Focused Crawling Using Content Classification and Link Priority Estimation

Shwetanshu Rohatgi, Sabarni Kundu

Abstract— Focused crawlers are used to crawl and index web pages that are specific to a given topic but due to this sheer amount of web pages and data generally, a large part of data gets ignored and various anchor tags are never indexed. In order to solve such problems, we propose an improved crawling technique by dividing current challenges into two modules and conquering them individually. We introduce a new weighing factor based on content blocks and mutual information to obtain relevant web pages. Further, we propose the use of Context graphs and content block partition technique in order to find relevant web links by using link priority calculator (LPC) based on cosine similarity. This paper illustrates experimentally that our focused crawler is better than other focused crawlers based on breadth-first, anchor text only and link-context only partition in terms of target recall. In conclusion, our proposed system is effective and efficient for focused crawling.

Index Terms— Sematic Web, Focused Crawler, Crawling Algorithms, Naïve Bayes, Context Graphs, Link Priority, Cosine Similarity.

1 Introduction

WWW is world wide web which is a collection of millions of web pages which act as a source of information. The information is classified into various categories like text, audio- visual and multimedia formats.

A Web Crawler is an automated system that has the capabilies to traverse the web graph and parsing various pages as well as forming a local repository of the URLs that have been visited by user. Crawling involves interaction with millions of web pages and thousands of web servers. The speed of web crawler is not only governed by the speed of once personal internet connection but also depends on the speed of various websites to be crawled. It is a very internsic application for gathering and preserving data and updating the information with the ever expanding internet. It acts as a tool to index, classify and update databases across servers.

A web crawler fetches a set of web pages and store them into relevant database which is further used for indexing. Pages once downloaded are then queued based on selection and revisit policies. Crawler has to revisit the pages to refresh the URL database. Seed URLs are needed to begin the crawling process. Links on seed URLs are extracted and tread recursively. Crawl frontier queue contains the hyperlink to be visited and Crawled frontier contains hyperlinks already visited.

This paper has been organized into five sections. Section – 2 highlights the basic architecture and working of Web crawler. Section -3 will pay emphasis on the related works based on different types of web crawler. Section – 4 will be about analysis of various algorithms related to focused crawler, Section -5 will elaborate on challenges faced by current focus crawler, Section -6 will present our proposed web crawling algorithm further Section -7 will infer our research findings experimentally, finally Conclusion and References will conclude our last two sections.

2 Working of Web Crawlers

2.1 Introduction

The main purpose of web crawler [1] is to fetch URL and download the corresponsding pages mention in the webpage. Web crawlers are essential part of search engine where they amass the corpus of webpages queued by the engine itself.

Initially web crawler starts its system by setting of URL request. All the important URLs that are to be retrevied and prioritiesed are kept in URL queue and from here the crawler gets a URL link and download the corresponding webpage. After page downloading URLs are passed to the extractor which would extract the required data given by the users and then data can be organized into groups and further URL can be pushed back to queue. This process is repeated over and over again till the URL queue is empty [1].

