Learning Capable Focused Crawler for Information Technology Domain

Mukesh Kumar
UIET, Panjab University
Chandigarh (INDIA)

Renu Vig
UIET, Panjab University
Chandigarh (INDIA)

ABSTRACT
The Web provides us with a huge and endless resource for information. But, the rapidly growing size of the Web poses great challenge for general purpose crawlers and search engines. It is impossible for any search engine to index the whole Web. Focused crawler collects domain relevant pages from the Web by avoiding the irrelevant portion of the Web. Focused crawler can help the search engine to index all documents present on the Web related to a specific domain which in turn provides the search engine's users complete and up-to-date contents. In this paper we present a focused crawler capable of learning from the previous crawl results to collect the relevant documents. Crawling results for three consecutive learning phases are shown. Results indicate significant improvement in terms of relevancy to the focused domain.

Keywords
Web, Internet, Retrieval, Focused Web Crawler, Search Engine etc.

1. INTRODUCTION
With the ongoing growth of web, finding the right information becomes an increasingly difficult task which often leads to undesired results. This made it important to develop document discovery mechanism. A crawler is a program used by search engine that retrieves Web pages by wandering around the Internet following one link to another. Web search engines such as Goggle, AtlaVista provides access to the Web documents. A search engine's crawler collects Web documents and periodically revisits the pages to update the index of the search engine. Due to the Web's huge size and dynamic nature, Ari Pirkola (2007), no crawler is able to cover the entire Web and to keep up all the changes. This fact has motivated the development of focused crawlers such as Martin Ester et al (2001), Bergmark, Lagoze and Sbityakov(2002), Ehrig, and Maedche (2003) etc. Focused crawlers are designed to download Web documents that are relevant to a predefined domain, and to avoid irrelevant areas of the Web. The benefit of the focused crawling approach is that it is able to find a large proportion of relevant documents on that particular domain and is able to effectively discard irrelevant documents and hence leading to significant savings in both computation and communication resources, and high quality retrieval results.

Related Work
Web crawling was simulated by a group of fish migrating on the Web, Bra and Post (1994),. In the so called fish search, each URL corresponds to a fish whose survivability is dependent on visited page relevance and remote server speed. Page relevance is estimated using a binary classification by using a simple keyword or regular expression match. Only when fish traverse a specified amount of irrelevant pages they die off. The fish consequently migrate in the general direction of relevant pages which are then presented as results. Cho, Molina and Page (1998) proposed calculating the PageRank given by Page et al. (1998) score on the graph induced by pages downloaded so far and then using this score as a priority of URLs extracted from a page. They show some improvement over the standard breadth-first algorithm. The improvement however is not large. This may be due to the fact that the PageRank score is calculated on a very small, non-random subset of the web and also that the PageRank algorithm is too general for use in topic-driven tasks. Ehrig and Meadche (2003) considered an ontology-based algorithm for page relevance computation. After pre-processing, entities (words occurring in the ontology) are extracted from the page and counted. Relevance of the page with regard to user selected entities of interest is then computed by using several measures on ontology graph (e.g. direct match, taxonomic and more complex relationships). Most of the existing focused crawlers (Boldi 2004; Brin and Page 1998; *Chakrabarti*, *Berg*; Cho and Molina 2000, 2002; *Domc* 1999; Page et al 1998) are based on simple keyword matching or some very complex machine learning techniques for guiding the future crawls.

2. PROPOSED CRAWLER
Tf-Idf ***(****Term frequency–Inverse document frequency)* weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus or in turn to the domain. If we are having a corpus of documents which are all highly related with a specific domain then the Tf-Idf score of a term in a document gives the importance of that term for that document with respect to the whole corpus. Now if we add Tf-Idf score obtained by a term for all documents in the corpus, then the resulting score can be seen as a meaningful, semantic, score for that term with respect to the whole corpus. Based upon this thought a TIDS **(**Term frequency–Inverse document frequency Definition Semantic) Score Table is constructed, whose entries are supposed to help the crawler for deciding the future crawls. The TIDS Score Table generation algorithm is given in Algorithm 1. The initial collection of Web pages (Seed pages) is generated from the hierarchical categories of ODP (Open Directory Project) from http://dmoz.org. ODP provides the categorical collection of URLs that are manually edited and not biased by any commercial user. From here we can find individual categories link. The categories ending with "Information", "computers", "internet", "information technology" and "computer science" were retrieved from the ODP, total 86 such categories were found, then links contained by these 86 categories were retrieved from ODP, total 927 such links were found from ODP or we can say that

927 pages related to the Information Technology domain were found, out of these 927 links. All these URLs were put in the Relevant_Page_Set.

