontology of its topical focus to make assessments on a URL. Although the structure of this algorithm has useful commons with our proposed algorithm, relying on its experimental results may not be insightful because of its semantic basis. They report a recall value of ~0.6 which is a good result although the number of the crawled pages is not indicated and an unlimited increase of the recall value is observed, which should normally approach a constant level.

Hybrid Focused Crawler (2006) uses two features of documents; link structure and content similarity of each page to the main topic to crawl the Web [4]. Initially the topic of interest is set for the crawler to launch the crawling using a single seed URL. All Web pages are mapped into one of three groups: Seed Pages, Candidate Crawled Pages and Uncrawled Pages. Seed Pages are repeatedly selected from Candidate Crawled Pages using a rank score. Only hyperlinks derived from Seed Pages are crawled. All crawled pages are first stored in the Candidate Crawled Pages table and their rank scores are calculated after each session. The one with the highest mark will be moved to the Seed Pages table. The rank score is calculated using a combination of the "link structure" and "content similarity" as below:

$$Rank\ (p)\ =\ (links\_to\_seed\ (p)\ +\ links\_from\_seed\ (p))\ \times\ (0.1\ +\ content\_similarity(p))$$

where p is a page from Candidate Crawled Pages; *links_to_seed* and *links_from_seed* are respectively the number of links from the page *p* to Seed Pages and the number of links from Seed Pages to it; *content_similarity* is a real number between 0 and 1 indicating the similarity degree of a page to a topic. From the above formula, we can observe that this algorithm emphasizes the link structure over content similarity and it is essentially a compromise of the PageRank algorithm. Using Seed Pages to mimic important pages in the PageRank algorithm is a promising way to remove the requirement of a large on-hand Web link structure database but it is also a potential limitation where the accuracy of this algorithm largely depends on the precision of Seed Pages. The *content_similarity* is calculated using vocabulary vectors and simple keyword matching. They deemed that its classification accuracy cannot be as good as SVM classifiers or Naïve Bayes classifiers. In addition, using only one seed URL to start the crawling process is an obvious flaw.

The Web crawler proposed by Jamali et al. [4] demonstrates a high harvest ratio in the retrieval of on-topic pages, although it follows a rather different methodology than our proposed system. Their system is intended to first download and then determine the relevancy of a Web page. This approach, however, is presumed to have a high overload on the performance of the system as the Web is rapidly increasing in size. In their experiment, Jamali et al. reported a harvest ratio of 67.2% ~ 85.5%, which is a significant outcome after crawling 25000 Web pages.

Relevancy Context Graph (2006) is designed based on the context graph algorithm [16]. There are two major problems with the context graph algorithm. The strict link distance definition of this algorithm brings too much classification burden on the system. Also, this homogenous link requirement lowers the classification accuracy because it does not tally with the real topology of all the topics on the Web. To overcome these problems, Hsu and Wu [16] proposed the relevancy context graph structure and reached an approximate value of ~0.22 for the precision in the category of "mountain".

HAWK (2008) is simply a combination of some well-known content-based and link-based crawling approaches, and results in a decreasing value of harvest ratio (0.6) after crawling 5000 pages [17]. This value is quite acceptable; however, its decreasing rate is daunting because it is not clear what constant value it will gradually reach.

Modified NB (2010) implements a search system, where the classifier utilizes reinforcement learning in order to train itself against topic taxonomy. It also stems and assigns proper weight to the textual data as well as to the URLs and anchor texts [5]. For this weight assignment, it uses a modified form of TF-IDF, and for a better classification, an adopted form of Naïve Bayes classifier is employed. The experimental results of this system, in comparison with our proposed system, show a better harvest ratio.

Intelligent Focused Crawler (2011) employs a modified form of the Naïve Bayes classifier and results in a significant harvest ratio. For this crawler, Taylan et al. investigated several link scoring methods, namely an updatable form and a threshold-restricted form of the Naïve Bayes classifier [6]. Their fruitful results demonstrate that the Naïve Bayes with threshold approach gives a higher harvest ratio among other algorithms. The two modifications they have made to the Naïve Bayes classification method improve the crawling performance in terms of the harvest ratio (around 80%), which in comparison with our system is much higher and more effective. The problem with their experiment is that they have only crawled 1000 Web pages which is not a satisfactory number to represent the whole Web.

OntoCrawler (2011) follows a fuzzy approach in weighting the concepts and page contents, while relying on ontologies in the semantic Web [18]. It demonstrates an increasing precision value in a crawl of 1000 pages, unlike other previously introduced algorithms in focused crawling. To demonstrate its improved performance, the precision rate of 90% of the OntoCrawler was compared to the resulted precision of the Best-First and Breadth-First crawlers on the topic of football. Although our proposed system has only a precision of ~50%, it is not boosted with fuzzy logic and semantic Web capabilities as in the OntoCrawler.

Clickstream-Based Parallel Crawler (2012) interestingly ignores the link-based structure of downloaded Web pages and takes the history clickstream analysis parameter into account to carry out a reliable URL prioritization on the frontier queue. It is noteworthy that this parallel crawler system needs authorized access to configured server log files through specific standard(s) [19]. As for the Web activity analysis, this approach highly depends on the previous users' activity and requires more improvements in order to best work on newly created pages. This can be an additional plug-and-play component for our design to increase the classification process.

BDS for SIS Crawlers (2014) is an attempt to solve the communication problem between social networks aka Social Internetworking Systems. This type of interaction is claimed to be impossible using BFS, MH, and RW approaches. The authors design a novel crawling technique called Bridge Driven Search which is applicable only to the problems of this sort and proven to have a high flexibility which is an important criterion in the context of SIS [20]. Our designed crawler will face similar situations at some point, where there are slight connections between highly populated on-topic areas on the Web. BDS can be a further improvement for such cases.

Beam Classifier (2014) follows the fact that decision trees (or just simply trees) are the best comprehensible structures to classify the lexical datasets. However, instead of using the greedy nature of DTs, Beam Classifier (extending the concept of LEGAL-Tree) helps the classification process be more accurate. In their paper, Basgalupp et. al. introduce a fitness function which intelligently takes into account the accuracy and F-measure to evaluate a DT's performance. Based on the analysis results, the Beam Classifier is justified to be highly flexible and accurate while used as a lexicographic algorithm [21]. This approach can be improved by using Trie data structure because of its linear time complexity. We are now trying to incorporate such solution into our system.