2.2 Architecture

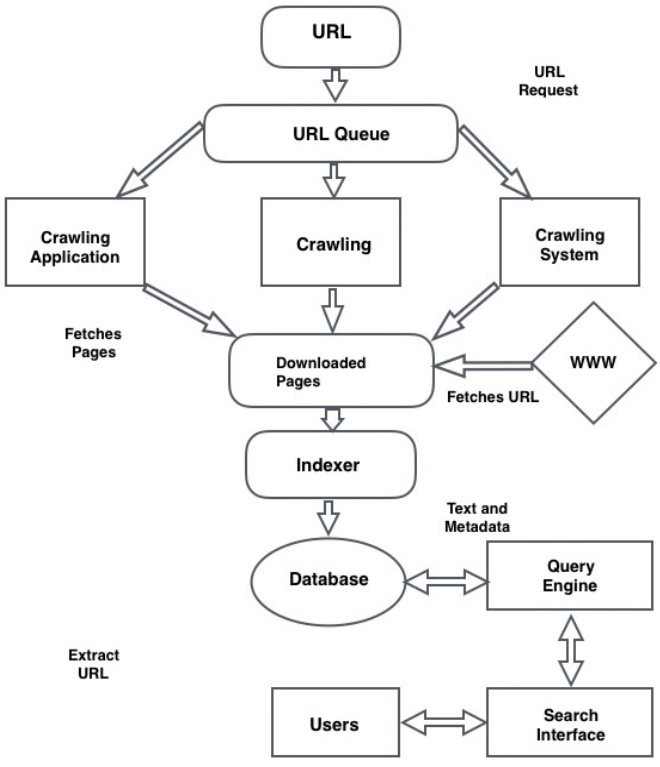

Fig. 1. Web Crawler Architecture.

2.3 Functioning of Web Crawler:

A list of URL seeds are passed on to URL queue via URL request. A set of crawler then gets a subset of URLs to crawl depending on the return subset. They are categorized into different domains. The crawler then fetches the web pages with the help of a downloader, which is further passed on to the extractor. This extractor extracts relevant information according to the users query.

3 TYPES OF WEB CRAWLERS

Depending on how the web pages are crawled and how successive pages are retrived, we can categories web crawlers into following types:

3.1 Focused Web Crawler

Focused crawler [2] selectes the relevant topics and obliterate the irrelevant one from the repository on the basis of relevance calculator algorithms.

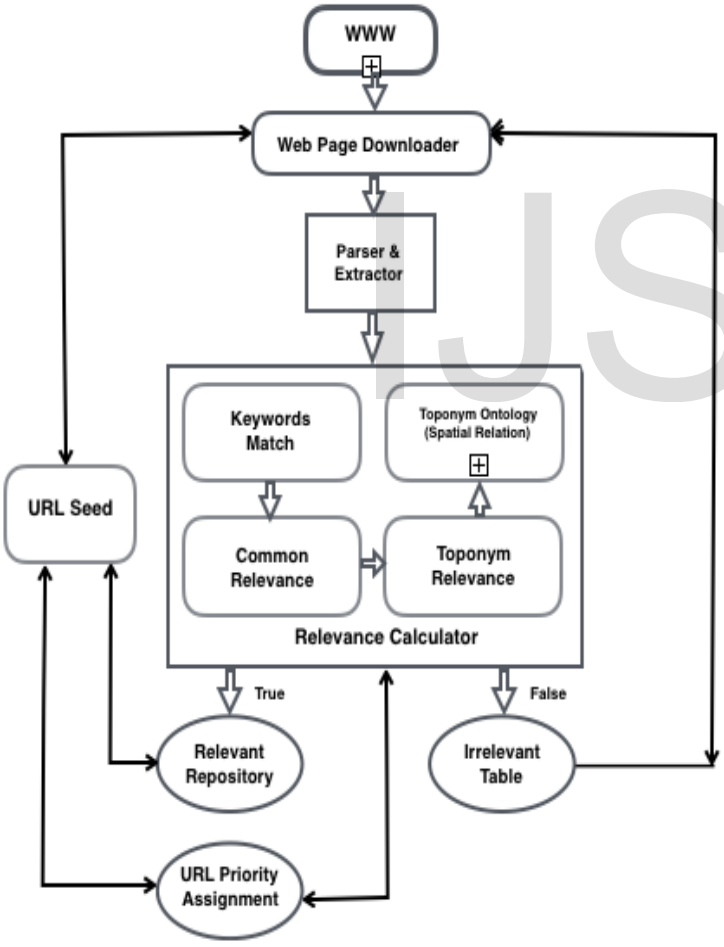

Fig. 2. Focused Web Crawler.

The master crawler downloads the URL with the correspondding web pages from the internet. It then passes the downloaded pages to the extractor which extracts the content and passes on to the relevance calculator. In relevance calculator, the content is judged according to the keyword matched, common relevance, toponym relevance and toponym ontology. On the basis of this the web pages are categorized into relevant and irrelevant data. Now, the relevant data are given priority number according to their relevance given by the user. Further, these webpages with assigned priority numbers are passed on to the URL seed and the process continues till queue is empty.