Algorithm 1: TIDS Score Table Generation

1. Initialize Relevant_Page_Set.
2. Remove Stop Words from each page in the Relevant_Page_Set
3. Apply Stemmer to each page in the Relevant_Page_Set
4. Generate Tf-Idf Score Inverted Index Table for all the documents in the Relevant_Page_Set.
5. For each term t in the Tf-Idf Score Inverted Index Table Do
   - 5.1. Calculate sum of the Tf-Idf score obtained by t in all documents from Tf-Idf Score Inverted Index Table, let it be TIDS_Score.
   - 5.2. Insert entry <t, TIDS_Score> into TIDS Score Table.
   - 5.3. Normalize the TIDS_Score values in TIDS Score Table.

According to the TIDS Score Table Generation Algorithm stemming, which is the process for reducing inflected (or sometimes derived) words to their stem, base or root form [14], generally a written word form, and stop words removal is performed upon the Relevant_Page_Set. Tf-Idf score of the collection is calculated. The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*, df_t is the document frequency of *t*, means the number of documents that contain *t*. The df_t is an inverse measure of the informativeness of *t* also $df_t \leq N$ where N is the total number of documents in the Relevant Page Set. Then the idf (inverse document frequency) of *t* is given by

$$\mathrm{idf}_t = \log\ (N/\mathrm{df}_t) \qquad (1)$$

The Tf-Idf weight of a term t in the document d ($w_{t,d}$) is the product of its tf weight and its idf weight and will be given by

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log\ (N/\mathrm{df}_t) \qquad (2)$$

The TIDS_Score of a term t is given by

$$\mathrm{TIDS\_Score}(t) = \sum_{d \in Relevant\_Page\_Set} \mathrm{tf.idf}_{t,d} \qquad (3)$$

Algorithm 2: First Crawl

1. Create TIDS Score Table using Algorithm 1, for all the pages present in Relevant_Page_Set.
2. Initialize SeedUrls by selecting 600 random links from Relevant_Page_Set.
3. While SeedURls is not empty
   - 3.1 URL=SeedUrls.Next();
   - 3.2 URL_Score= Similarity score of URL.discription terms from TIDS Score Table.
   - 3.3 Enqueue(CrawlQueue,URL, URL_Score);
4. While CrawlQueue is not empty
   - 4.1URL=Dequeue(URL_with_maximum_score, CrawlQueue);
   - 4.2 Doc= Download(URL)
   - 4.3 If Doc is not present in the Crawler Repository then add Doc to the Crawler Repository else GOTO 4.
   - 4.4 Doc_Score= Similarity score of URL.text terms from TIDS Score Table.
   - 4.5 If Doc_Score is greater than or equal to the text Similarity score of Relevant Page Set pages and the Doc is not present in the Relevant Page Set
   - 4.5.1 Add Doc to Relevant Page Set and regenerate TIDS Score Table.
   - 4.6 For all Link in Doc.links
   - 4.6.1 Linkscore= Similarity score of Link.anchor terms from TIDS Score Table.
   - 4.6.2 Score= Doc_Score + Linkscore;
   - 4.6.3 If Score > Relevancy_Threshold
   - 4.6.3.1 Enqueue(CrawlQueue, Link, Score);

According to the Algorithm 2, SeedUrls is initialized by 600 random links chosen from the Relevant_Page_Set. SeedUrls were inserted one by one in the crawler queue, which is a priority queue, as according to their similarity score from TIDS Score table. The crawler picks the URL with maximum score from the queue and downloads the corresponding document. The content similarity score of the page is calculated, and a value for each link present in the document is obtained by merging the parent's content similarity score with the link's own anchor text similarity score, and the link is inserted into the crawler queue. The complete process is repeated until the crawl queue is empty or the maximum crawled page limit is not reached. We executed the First Crawl for collecting 20000 pages, which will act as the relevant page set, R, for the future crawls as they came by crawling seed pages which were related to the focused domain.Hub URL is the one which is pointing to many other URLs and authority URL is the one which is pointed to by many URLs. Best hub is the one which is pointing to many relevant pages and the best authority is the one which is pointed to by many relevant pages. Hubs and authorities exhibit mutually reinforcing relationship. We used the hub score as a learning parameter for the crawler to select best seed pages for the next crawling phase. Let R be the set of pages which are related to the domain and the page P in R bears the interlinked behavior shown in Fig.:1