Table 1 juxtaposes the architecture of selected crawlers from the list above and that of the Treasure. A comparison of the experimental results has also been carried out but is beyond the scope of this paper, and will be discussed in our next article on the evaluation of the Treasure-Crawler. The significance of our proposed architecture is in employing a simple custom method to classify each unvisited Web page, as compared to other more complicated techniques used in the literature of focus crawlers. Also, the usage of the T-Graph as the exemplary dataset for the task of URL prioritization is one more important advancement.

Table 1. Comparison of different crawlers' relevance calculation and classification methods

Focused Crawler (FC)	Relevance Calculation					Classification			
	Whole Content	URL	Anchor Text	Terms Around Links	Custom Method	Naïve Bayes	Decision Tree	Custom Model	Semantic
CFC	*					*	*		
LSCrawler		*	*	*				*	*
Relevancy CFC		*	*	*					
Hybrid FC	*							*	
Meta Search					*			*	
HAWK	*	*	*	*				*	
Intelligent FC					*	*			
Modified NB		*	*			*		*	
OntoCrawler	*						*	*	*
Treasure-Crawler		*	*	*	*		*	*	

3. Requirements and Evaluation Criteria

In order to define the requirements of a focused crawler, some scenarios are used that describe how the system works in practice. These scenarios are helpful in requirements elicitation, elaboration and validation. To describe the scenarios of the proposed system, the Viewpoint-Oriented method (VORD) [22] is used with the following steps:

1. System admin provides the data structure (tree or graph) with a sample data set of interlinked Web pages, and also provides the crawler with seed URLs.
2. Data structure automatically downloads the required information from the Web to build and update itself. This step may be taken repeatedly in the crawling process.
3. Crawler fetches the requested Web pages from the Web.
4. Crawler hands the downloaded Web page to the main relevance calculator of the search engine.
5. Relevance calculator uses its internal algorithms and techniques along with the output from the data structure to make specific decisions on the content of the page and its unvisited URLs.
6. Web page content and all the related data are stored in the search engine repository.
7. Extracted URLs from the Web page are supplied to the crawler.

It must be stated that these requirements are listed based on the technical purposes of a focused Web crawler as existing in other state-of-the-art approaches. We obtained the requirement by studying similar crawlers as well as homing in on the topic specific nature of indexing. Also, the way these requirements are described and elaborated follows the standards of requirements elicitation as in software engineering domain. While functional requirements are quantitative (i.e. what the system is supposed to do) and must be satisfied, non-functional requirements lean toward the qualitative side and their conditions/values can relatively affect the performance of the crawler, either directly or indirectly depending on the computing system hosting the crawler.

3.1 Functional Requirements

The list of functional requirements consists of the services that the system should provide, the system's reactions to particular inputs and the system's behavior in particular situations:

System initialization*:* The system shall start from a stable point provided that all the initial inputs are supplied to the system. The system shall then receive the required inputs such as the seed URLs, documents, values and thresholds. The queues and all data structures shall be emptied, initialized and fed with the required initial data that are provided by the system administrator. All the system's hardware should be checked, including all types of server(s) and the network connection(s) along with their performance. The starting pages (seed URLs) shall be downloaded correctly and be ready for usage.

Relevance calculation: The system shall be able to calculate the relevance of a Web document based on its textual data and assign a topic code as in a specified classification system. The URLs in a Web page and their

topical boundaries should be detected and extracted. All the text data should be stemmed before any calculation. The topical focus of the stemmed textual data should be inspected by using a comparative algorithm.

Priority calculation: The system shall use a specific structure to assign a priority score to each and every unvisited link present in the downloaded Web page. The graph structure should be initialized by receiving a sample data set of interlinked documents, or only the target documents if it is designed to build itself automatically. The graph structure should search for and download the relevant pages in order to fill the nodes. This process should continue correctly until the desired number of levels and nodes is gained. The HTML data of the downloaded Web page should be extracted in order to be compared with the data in the graph nodes. This comparison needs a specific text similarity measurement algorithm. The overall priority of the unvisited link should be calculated by using a specific formula. This priority score should be then assigned to the URL.

Data storage and indexing for later search: If the downloaded Web page is recognized to be relevant to the specialized topic of the system, it shall be stored in the search engine repository. The contents of the repository are then fetched and presented in response to specific user queries. Relevant Web pages shall be stored in the search engine repository along with all their related information that is provided or calculated by the system.

3.2 Non-Functional Requirements

Defined as the constraints and limitations on the services or functions offered by the system, two examples of these requirements are I/O device capabilities and system representations. They also define the system's properties such as reliability, response time and storage requirements. It must be stated that non-functional requirements may be more critical than functional requirements. If they are not met, the system may be useless as for safety measures such as trust, protection, audit trails, security, privacy, digital forensics and identity record management. The non-functional requirements of a Web crawler are as follows:

Flexibility: The system should be usable in various scenarios with the least possible modifications.

Low cost: The system should run on inexpensive hardware and the number of servers should be minimized, provided that the overall performance is not affected.

High performance: Basically, several hundred pages should be processed per second and surely several millions per each run. Therefore, the disk access and usage is vital, especially when the data structures such as queues and graphs overfill the memory. This situation occurs when millions of pages are downloaded.

Robustness: Dealing with the Web as the largest existing collection of data and information, the crawler system interacts with millions of servers and pages. Thus, it has to tolerate corrupted HTML data, unexpected server configurations and behavior, and many other strange issues. Since each crawl consumes weeks or even months of time, the system should be able to handle network crashes and interruptions with losing the least possible amount of data.

Standardization: Following the standard conventions is highly important for a crawler. The crawler should respect the standards such as "robots.txt" which is a robot exclusion protocol [23] and "sitemaps" which is a robot inclusion protocol. In addition, supplying the crawler with a contact URL and supervising the crawl are very important.

Speed control: The system should match the crawl speed with the load of its main routers. A sliding window can limit the crawl speed at peak usage hours and increase the speed during the lower usage periods of time.

Manageability: The system should have proper interfaces to show the monitoring information, such as crawler speed, size of main data sets, and some statistics about pages.

Reconfigurability: The crawler (system) administrator should be able to change different settings and configurations, record various logs, add or remove components, and blacklist some links that the crawler should avoid traversing.

Modularity: The system should be designed and implemented modularly. This will help easy and rapid accommodation of changes in the configuration of the system.

Reliability: For each crawl of a specific number of pages, the output of the system should be reliable to be used for making further changes in the system. Also it should be guaranteed that the downloaded pages are identified and prioritized in the best possible way.

Software and hardware independency: The crawler needs to access pages that are situated on various servers with various platforms. Therefore, it should be flexible and act independent of the hardware and software it is encountering.