3.2 Distributed Web Crawler

A Distributed crawler [3] allows various spiders to crawl through a number of web pages simultaneously. The whole crawling task is divided into various spiders so as to improve efficiency and fasten the crawling process. Distributed crawler Obiviates the duplicate content which as repeating URLs. In this the master crawler fetches and downloads numerous number of webpages at the same time and it then pass the content to extractor & parser that extract out the content. Further, this content is passed on to the decision block that checks for the content repetition. If it returns true, it passes the content to duplicate table and further on to irrelevant table. If the decision block returns false it passes the content into repository which further passes the content to another decision block that checks for duplicate key error if the decision is false it pass data to unique table and further to relevant repository.

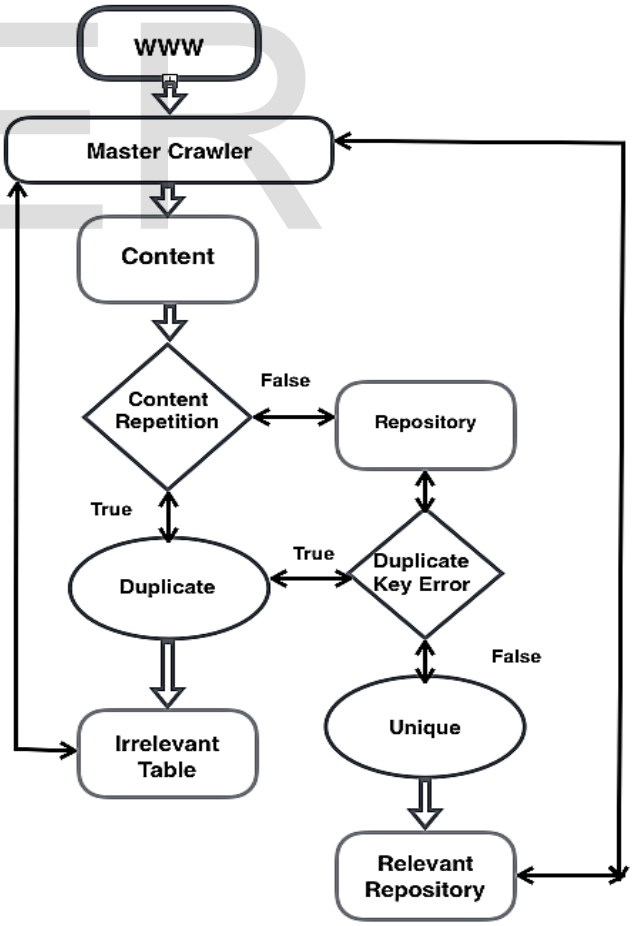

Fig. 3. Distributed Web Crawler.

3.3 Parallel Web Crawler

In parallel crawler the downloading rate is generally high as the search engines run with multiple process in parallel to download the web pages.[3]This process depends on the freshness of page and accurate selection of page [4].The given figure is the basic architecture of Parallel Crawling. Parallel Crawler includes the processes of multiple crawling, that is generally referred as C-proc's. These C-procs functions like the basic crawler only. Initially it downloads webpages from internet and store in repository and extract the relevant content. C-procs then splits the downloading task and according to the splitting the extracted links may be distributed among the other crawlers.These C-procs then allocate the task either on local network or at geographical distant location.[5]