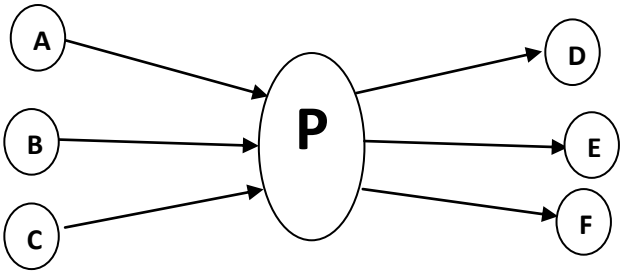
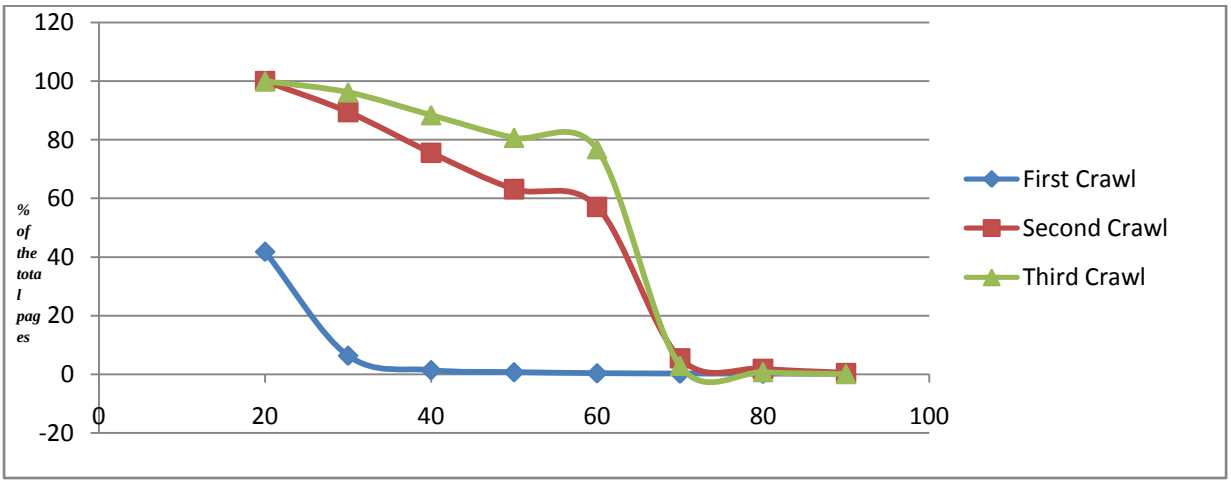


Fig 1: Interlinked behavior of Web page P in R where *{A,B,C,D,E,F,P} Є R*

Fig 2: Graph between the percentage of the total pages retrieved by the crawler (vertical axis) and TIDS relevancy score (horizontal axis).



TIDS relevancy

Then the hub score for the page P in R is given by

$$HUB_p = \sum_{\forall Q \ in \ R \ \exists \ Link\,(P \to Q) \ in \ R} AUTHORITY_Q \qquad (4)$$

And authority score of P is given by

$$AUTHORITY_p = \sum_{\forall Q \ in \ R \ \exists \ Link\,(Q \to P) \ in \ R} HUB_Q \qquad (5)$$

After finding the hub and authority scores we normalize those using mean square root method.

Algorithm 3: Consecutive Crawl

1. Calculate Hub score and Authority score for all the pages present in the set of relevant pages, R, came as a result from the previous crawl.
2. Choose top 600 pages with highest Hub score from R, and initialize them to the SeedUrls.
3. While SeedUrls is not empty
   - 3.1 URL=SeedUrls.Next().
   - 3.2 URL_Score=Hub score of the URL
   - 3.3 Enqueue(CrawlQueue,URL,URL_Score).
4. While CrawlQueue is not Empty
   - 4.1 URL=Dequeue(URL with maximum URL_Score, CrawlQueue).
   - 4.2 Doc=Download (URL).
   - 4.3 Doc_Score=Similarity score of the Doc text and URL anchor text from the TIDS Score Table.
   - 4.4 For all links in Doc.Links
     - 4.4.1 LinkScore=Similarity Score of Link.Anchor terms from the TIDS Score Table.
     - 4.4.2 Score=Merge(Doc_Score,LinkScore)
     - 4.4.3 Enqueue(CrawlQueue, Link, Score)

Consecutive Crawl algorithm works by finding the best hub and best authority pages among the pages which came as result of the previous crawl attempt, top 600 best hubs were chosen to act as the seed pages. All the seed pages are inserted one by one into the crawl queue, which is a priority queue, as according to their hub score. The URL with maximum score is chosen and the document corresponding to it is downloaded. The content similarity score of the page is calculated, and a value for each link present in the document is obtained by merging the parent's content similarity score with the link's own anchor text similarity score, and the link is inserted into the crawler queue. The complete process is repeated until the crawl queue is empty or the maximum crawled page limit is not reached.