3.3 Evaluation Criteria

Focused crawlers share many commons with the traditional information retrieval systems and are therefore evaluated regarding the criteria that apply to all of the novel and traditional IR systems. Moreover, some of these criteria are set to be measured within the context of software architecture as well as the Web and its attributes. Although only some of these criteria are measured for academic researches, all of them should be satisfied to put a Web crawler into work. It may be possible to introduce new metrics per use, especially when time constraints are vital. In order to collect data for these evaluations, our implementation of the Treasure-Crawler records all its activities as the system is run and in a timely manner. For this purpose, all HTTP responses, detailed calculations and even temporary results are stored in a database to perform further off-line analysis. A follow up paper will elaborate these results, analysis and evaluations.

Harvest ratio: This is the rate of detecting and (high) prioritizing relevant pages, and efficiently discarding or (low) prioritizing irrelevant pages. This value should reach a threshold; otherwise an exhaustive crawler may be preferably used.

Recall: This is the proportion of relevant retrieved Web pages to all the relevant pages on the Web. To approximate the system recall, we presumed our crawled pages as the whole collection and only took the pages that were topically known within the DMOZ dataset [24].

Precision: This is the proportion of relevant retrieved Web pages to all the retrieved Web pages. This metric should also be high.

Performance: This is a general term to use but what is important is that the system should have good performance in terms of speed and other IR accuracy metrics.

Scalability: In any scale, from a small dataset to the huge Internet, the system should be adjustable and scalable. Our prototype is implemented to scale the Web as well as any provided dataset of documents and URLs, such as the DMOZ, on which it has been run and tested. However, it is not evaluated against this criterion because the results of such examination will not be substantial unless actual parallel Web servers and high bandwidth Internet connection are available.

Modularity: The system should be implemented by respecting as many standards as possible, to provide a flexible basis for further developments on different platforms. In the case of replacement of any components of the system, the other parts should require the least change. Although this criterion is not evaluated in detail, the prototype system of this study is specified, designed and implemented to follow software engineering modularity philosophy whereby one module can be easily replaced with another new one without affecting any other modules of the system. Since the internal working functions of the modules are self-contained, any required changes could be only at the interface between modules if a new or unwanted parameter exchange demands it. As we are following the object oriented approach, the classes and their objects are designed and interconnected in such a way to flexibly accommodate changes and replacements.

Robustness: The system has to tolerate corrupted HTML data, unexpected server configurations and behaviors and any other strange issues and errors. Our system detects and records these situations and makes specific decisions or provides error messages for the system administration function.

Our proposed system is implemented as a prototype and tested on a typical machine with a permanent high speed Internet connection. The network, hardware and software conditions of such experimental prototype limited our evaluations to focus on the information retrieval concepts, hence, only those criteria that represent the accuracy of the retrieval will be measured and presented. It must be stated that the prototype is designed and implemented in such a way to satisfy those criteria that relate to its software engineering fundaments (e.g. scalability, modularity, and robustness).

4. Framework and Architecture

The conceptual framework architecture of the Treasure-Crawler-based search engine is illustrated in Figure 2, followed by a description of the modules. To target the design optimality, the Treasure-Crawler architecture and prototype was tested and evaluated against a set of criteria, and the results are discussed in great detail in a follow-up related paper [8]. However, a summary of the results is given at the end of Section 6.

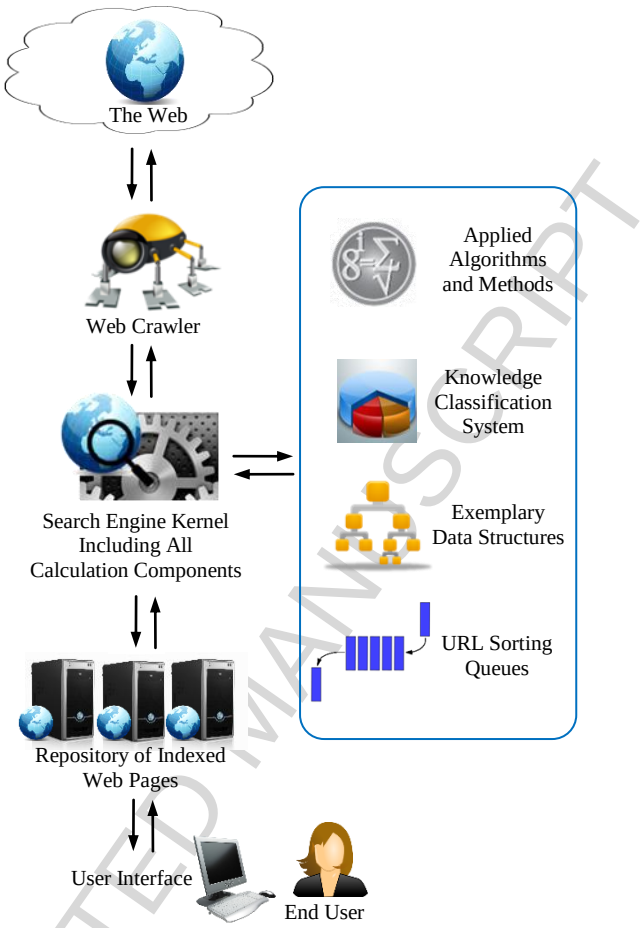


Figure 2. Conceptual framework of the Treasure-Crawler-based search engine

Web Crawler is a software agent that receives a URL from the URLs queue (supervised by the kernel) and downloads its corresponding Web page. The page is then parsed and analyzed by the kernel to be ready for the next stages. If the crawler receives any other HTTP response instead of the page content, it tries to store the URL along with its specific message.

Search Engine Kernel is the main computational component of the search engine. All the decision makings and necessary calculations are performed in the kernel. It controls all other components while utilizing their data. In focused crawling, the relevancy of the pages and priority assignment to each URL are the main duties of the kernel. In our proposed architecture, the kernel predicts the topical focus of an unvisited link through analyzing its anchor text and the content of the parent page. Thus, the system is set to combine link-based and content-based analysis.

Applied Algorithms and Methods are variously utilized to achieve the ultimate goal of the system, which is to effectively index and search the Web. In our system, these custom methods are designed to be simple while efficient. Alternatively, one may use other predefined algorithms such as specific classifiers and text similarity measurements.

Knowledge Classification System is used as a reference to classify each Web page or unvisited URL into its appropriate sub-category of human knowledge. In the context of information retrieval and library science there are several classifications systems that are widely used. For our system we chose Dewey Decimal system according to its wide range of knowledge coverage.

Exemplary Data Structure is a reference hierarchy based on which the priority of the URLs can be calculated. This reference hierarchy is basically a data structure of some sample pages. Similar to our utilized T-Graph, one can design and use a custom structure for this purpose.