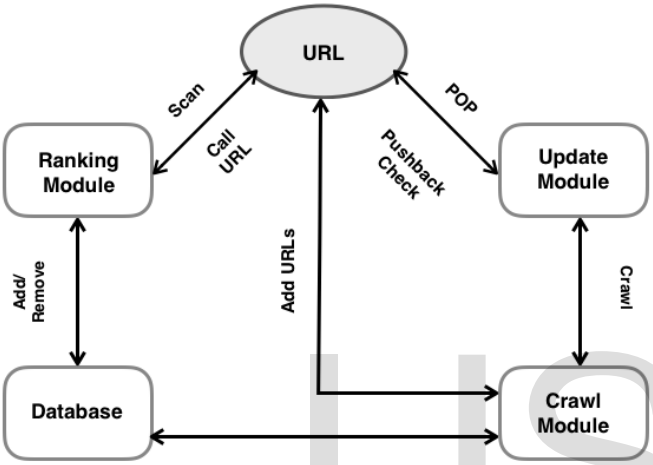

Fig. 4. Parallel Web Crawler.

4 Web Crawling Algorithms

4.1 Breadth First Search

This is one of the oldest web crawling algorithms which is being used since 1994. It uses unvisited URL queue as FIFO queue, crawling links in the order in which they are encountered. A crawler starts at the root node and traverses all the adjacent neighbouring nodes that are on the same level. If the required document is reached then it is a success else it proceeds down the next level in the web tree. In this way this algorithm keeps searching till required node is not searched. If objective is met then it's a success else it's a failure.

Breadth first search is a good technique only if the relevant document node is found in upper portions of the tree but if not then this algorithm will have a very high time complexity as it will have to search through each level and then only it can go deeper into web tree.

Andy yoo et al [6] proposed a distributed BFS for large number of branches using Poisson random graphs and achieved high scalability through a set of clever memory and communication optimizations.

4.2 Depth First Search

This is a novel web graph traversing technique where algorithm starts from seed URL or root node and traverse deeper through a particular child node. This technique will prefer left child and after traversing to the deepest node of left child it backtracks to the next unvisited node and then continues the processes till it finds the document node.

The algorithm is better than Breadth first search but it can get trapped in an infinite loop if there are large number of nodes [7], which is the case with today's web graph.

4.3 Genetic Algorithm

Genetic algorithm is a biological evolution based algorithm. Here a crawler starts with a set of seed urls and a fitness function is applied on these seed urls that determine the selection of the fittest offsprings. These offsprings undergo different genetic operations like – mutation, crossovers etc. These operations help to determine the best documents out of the complete set of document and reduces the risk of being stuck in the local minima.

There are various algorithms but genetic algorithms can outperform them as less time is spend in searching a large database. Genetic Algorithms can also handle multimedia requests. Genetic algorithms can target a complete pool of URL together unlike other search algorithms where there is only one root node and traversing takes place node by node. Genetic algorithms are robust and hence there have been various contributions to genetic algorithms [8].

4.4 Naïve Bayes Classification Algorithm

Naïve Bayes Classification is a Machine Learning based classification technique which is based on Baysian Probability Theorem. It is based on 3 units within web crawler. First, is a Page Analyzer which will download the page and extract information in ordr to decide on particular link to follow. It takes up HTML content and forms a HTML label tree. Second, is a characterstic extraction that makes sense of content of pages with TF-IDF (Term Frequency – Inverse Document Fre quency) [9] which gives weightage to a word in a given link context whereas IDF will reduce weightage of a word if it makes a lot of occurences on the page. Finally a Relevance analysis based on Bayes theorem is used to calculate relevance of the page to that of the topic.

Naïve Bayes classifier can solve trap problem with crawlers where crawler gets trapped if a large number of URLs pointing to the same page. Further reinforcement learning is applied to web crawlers to push its limits [10].

4.5 Page Rank Algorithm

This web crawling algorithm is about ranking a page based on its importance that is calculated from the backlinks and citations given to a web page [11]. A simple page ranking algorithm is formulated as:

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))\,PR(A)$$

PR(A) = Page Rank of a Website, d = damping factor, T1,….Tn = links (1)

Using this formula pages are ranked but to come up with a more balanced page ranking algorithm, Yongbin Qin and Daoyun Xu [12] came up with an approach to use human fac-

tor into page ranking equation and proposed a noval page belief recommendation mechanishm. It takes into account subjective needs of the user and in this way minimizes chance of a topic drift. A new and revised page ranking algorithm was proposed based on similarity measure of vector space model, called SimRank. This algorithm partitions web database into web social networks (WSNs) based on similarity measure.