4. EXPERIMENTAL RESULTS

The learning capability of the proposed focused crawler is studied for retrieving documents related to information technology domain. The initial collection of Web pages (Seed pages) is generated from the hierarchical categories of ODP (Open Directory Project) from http://dmoz.org as suggested by .Rungsawang, N.Angkawattanawit (2005). ODP provides the categorical collection of URLs that are manually edited and not biased by any commercial user. From here we can find individual categories link. The categories ending with "Information", "computers", "internet", "information technology" and "computer science" were retrieved from the ODP, total 86 such categories were found, then links contained by these 86 categories were retrieved from ODP, total 927 such links were found from ODP or we can say that 927 pages related to the Information Technology domain were found, out of these 927 links 100 links at random were chosen to act as starting URLs for the crawler. The learning effect for three consecutive crawls is observed by finding the number of documents retrieved by the crawler within the different relevancy scores with the TIDS Table. The results are plotted as graph, Fig: 2, between the percentage of the total pages retrieved by the crawler (vertical axis) and TIDS relevancy score (horizontal axis).

The results shows that the second crawling phase retrieves much better results for pages having similarity ranging from 20 to 70. The third crawling attempt improves the pages being retrieved between similarity range 30 to 65, and the number of pages retrieved with relevancy score more than 65 remains nearly same as that of the second crawl. The proposed crawler tends to increase the number of pages retrieved with more relevancy and hence justifying effect of learning upon the crawler performance.

5. CONCLUSION

Focused crawler capable to learn is proposed. Three consecutive runs of the proposed crawler were made to study the effect of learning. The results are plotted as graph between the percentages of the total number of pages retrieved versus the relevancy score of the pages. Results show great improvement in the number of pages having relevancy score between 20 and 70 by the second crawling attempt. A significant improvement is observed in number of pages having relevancy score between 30 and 65 by the third crawling attempt, hence justifying the learning effect of the crawler. The quality of the pages retrieved is increasing with the increase in number of learning phases, and hence providing the pages which are most relevant to the domain.

6. REFERENCES

[1] Brin, S. and Page, L. (1998), 'The anatomy of a large scale hypertextual web search engine',*Computer Networks and ISDN Systems*, 30, pp. 107-117.

[2] C. Aggarwal, F. Al-Garawi and P. Yu.*(* 2001), 'Intelligent Crawling on the World Wide Web with Arbitrary Predicates' ,Proceedings of the 10th international conference on World Wide Web, Hong Kong, pp. 96-105.

[3] D. Bergmark, Carl Lagoze and Alex Sbityakov(2002),'Focused Crawls, Tunneling, and Digital Libraries',Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries, Rome, Italy, pp. 91-106.

[4] Ehrig, M. & Maedche, A.(2003),' Ontology-Focused Crawling of Web Documents', *Proceedings of the Symposium on Applied Computing 2003 (SAC 2003)*. Melbourne, FL, USA, S. Pp 1174-1178.

[5] J. Cho and Hector Garcia-Molina(2002), 'Parallel Crawlers', *Proceedings of the World Wide Web conference (WWW)*, Honolulu, Hawaii.

[6] J. Cho and H. Garcia-Molina(*2000)*,'The evolution of the web and implications for an incremental crawler', Proceeding of 26th International Conference on Very Large Database, Cairo, Egypt, , pp. 200-209.

[7] J. Cho, H. Garcia-Molina, L. Page (1998),'Efficient Crawling Through URL Ordering': **Proceedings of the** *Seventh International* World Wide Web *Conference*, *Brisbane*, *Australia*, pp. 379-388.

[8] L. Page, S. Brin, R. Motwani, T. Winograd (1998), 'The PageRank Citation Ranking: Bringing Order to the Web', Technical report, Stanford Digital Library Technologies Project, pp. 1-17.

[9] Martin Ester, Matthias Groß, Hans-Peter Kriegel(2001), 'Focused Web Crawling: A Generic Framework for Specifying the User Interest and for Adaptive Crawling Strategies',Proceedings of the 27th International Conference on Very Large Database,VLDB2001,Roma,Italy,pp.633-637.

[10] P. Boldi, B. Codenotti, M. Santini, and S. Vigna (2004),'Ubicrawler: a scalable fully distributed web crawler', *Software Practice & Experience*, 34(8), pp. 711–726.

[11] P.M.E. De Bra and R.D.J.Post (1994),' Information retrieval in the World-Wide Web: Making client-based searching feasible', Computer Networks and ISDN Systems. vol. 27, no. 2, pp. 183-192.

[12] *S. Chakrabarti*, *M. van den Berg*, *B. Domc*(1999), '*Focused crawling*: a *new approach* to *topic-specific Web resource discovery*', Proceedings of the *8th international World Wild Web Conference*, *Toronto*, *Canada*, *pp. 1623-1640.*

[13] Ari Pirkola (2007*),' Focused Crawling: A Means to Acquire Biological Data from the Web'*, VLDB '07, Vienna, Austria.

[15] http://en.wikipedia.org/wiki/Stemming, (visited on 10-02-2012).

[16] A.Rungsawang, N.Angkawattanawit (2005)*,' Learnable topic-specific web crawler',Journal of Networks and Computer Applications',pp.* 97-114.