URL Sorting Queues are key data structures of any search engine, wherein URLs are lined to be utilized in the intended manner. They are mostly organized as priority queues; however first-in-first-out structure is also widely used.

Repository stores an indexed version of the Web. Although this repository cannot keep the live version of the Web, it must be indexed for fast retrieval. Also the repository should stay as up-to-date as possible by use of an intelligent and fast module.

User Interface is the only user-side component of a search engine. This interface should be graphically designed in such a way to receive the search query easily and present the search results quickly and clearly.

Given the above framework, the architectural design of the Treasure-Crawler Web search system is shown in Figure 3, followed by the descriptions of the processes in a sequential order.

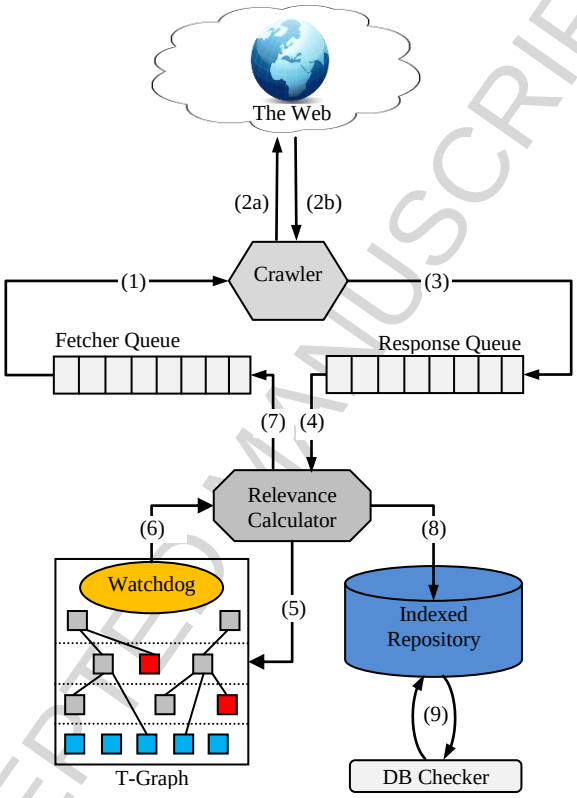


Figure 3. Architectural design of the proposed Treasure Crawler Web search system

1. The crawler component dequeues an element from the fetcher queue, which is a priority queue. Initially the seed URLs are inserted into the queue with the highest priority score. Afterwards, the items are dequeued on a highest-priority-first basis.
2. (a) The crawler locates the Web documents pointed by the current fetched URL and (b) attempts to download the actual HTML data of the page, or otherwise, the server's HTTP response.
3. For each downloaded document, the crawler deposits the retrieved HTML data or otherwise, the HTTP response (relating to temporary unfreshness or unavailability of the link) in the response queue.
4. The document is then handed to the relevance calculator which analyzes and determines the topic(s) of the document and its relevancy to the crawler's specialized topic.
5. If considered as on-topic, particular elements of the page are then sent to the T-Graph to carry out specific comparisons and calculations. The T-Graph data is used to determine the importance of the unvisited links within the page.
6. T-Graph associates a priority score to each unvisited link. Even the off-topic URLs are assigned a lowest value as the priority. This helps the crawler to harvest unlinked on-topic regions on the Web, which are connected through off-topic pages. There is an observing component called the watchdog on the T-Graph. It periodically updates the T-Graph in order to incorporate its experiences.
7. After completing all analyzes, the relevance calculator inserts the URLs and their assigned priority scores to the fetcher queue. The priority score of the fetcher queue items is cyclically incremented to prevent starvation.
8. HTML data of the analyzed Web page is fully stored in the repository along with all the measurements such as the priority scores given to the links.

9. A database checker component constantly runs specific checkups on the repository and updates its indexes with the ultimate goal of keeping the repository up to date.

Note that depending on software engineering object oriented concepts and Web application operational constraints [22], our system framework and architecture respects all the standards both related to modular programming and W3C regulations.

5. Practical Analysis

There are two significant tasks for a focused Web crawler; detecting the topical focus of an unvisited page and assigning an appropriate priority score to its URL. This is achieved by using the T-Graph structure and other custom methods described below.

5.1 Detecting the Topical Focus

As the knowledge classification system, we exploit the Dewey decimal system [25], where the human knowledge is divided into super-categories, each of which is divided into 10 categories and further divided into 10 sub-categories. After the third digit, there is a decimal point and the numbering continues to take a topic into more detail and specificity. For our prototype, we only focused on the first 3 digits and not the decimal part. However, the system is implemented in such a way that any length of the codes can be considered, provided that the complete version of the Dewey system is available. We call each code of the Dewey system a D-number.

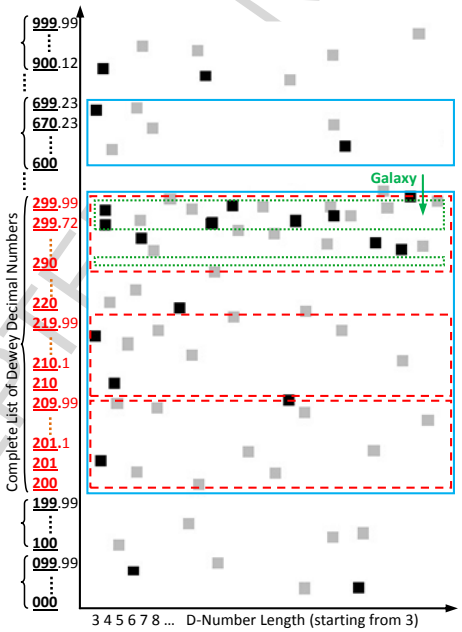


Figure 4.Example of processing the D-Numbers and retrieving the galaxy

Based on the words of each link (unvisited) in the current Web page, a (conceptual) graph is sketched with the list of D-Numbers in one dimension and D-Numbers' lengths in the other dimension. Before any textual comparison, the text is stemmed using the Porter stemming algorithm [26]. This way, the textual data is normalized and the errors are minimized.

In order to segregate the textual data of the Web page into words/phrases, an HTML document parser is used. It gives phrases if any phrase is directly present in DDC, otherwise it gives the words. It is possible for a word/phrase to have more than one D-Number. The word "clothing" is an example which belongs to several disciplines. It belongs in 155.95 for its psychological meaning; customs associated with clothing belong in 391 as part of the discipline of customs; and clothing in the sense of fashion design belongs in 746.92 as part of the discipline of the arts. Therefore, for each word/phrase, more than one point may be plotted. The plotting is stopped when the paragraph boundary is reached (called break points, which initially could constitute the starting and ending of the paragraph itself). If the unvisited link is a list item, the plotting is stopped after plotting all the list items. Then the plotted points are analyzed. [7]

An example of the discussed conceptual graph is shown in Figure 4 with the following explanation:

- The dots represent the words located between the break points.