4.6 HITS Algorithm

Kleinberg's HITS algorithm, a method of link analysis, uses the link structure of a network of webpages to assign relevance and weights to each page. Topics are weighed using this algorithm. Further it was found that certain tree-like web structures can lead the HITS algorithm to malfunctioning of algorithm which produces no insightful results. Further, two modifications were made to the adjacency matrix input to the HITS algorithm. Exponentiated Input, first modification, includes information not only on direct links but also on longer paths between pages. It solved both problems related to HITS algorithm [13]. Usage Weighted Input, in second modification, weights links according to how often they were followed by users in a given interval; it incorporates user feedback without requiring direct user involvement.

5 Challenges faced by focused crawler

Now we will consider a typical focused crawler that crawls only promising links and ignores off topic crawling by associating a score with each link in the page that has been downloaded. For instance if a crawler starts from a document which is n steps from a target document, then it will download only limited set of pages till n-1 steps unlike a typical crawler that will download all the webpages in the web graph till n-1 steps.

A focused crawler will consider link structure of the web graph and content of a webpage [15]. A focused crawler will associate a score based on relevance of the page and then best-first search is applied that pops out the most relevant page in the queue and in this way most relevant pages are crawled first and this ensures that an optimal crawling path has been traversed.

A major problem faced by this focused crawlers is that it is frequently difficult to learn that some sets of off-topic documents or less relevant web pages often lead to highly relevant documents. This ignorance of less relevant pages causes problems in traversing the hierarchical page layouts that commonly occur on the web. Another difficulty faced by existing crawlers is that links on the web are uni-directional, which restricts searching to top-down traversal, a process that we call "forward crawling", so incase we start our search from a leaf node then we will only be able to traverse down the tree until a link to upper leaf nodes is explicitly mentioned in web page.

6 Proposed crawling algorithm

In order to optimize our focused crawler we need to optimize two steps: Classifying the web pages and selecting the URLs are two most important steps of the focused crawler. To come up with more relevant pages we propose an improved system for web page classification and context based link priority evaluation technique.

6.1 Web Page Classification

A typical TF-IDF is used in information retrieval systems to determine the relevance of a page with respect to topic.

We have optimized the TF-IDF technique by applying it to HTML5 documents that are partitioned into four sections: body, anchor text, keywords, and headline. These are assigned weights based on their expression ability for page content. That means, strong expression ability is propotional to high relevance of page to that topic. A new weighing system will improves the convergence of the information retrieval system by optimizing information gain term.

To enable this first we need to prun our feature space as this will make our classifier faster and efficient. Web page classifier embeds the documents into some large feature spaces, and also for one with very large vocabularies. We will use mutual information (MI) to prune the feature space. Correlation of two events can be determined using MI. It can be concluded that higher the MI factor high is the correlation between two events.

$$MI(Tj, Ci) = \log [p(Tj,Ci)/p(Tj)\ p(Ci)] \quad (2)$$

where MI(Tj, Ci) denote the MI between the feature Tj and the class Ci; denote the probability that a document arbitrarily selected from the corpus contains the feature Tj; p(Ci)] denote the probability that a document arbitrarily selected from the corpus belongs to the class Ci; p(Tj,Ci) denote the joint probability that this arbitrarily selected document belongs to the class Ci as well as containing the feature Tj at the same time [16].

After pruning we need to calculate the weight of each term which is done by our optimized version of TF-IDF. We will we giving weights to Tj in the order headline > keyword > anchor tag text > document.

$$TFij = \Sigma\ \alpha i \times TFij\ ,\ 0< i < 4 \quad (3)$$

New step is to calculate weight of the term in the document Di where TFi is the term frequency and IG is the information gain IGj that is received with respect to term Tj.

$$Wij = TFij \times idFj \times IGj\ /\ \sqrt{\Sigma\ (TFij \times idFj)^2} \quad (4)$$

Once weights have be calculated now using Naïve Bayes machine learning algorithm we can build a classifier that will be able to classify our web page as relevant or irrelevant. We need to compute P(Tk|Ci) using N(Tk,Ds) is number of relevant term occurences in document Ds.