- Each black dot corresponds to the words of the anchor text (of the unvisited link). Gray dots represent the words out of the anchor text.
- The target is to find the area with the highest population of dots called the Galaxy, which will show the topical focus of the text.

Although there are many methods in the Data Mining subject field to eliminate the outliers from a sample data set [10], in our system a simpler method is used to detect the Galaxy, while taking several factors into account. This algorithm takes three characteristics of the words existing in the particular Galaxy. First is the number of the words which determines the thickness of the population. Second is the D-Number length which explains the preciseness of the topic. Third is called the anchor impact, where black colored dots have a higher impact since they belong to the words in the anchor text. These words have an effect of about 40% more than the text around the unvisited link [27].

To graphically demonstrate the process, rectangular regions are shown in Figure 4. We define *W* as the weight of each rectangle. The blue (lined) rectangles show the regions where D-Numbers have the same first digit. For all of these regions (ten regions), W is calculated and the rectangle with the maximum value of W is considered. For example, the rectangle for D-Numbers starting with 2 is selected which is related to the super topic "religions" as in DDC. It is also possible to consider more than one region with the highest W. Then only the dots (D-Numbers) inside the selected blue rectangle are considered. This time, the rectangular regions (red dashed rectangles) are selected based on the second digit (ten regions) and again for all of them W is calculated. The one(s) with the highest value of W are considered. In this example, the selected region constitutes D-Numbers with 2 as the first and 9 as the second digit, which is related to "other religions" as in DDC.

Again, the dots in the selected red rectangle(s) are divided into ten smaller rectangles, according to the third digit of their D-Number (green dotted rectangles) and W is calculated for each of them. Since in this example the process has to continue until three digits, the green rectangle(s) with the maximum value of W is (are) considered as the Galaxy and show the topical focus of the text, which in this example is "Religions not provided for elsewhere" according to DDC. Parameter W is calculated as follows:

$$W = n \times \sum_{i=1}^{n} \{length(d_i) \times anchor\_impact(d_i)\}$$

where:

n = Number of dots in a rectangular region
length (d_i) =D-Number length
d_i = D-Number
anchor_impact (d_i) = $\begin{cases} 1.4 & \text{if } (anchor\_text = true) \\ 1.0 & \text{if } (anchor\_text = false) \end{cases}$

This method of calculating the weight of regions is significant in that the number of nodes, their D-Number length (preciseness according to DDC) and their being a part of the anchor text are simultaneously taken into account. Note that substituting the outlier detection algorithm with this method will decrease the overall calculation load of the system since practically this process is performed by a simple string filtering.

If the predicted topical focus of an unvisited link complies with the specified topic of the crawler, the priority of the link should be calculated by utilizing the T-Graph. Otherwise, if the predicted topic does not match any of the topics assigned to the crawler, the link is given the least priority (0.01 by default). This is in order to force the crawler to gradually connect to and traverse other areas on the Web where there is a thick population of relevant pages.

5.2 Structure and Construction of T-Graph

Firstly introduced by Patel [7], the T-Graph can be described as a highly flexible tree structure in the context of graph theory. The basic idea is very much similar to that of directed graphs in that each node can have more than one parent as well as more than one child. However, their significant difference is that the nodes of T-Graph are placed in several distinct levels, as in a tree. T-Graph is also distinguished for the connection type between its nodes; each node in a specific level of the tree can directly point to the nodes of any levels below. It must be stated that in the Treasure-Crawler, the tree-like structure of the T-Graph is constructed based on the data set of interlinked Web pages that is provided by the system admin. This will help the graph to be initially constructed and made ready for the next usage. However, another possible approach is similar to that of context focused crawler [12], where the target documents are handed over to the graph and each graph automatically builds itself in a bottom-up fashion. In the initialization phase of this approach, another search engine is used to fetch the pages that point to the current document (its parents). This process continues until a desired number of

levels are formed. In addition, the nodes are flexible and programmable to contain different forms of data or data structures. As shown in Figure 5, in our proposed Treasure-Crawler system, each node of the T-Graph contains a structure of five parts that contain the following HTML attributes of a Web page:

- Immediate Sub-section Heading component (ISH).
- Heading of the Section which contains ISH (SH).
- Main Heading component (MH).
- Data Component (DC) contains the text around the link.
- Destination Information Component (DIC).

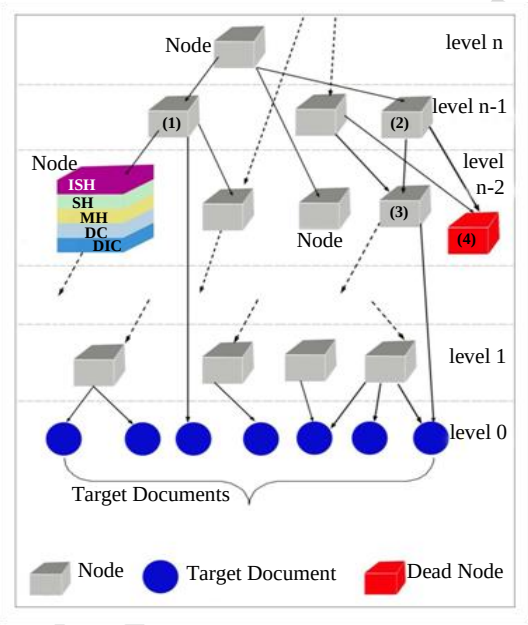


Figure 5.T-Graph hierarchical structure

Practically, for each paragraph containing a hyperlink in a Web page one node is made in the T-Graph. When a paragraph contains two or more links, or a list of hyperlinks is present, one node will be made corresponding to all of them. This is the reason for a node having more than one child. Thus, for one Web page more than one node may be made in the graph. Figure 6 shows the real T-Graph construction from example Web pages. Note that it is assumed that the links to other pages from each page are not in one common paragraph or list.

The central assumption is that the importance of any un-visited URL entirely depends on the content of the topical boundary that encloses it. It is also assumed that an existing URL in the DC component usually points to a document containing some URLs, each representing a node in the graph. For information about the *dead nodes* and the *watchdog component* please refer to [7].