6.2 Link Priority Estimator using Context Graphs

Link priority is an important metric that determines which link will lead to a relevant and vice-versa. In order to compute link priority we use Link Priority Calculator (LPC). Various link priority estimators only consider anchor text to determine priority but there are various pages that use "Click Me", "Click to proceed", "Continue", etc that may not be seem rele-

vant at all but may link to a highly relevant page. This can be overcome by breaking our problem into two modules.

We use Content Block Partition (CBP) [17] to first find the relevance of the content block as per different partions of HTML document. If the content block is highly relevant to the topic then we download the content and extract all links which are added to a priority queue of unvisited link.

Further if a link falls in an irrelevant partition then we further apply LPC strategy to compute link relevance by using cosine similarity and team weighing scheme between link feature and topic [18].

$$\text{Sim}(X, Y) = X \cdot Y / |X| \times |Y| \quad (4)$$

Here X is Eigen vector of given topic and Y is Eigen vector of the given content block. We compute Cosine similarity for each of the content block to compare.

This measure does not allow us to enable significant back crawling although we have queued a lot of topic relevant URLs. So in order to further enhance we introduce context graphs to calculate LPC. We back crawl a significant fraction of the whole web using the seed URLs from the queue containing on-topic document and URLs. Using context graph we can discover hierarchies within a web graph.

We will take one URL as seed initially from our queue. This will form layer-0. Now all relevant URL links we extracted using cosine similarity and LPC will be treated as layer–1. There will be no connections within a layer. Further iterating on layer-1 and finding LPE will help us form a layer-2 and so forth we will be able to capture different context graphs, yielding a layered structure called Merged Context Graph.

After forming a Merged Context graph we need to apply our Webpage Classifier as dicussed in section 6.1 and figure out the relevance of each page in context graph. The classifier of layer 0 is used as the most important determinant of whether a document is topically relevant. The discriminant and likelihood functions for the successive layers are used to predict number of steps must be taken from current page before a target is found in a given content block within a page.

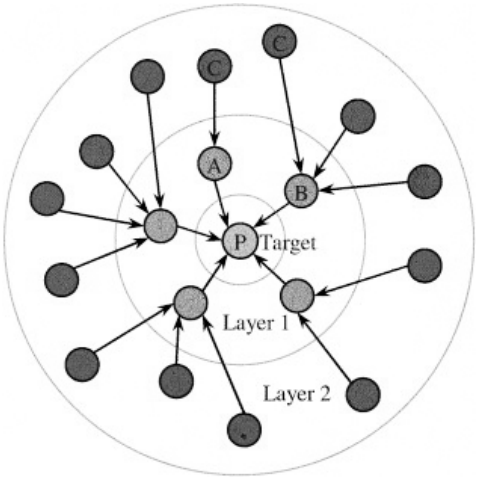

Fig 6. Context Graph Visualization

7 Experimental Results

In order to verify our focused crawler for its efficiency. We have tested it on 20 newsgroups and Open Directory Project corpurs as our training and testing dataset. Out of 135 topics available in corpus we have chosen 9 topics to condense a graphical inference of improved results using our proposed crawler. To test the performance of our crawler, we choose nine topics as samples and 400 samples are chosen from individual topics.

Total recall and Precision play a pivot role in determing performance of our webpage classifier but they have certain accuracy issues as described in [17]. So, we use F-measure proposed by Lewis [19] which can accurately measure performance of our classifier.