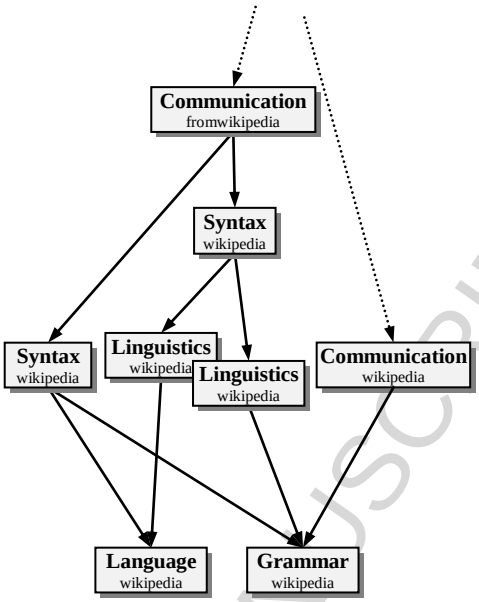


Figure 6.T-Graph real structure from example Web pages

Therefore, the graph is constructed in two stages; in the first run the graph is made out of the pages and their interlink structure and then in the second stage the link distance of each page to reach the target level is calculated and stored in its respective component of the node. In Figures 5 and 6 respectively, the *two lines* at the top do not come from a common *root* node because they can go up to any unknown levels or nodes in a search and retrieve cycle.

5.3 Calculating the Priority

As the Web page is downloaded, its unvisited hyperlinks are extracted and their topical boundaries are processed to detect the topical focus of the destination Web page in terms of D-Numbers. In order to assign a priority score to the target document, the corresponding HTML components of the Web page and those of *all* T-Graph nodes are compared according to the concepts and algorithms of text similarity. Each comparison gives a similarity score with the following details and all textual data is stemmed before being processed or compared.

To retrieve certain HTML elements of the document, its tag tree and source code are analyzed by the HTML parser. The hierarchical structure of the tag tree helps find the context of a URL (located inside the anchor tag). This context is referred to as the topical boundary. After the HTML components of the Web page are extracted, they are employed for the following processes:

The ISH data of each T-Graph node is compared with the subheading's (say U) words containing the unvisited link, to obtain a similarity measure say sim_{ISH}. The subheadings' words (if any) containing the U are compared with SH data to obtain a similarity measure of sim_{SH}. The main heading of the document containing the unvisited link is compared with MH component to obtain a similarity measure of sim_{MH}. The words found within the topical boundary of the unvisited link are compared with the words present in DC component to obtain a similarity measure of sim_{DC}. The DIC component contains the information such as the least number of documents that are to be downloaded to get the target documents (link distance). The sim_x is the cosine similarity of two term frequency vectors.

Cosine similarity also normalizes the documents' length during the comparison, and is defined as the ratio of dot product to magnitude product of V_x and V_y, calculated as:

$$Sim_x(V_x, V_y) = \frac{V_x \bullet V_y}{\|V_x\| \|V_y\|}$$

where:

V_x is the term frequency vector. Subscript x assumes ISH, SH, MH and DC.
V_y is the term frequency vector of the words, where y assumes either the words of the subheading containing the unvisited URL to compare with ISH, each of the subheadings that contain the subheading containing the URL to compare with SH, the main heading to compare with MH, and the words in the topic boundary of an unvisited URL to compare with DC.

$V_x \bullet V_y$ is the dot product of the two explained term frequency vectors defined as:

$$V_x \bullet V_y = \sum_{i=1}^{n} V_{x_i} \times V_{y_i}$$

$||V_x|| \times ||V_y||$ is the product of magnitudes of vectors V_x and V_y defined as:

$$\|V_x\| \times \|V_y\| = \sqrt{\sum_{i=1}^{n} (V_{x_i})^2} \times \sqrt{\sum_{i=1}^{n} (V_{y_i})^2}$$

Therefore the cosine similarity is calculable as:

$$Sim_x(V_x, V_y) = \frac{V_x \bullet V_y}{\|V_x\| \|V_y\|} = \frac{\sum_{i=1}^{n} V_{x_i} \times V_{y_i}}{\sqrt{\sum_{i=1}^{n} (V_{x_i})^2} \times \sqrt{\sum_{i=1}^{n} (V_{y_i})^2}}$$

Note that the cosine similarity measure of the two compared documents ranges from 0 to 1 based on the fact that the term frequencies cannot be negative. Thus, the *OSM (Overall Similarity Measure)* is calculated as a function of sim_{ISH}, sim_{SH}, sim_{MH}, sim_{DC} for each node and the words present within the topical boundary of an unvisited URL. The *OSM* of the unvisited URL is calculated as:

$$OSM = average\ (sim_{ISH}, sim_{SH}, sim_{MH}, sim_{DC})$$

The nodes whose *OSM* is above a threshold value are considered. In our experiment, the threshold is considered as 0.05 by default which means the similarity should be at least 5% so that a node will be considered as similar. Once the matching nodes are picked, their DIC data are checked to calculate the (least) number of documents to download (link distance). Based on that value, the priority score of the URL is calculated as:

$$Priority\ score = \frac{1}{min\ \{link\ distance\}}$$

In the case that none of the considered node's *OSM* value exceeds the minimum threshold, the URL is prioritized with some least value, but visited. The priority of such links is assigned to a fixed constant value, which is the inverse value of number of T-Graph levels incremented by 1, as studied by Diligenti et al. [12]:

$$Priority\ score = \frac{1}{Graph\ levels + 1}$$

6. System Test and Validation

Treasure-Crawler is implemented as a prototype and its performance is consequently analyzed, as well as compared to the performance of some other comparable systems, to observe its characteristics, particularly in terms of *accuracy* in the detection of *topical focus* of retrieved pages. Our prototype tests additionally produce alternative values for several parameters of the system that can be applied to its later version(s). Typically, the results show the values of precision and recall both as ~0.5. The results assert that the proposed algorithm outperforms the Context Focused Crawler in terms of *accuracy* with the improved number of retrieved on-topic pages by 14% for crawls with generic seed URLs and 22% for crawls with on-topic seed URLs. Extensive details on the implementation and evaluation results of the Treasure-Crawler are elaborated in a follow-up paper titled: "Analysis and Evaluation of the Link and Content Based Focused Treasure-Crawler" [8] while the gist of the significant result is presented herein to demonstrate its effectiveness.

To target the design optimality, the framework and architecture of the proposed system has been verified against the inefficiencies of the current Web crawlers, and all the functional and non-functional requirements stated in Section 3. Pilot operational runs along with the results analysis were implemented and tested on a fully dedicated machine with a permanent high speed Internet connection. The network, hardware and software conditions of such experimental prototype limited our evaluations to focus specifically on the Web and information retrieval common concepts; hence, we only focused on those criteria that represent the accuracy of the retrieval of relevant pages. The prototype is implemented to satisfy those criteria that relate to its software engineering principles such as scalability, modularity and robustness.

To assure the practicality of the system architecture, we built the T-Graph from an initial set of example pages and input its data into a database. One generic seed URL was supplied to the system and the parameters were initiated. This run was designated to observe the system behavior in different conditions and identify any possible bugs. The modularity and activity discretion of the modules were closely watched and verified. In some

cases, we changed the initial parameters with alternate values to compare the change in outputs. The behavior of the T-Graph was also carefully monitored to test its accuracy. Finally, this pilot test provided fruitful results and prepared the system for the major large-scale operational test runs, with the extensive and detailed results reported in our forthcoming related paper [8]. It is safe to say that we have tested, verified and validated the proposed architecture in terms of the stated requirements as well as functionality and practicality of the Treasure-Crawler system. A summary of the results follows:

- To enhance the accuracy of the topic prediction of an unvisited page, it is proved that in addition to the anchor text, taking into account some specific HTML elements of the current page is determinant.
- To decrease the processing overhead caused by the textual similarity calculations and analysis, our proposed custom method can effectively replace many statistical methods.
- To improve the quality of priority assignment to unvisited URLs, the use of T-Graph arms the system with a flexible and reliable guide map, which leads to a better determination of the links to follow.

To present the overall significant gist of the results, Figure 7 shows the performance comparison of the proposed system against that of context focused crawler. The ratio of relevant-to-all crawled pages was gained by multiplication of the number of retrieved pages by the calculated recall value (0.5) and as observed, our system outperforms the context focused crawler when initialized with both generic (G) and on-topic (T) URLS. Moreover, it should be stated that our proposed architecture is novel in both the custom technique to predict the topical focus of the unvisited link, and the custom approach to build and employ the T-Graph for URL prioritization.

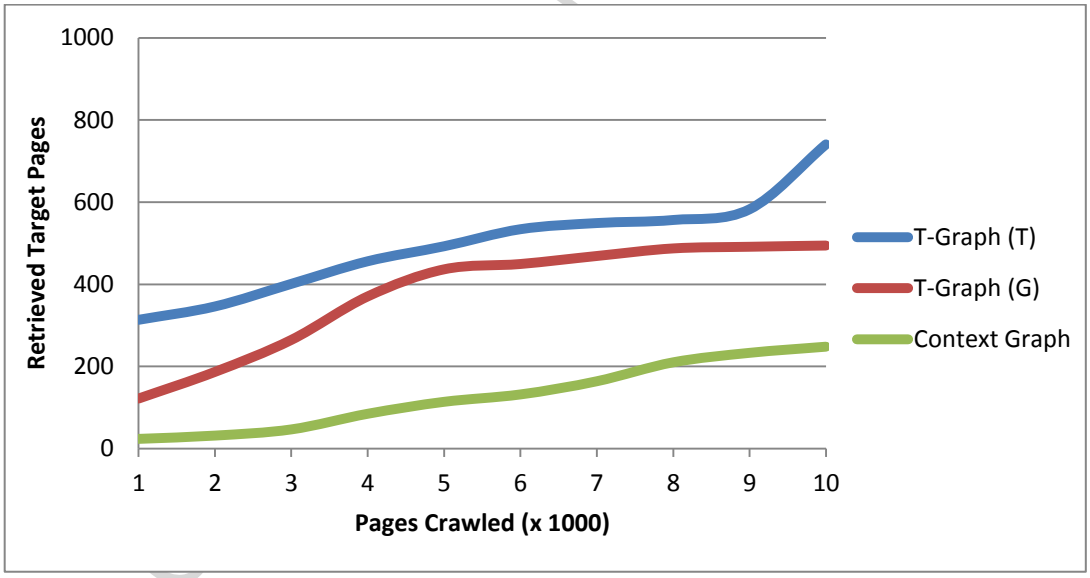


Figure 7. Retrieved documents multiplied by recall value (0.5)

7. Discussion and Conclusion

In this paper, we presented the architectural design of the Treasure-Crawler system. This architecture satisfies the requirements of a focused Web crawler and asserts the correctness of the system structure as well as all its modules. Based on the proposed design and the modularity of the system components, the Treasure-Crawler can easily be implemented with the waterfall model of software development process. However, the modules should first be made with respect to the object oriented or component based software development concepts. They are then weaved using specific interface to make the complete system. Throughout this process, all the necessary standards should be respected since they will affect the efficiency of the results.

In order to satisfy the objectives of this research, the architectural design of our proposed Web search system is set to comply with all the requirements of a focused Web crawler. As a result, the modules of the proposed architecture all act towards the satisfaction of two significant goals. In order to best predict the topic focus of an unvisited Web page, a custom algorithm and set of calculations are utilized based on the Dewey Decimal system.

Since the unvisited URL needs to be prioritized – to make the crawler harvest as many on-topic pages as possible – the T-Graph hierarchical structure is used as a guide to associate each unvisited Web page to a level of closeness to the sample target documents. This way, the URLs receive a score which indicates how related they are to the main topic of the crawler.

The proposed architecture is simple while thoroughly covering all its requirements. The system modules can simply be implemented based on either object oriented techniques, component based software development, or even aspect oriented techniques since they all share the support for modularity [28]. According to these facts, the outcome of the proposed Treasure-Crawler architecture asserts the effectiveness of such design and therefore the system prototype is ready to be implemented, offering significant performance advantages as elaborated in a follow-up paper titled: "Analysis and Evaluation of the Link and Content Based Focused Treasure-Crawler" [8].

Finally, many amendments can be applied to the proposed design. The text classification and similarity formulas and metrics can be diversely designed and decided. HTML and XML elements can affect the importance of a Web page differently. Recently, some serious attempts have been made on designing intelligent techniques especially depending on the semantic Web concepts such as RDF and ontologies. One of these approaches not only considers the content similarity at definition level, but performs a context-aware similarity assessment by looking at and connecting the dispersed pieces of information in an XML structure hierarchy [29]. Another issue known to affect the classification process of a Web page is called "noise" and is caused by using specific design templates and results in very similar HTML DOM trees. The noise detection/reduction technique introduced in [30] can effectively be embedded into the design of the crawler (if it relies on HTML elements of the page) and results in a higher classification accuracy.

In conclusion, the subject field of focused Web crawlers is very interesting both theoretically and practically. To-date, there are many Web search systems designed and implemented, all targeting to overcome one major shortcoming: *gathering and freshly maintaining an up-to-date index of the whole Web as efficiently and effectively as possible*.