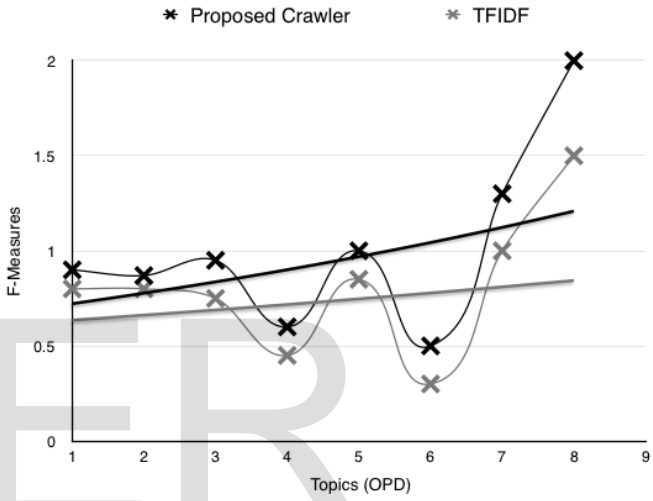

Fig 6. Comparison of F-Measure by Classifiers on dataset

As it can be seen from Fig 6 we have obtained a better F-measure for our proposed web classification technique based on content based weighing of terms. Hence it can be inferred that our classifier is effective in solving classification problems.

Now we look forward to analyzing performance of our Link priority calculator based on context graphs and cosine similarity measures. In this experiment, we selected various revevant pages for 10 different topics like cricket, salsa, robots, machine learning, typhoid etc and the number of those web pages for each topic was set to 35. At the same time, we used the random way of selecting seed URLs corresponding to our topic and the seed URLs for each topic.

However, the relevant URL link set for any above topic is hard to determine in the web, so the true recall is measured inaccurately. Therefore, we adopt target recall to evaluate the performance metrics of our focused crawler.

We use Total recall as proposed by Pant [20] stating that the fraction of relevant pages crawled, which measures how well it is doing at finding all the relevant web pages. Here we create a virtual WWW and give seed URLs and depth of web. Our virtual web consists of web traversed by breath-first crawling strategy where total recall is applied.

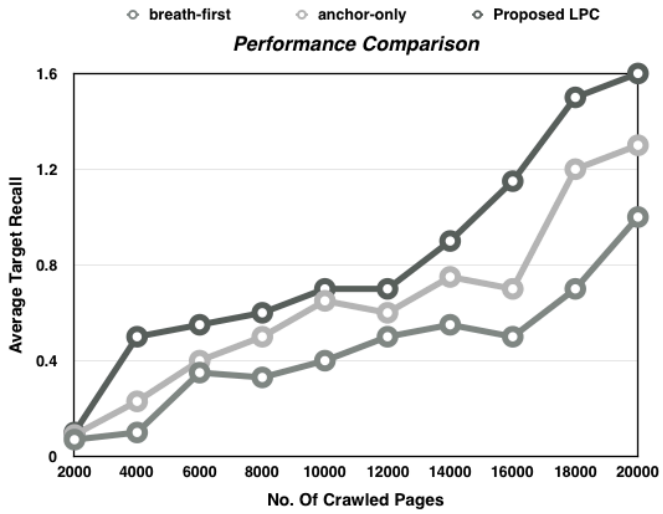

Fig 7. Performance Comparison of Avg. Target Recalls

As can be seen from Fig 7, with the rise of crawling rate of various crawling algorithms, the average target recall of six crawling methods is rising. This happened because the number of crawled web pages is increasing, however target set is untouched. At 14000 pages of crawl we can observe breath-first, anchor-only and proposed LPC giving a total recall of 0.5, 0.75 and 0.9 with ever increasing no of relevant pages crawled. We can infer from the Fig 7 graph that our LPC and context graph based crawler is capable to predict more accurate topical priorities of web links than other crawling algorithms. Therefore, the LPC, aided by Context graph and Cosine similarity strategy, improves the performance of the focused crawlers.

8 Conclusion

In this paper, we proposed and demonstrated a novel focused crawler that builds on top of a content based weighting methodology and classifies relevant links using principle of context graphs and link priority calculator. The approaches proposed and the experimental results draw the following conclusions.