References

[1] M. d. Kunder. (30 June 2015). *The size of the World Wide Web*. Available: http://www.worldwidewebsize.com

[2] M. de Kunder, "Geschatte grootte van het geïndexeerde World Wide Web," *Tilburg University*, vol. 63, 2008.

[3] Google. (30 June 2015). *Google Inside Search*. Available: http://www.google.com/search/about/insidesearch/howsearchworks/thestory

[4] M. Jamali, H. Sayyadi, B. B. Hariri, and H. Abolhassani, "A Method for Focused Crawling Using Combination of Link Structure and Content Similarity," in *IEEE/WIC/ACM International Conference on Web Intelligence*, 2006.

[5] W. Wang, X. Chen, Y. Zou, H. Wang, and Z. Dai, "A Focused Crawler Based on Naive Bayes Classifier," in *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, 2010, pp. 517-521.

[6] D. Taylan, M. Poyraz, S. Akyokus, and M. C. Ganiz, "Intelligent Focused Crawler: Learning which links to crawl," in *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*, 2011, pp. 504-508.

[7] A. Patel, "An Adaptive Updating Topic Specific Web Search System Using T-Graph," *Journal of Computer Science*, vol. 6, pp. 450-456, 2010.

[8] A. Seyfi and A. Patel, "Analysis and Evaluation of the Link and Content Based Focused Treasure-Crawler," *Computer Standards & Interfaces (under review)*, 2015.

[9] F. Menczer, "ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery," in *14th International Conference on Machine Learning (ICML97)*, 1997.

[10] S. Chakrabarti, M. v. d. Berg, and B. Domc, "Focused Crawling: A New Approach to Topic-Specific Web," in *The 8th World-wide web conference (WWW8)*, 1999.

[11] G. Pant, K. Tsioutsiouliklis, J. Johnson, and C. L. Giles, "Panorama: Extending Digital Libraries with Topical Crawlers," in *2004 Joint ACM/IEEE Conference on Digital Libraries*, 2004, pp. 142 - 150

[12] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori, "Focused Crawling Using Context Graphs," in *26th International Conference on Very Large Databases, VLDB*, Cairo, Egypt, 2000, pp. 527-534.

[13] M. Ester, M. Groß, and H.-P. Kriegel, "Focused Web Crawling: A Generic Framework for Specifying the User Interest and for Adaptive Crawling Strategies," in *27th International Conference on Very Large Database, VLDB*, Roma, Italy, 2001.

[14] J. Qin, Y. Zhou, and M. Chau, "Building Domain-Specific Web Collections for Scientific Digital Libraries: A Meta-Search Enhanced Focused Crawling Method," in *Joint ACM/IEEE Conference on Digital Libraries*, Tucson, Arizona, USA, 2004.

[15] M. Yuvarani, N. C. S. N. Iyengar, and A. Kannan, "LSCrawler: A Framework for an Enhanced Focused Web Crawler Based on Link Semantics," in *IEEE/WIC/ACM International Conference on Web Intelligence*, 2006.
[16] C.-C. Hsu and F. Wu, "Topic-specific crawling on the Web with the measurements of the relevancy context graph," *Information Systems*, vol. 31, pp. 232-246, 2006.
[17] C. Xiaoyun and Z. Xin, "HAWK: A Focused Crawler with Content and Link Analysis," in *e-Business Engineering, 2008. ICEBE '08. IEEE International Conference on*, 2008, pp. 677-680.
[18] O. Jalilian and H. Khotanlou, "A New Fuzzy-Based Method to Weigh the Related Concepts in Semantic Focused Web Crawlers," in *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, 2011, pp. 23-27.
[19] F. Ahmadi-Abkenari and A. Selamat, "An Architecture for a Focused Trend Parallel Web Crawler with the Application of Clickstream Analysis," *Information Sciences*, vol. 184, pp. 266-281, 2012.
[20] F. Buccafurri, G. Lax, A. Nocera, and D. Ursino, "Moving from Social Networks to Social Internetworking Scenarios: The Crawling Perspective," *Information Sciences*, vol. 256, pp. 126-137, 2014.
[21] M. P. Basgalupp, R. C. Barros, A. C. P. L. F. de-Carvalho, and A. A. Freitas, "Evolving Decision Trees with Beam Search-Based Initialization and Lexicographic Multi-Objective Evaluation," *Information Sciences*, vol. 258, pp. 160-181, 2014.
[22] I. Sommerville, "Software Engineering," 10 ed, 2015.
[23] M. Koster. (1994, 30 June 2015). *A Standard for Robot Exclusion*. Available: http://www.robotstxt.org/orig.html
[24] DMOZ. (1998, 30 June 2015). *Open Directory Project*. Available: www.dmoz.org
[25] OCLC. (30 June 2015). *Dewey Services at a glance*. Available: http://www.oclc.org/dewey/
[26] M. F. Porter, "An Algorithm for Suffix Stripping," *Program: electronic library and information systems*, vol. 14, pp. 130-137, 1980.
[27] A. Passerini, P. Frasconi, and G. Soda, "Evaluation Methods for Focused Crawling," in *AI*IA 2001: Advances in Artificial Intelligence*. vol. 2175, F. Esposito, Ed., ed: Springer Berlin / Heidelberg, 2001, pp. 33-39.
[28] A. Patel, A. Seyfi, M. Taghavi, C. Wills, L. Na, R. Latih, *et al.*, "A Comparative Study and Analysis of Agile, Component-Based, Aspect-Oriented and Mashup Software Development Methods," *Technical Gazette*, vol. 19, pp. 175-189, 2012.
[29] P. D. Hossein-Zadeh and M. Z. Reformat, "Assessment of Semantic Similarity of Concepts Defined in Ontology," *Information Sciences*, vol. 250, pp. 21-39, 2013.
[30] D. Alassi and R. Alhajj, "Effectiveness of Template Detection on Noise Reduction and Websites Summarization," *Information Sciences*, vol. 219, pp. 41-72, 2013.

Research Highlights:

- We present the architectural design of a focused Web crawler that combines link-based and content-based approaches to predict the topical focus of an unvisited page.
- We present a custom method using Dewey Decimal Classification system to best classify the subject of an unvisited page into standard human knowledge categories.
- To prioritize an unvisited URL, we use a dynamic, flexible and updating hierarchical data structure called T-Graph. It helps find the shortest path to get to on-topic pages on the Web.
- For the background review, the experimental results from several crawlers are presented.
- The functional and non-functional requirements of a focused Web crawler are elicited and described, as well as standard evaluation criteria.