TFIDF is a well know algorithm for relevance calculation but it fails to consider page content relevance and expression ability. Our proposed Webpage classifier based on feature space pruning and content block relevance helped us to improve web classifier. Results show that our classifier outperforms TFIDF for each dataset. In addition, in order to gain better selection of the relevant URLs, we optimised link priority calculation algorithm using cosine similarity and context graphs to prevent ignorance of highly relevant pages. The algorithm was classified into two stages. First, the web pages were partitioned into content blocks by the CBP algorithm and classified using relevance weights. Second, we calculated the relevance between links of blocks using cosine similarity and LPC algorithm then form context graph to enable backward crawl and capture relevant pages.

9 References

[1] A comparative study on web crawling for searching hidden web by IJCSIT
[2] A Study of focused web crawler for sematic web by IJCSIT
[3] Shruti Sharma, A. K. Sharma and J. P. Gupta,"*A noval architecture of Parallel web crawler*", In the Int. Journal of International Journal of computer applications(0975-8887), volume 14, Issue 4, 2011.
[4] AHChungTsol, DanieleForsali, MarcoGori, Markus Hagenbuchner and Franco Scarselli, "*A simple focused crawler*", In the Proceeding of 12th Int. WWW Conf., pages 1,2003.
[5] Junghoo Cho, Hector Garcia-Molina , " parallel crawler"
[6] Andy Yoo,Edmond Chow, Keith Henderson, William McLendon, Bruce Hendrickson, ÄUmit CatalyÄurek "A Scalable Distributed Parallel Breadth-First Search Algorithm on BlueGene/L" ACM 2005
[7] Narasingh Deo "Graph theory with applications to engineering and computer science" PHI, 2004 Pg 301
[8] S.Siva Sathya and Philomina Simon," Review on Applicability of Genetic Algorithm to Web Search" International Journal of Computer Theory and Engineering, Vol. 1, No. 4, October2009
[9] LUO Xin; XIA De-lin; YAN Pu-liu. Improved feature selection method and TF-IDF formula based on word frequency differentia. Computer Applications, 2005, 25(9): 2031-2033.
[10] Wenxian Wang, Xingshu Chen, Yongbin Zou, Haizhou Wang, Zongkun Dai "A Focused Crawler Based on Naive Bayes Classifier" Third International Symposium on Intelligent Information Technology and Security Informatics, 2010
[11] Sergey Brin and Lawrence Page "Anatomy of a Large scale Hypertextual Web Search Engine" Proc. WWW conference 2004
[12] Yongbin Qin and Daoyun Xu "A Balanced Rank Algorithm Based on PageRank and Page Belief recommendation"
[13] Joel C. Miller, Gregory Rae, Fred Schaefer "Modifications of Kleinberg's HITS Algorithm Using Matrix Exponentiation and Web Log Records" Proc. SIGIR'01, ACM 2001.
[14] Apoorv Vikram Singh , Vikas , Achyut Mishra, "A Review of Web Crawler Algorithms", International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014
[15] S. Chakrabarti, M. van der Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," in Proc. of the 8th International World-Wide Web Conference (WWW8), 1999.
[16] Houqing Lu, Donghui Zhan, Lei Zhou, and Dengchao He, "An Improved Focused Crawler: Using Web Page Classification and Link Priority Evaluation," Mathematical Problems in Engineering, vol. 2016, Article ID 6406901, 10 pages, 2016. doi:10.1155/2016/6406901
[17] T. Peng and L. Liu, "Focused crawling enhanced by CBP-SLC," Knowledge-Based Systems, vol. 51, pp. 15–26, 2013.
[18] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Information Processing and Management, vol. 24, no. 5, pp. 513–523, 1988.
[19] D. D. Lewis, "Evaluating and optimizing autonomous text classification systems," in Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '95), pp. 246–254, ACM, Seattle, Wash, USA, July 1995.
[20] G. Pant and P. Srinivasan, "Link contexts in classifier-guided topical crawlers," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 1, pp. 107–122, 2006.