2.1 Introduction

In this Chapter the design and implementations for Breadth First Crawler and Vector Space Model based Best First Crawler are given. All the pre-requisite for both the crawlers are given along with their structure and the way of use. The Chapter is concluded by showing the crawling results for both the crawlers for retrieving the Web pages.

2.2 Open Directory Project (ODP)

ODP also known as DMOZ is a Website index edited by human. DMOZ is an acronym for "Directory Mozilla." It is associated with the Mozilla browser and, is administered and hosted by Netscape Communications and Weblogs, Inc. According to the official Web site, ODP hosts the largest and most comprehensive Web site directory in the world. The purpose of the ODP is to list and categorize Web sites, not to rank or promote them. Although ODP offers a search feature, it differs from conventional search engines that build massive lists of Web sites by automatically searching pages. Each entry is carefully reviewed before inclusion in the directory.

ODP is an open source project. Anyone can submit, download and use ODP data at no cost as long as they accept and heed the site policy and license agreement. The search feature of the ODP works by displaying two kind of results for each query topic submitted to it. The first part shows the
Breadth First and Vector Space Model Based Best First Crawlers

list of categories to which the given topic may belong; the second part gives the list of URLs which are supposed to be related to the given query. The snapshot showing the homepage of ODP is shown in Fig. 2.1. The ODP returned 100 categories for a search query containing the word “software” as shown in Fig. 2.2 in which first 20 categories are displayed out of the 100 categories. A total of 98171 URLs were returned by the ODP for the query “software”, of which first 11 are shown in Fig. 2.3.

The ODP is used for initializing the seed pages for all the crawlers to avoid any biased initialization. There are three main tables around which ODP is working namely resources, xurls, and structure.

Fig. 2.1: ODP homepage from http://www.dmoz.org/ on 15th November 2013
Fig. 2.2: ODP categories for the "software" as the search topic from http://www.dmoz.org/search?q=software&start=0&type=mixed&all=no&cat= on 15th November, 2013

Fig. 2.3: ODP list of URLs returned as a result of the search on "software" topic from http://www.dmoz.org/search?q=software on 15th November 2013
The resources table contains category ID, resource type and resource URL for the category ID as its fields. The xurls table contains category ID, URL belonging to the category ID, URL title, and description for the page pointed by the URL along with other fields. The structure table contains the category ID, topic pointed by that category along with other fields. These three table structures are used to generate the seed URLs. The process for generating seed URLs for a specific domain is given as Algorithm 2.1. The topic for which the related pages are to be retrieved is given as input to the algorithm and the list of URLs which can act as initial seeds are returned by the algorithm.

**Algorithm 2.1: Seed URLs Generator**

*Input: Topic to be crawled, ODP database*

*Output: Seed URLs*

1. Find the entire set of category IDs which points to the topics ending with the Topic to be crawled from the structure table.
2. Retrieve all the URLs from the resources table which are pointed by the set of category IDs found in step 1.
3. Return the retrieved URLs.

The algorithm works by joining the structure table and resources table upon the category ID field for all the topics ending with the topic to be crawled. The set of seed pages is returned as output of Step 3 of the algorithm.
Term Frequency-Inverse Document Frequency (tf – idf) and Vector Space Model (VSM) are used for finding similarity score between a text and the crawler domain.

### 2.3 Term Frequency-Inverse Document Frequency (tf – idf)

The term frequency $t_{f,d}$ of term $t$ in document $d$ is defined as the number of times that $t$ occurs in $d$, $d_{f_t}$ is the document frequency of $t$, and means the number of documents that contain $t$. The $d_{f_t}$ is an inverse measure of the informativeness of $t$ also $d_{f_t} \leq N$ where $N$ is the total number of documents in the set of domain related pages. Then the idf (inverse document frequency) of $t$ is given by

$$
\text{idf}_t = \log\left(\frac{N}{d_{f_t}}\right)
$$

(2.1)

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

$$
W_{t,d} = \log\left(1 + t_{f,d}\right) \times \log\left(\frac{N}{d_{f_t}}\right)
$$

(2.2)

Some of the special cases exist while calculating the $tf – idf$ score. Assume that the value of $N$ is nearly same as that the frequency of $t$ over the whole document collection, $d_{f_t}$. If

$$
1 < \log\left(\frac{N}{d_{f_t}}\right) < \text{small constant}
$$

(2.3)
Then $W_{t,d}$ will be small be smaller than $tf_{t,d}$ but still is positive. This implies that $t$ is relatively common over the entire corpus but still holds some importance throughout the overall collection of documents. This can be the case with the stop words like is, was, this, that etc.

Assume that $tf_{t,d}$ is large but $df_t$ is small then $\log \left( \frac{N}{df_t} \right)$ will also tend to be larger and hence $W_{t,d}$ will also be larger. This implies that the $t$ is very important in that document but may appear in only a few documents in the overall corpus.

### 2.4 Vector Space Model (VSM)

Vector Space Model (VSM) assumes that $t$ distinct terms are there in the document. Each of these $t$ terms will form a vector space of $|t|$ dimensions. Each term of this vector will be given a weight $w_{ij}$. Now the document as well as query is represented in the form of this $t$-dimensional vector

$$d_j = (w_{ij}, w_{2j}, \ldots, \ldots, \ldots, \ldots, w_{tj})$$

(2.4)

In such a way collection of $n$ documents can be represented as a matrix of the form

$$
\begin{bmatrix}
  w_{11} & w_{21} & \ldots & w_{t1} \\
  w_{12} & w_{22} & \ldots & w_{t2} \\
  \vdots & \vdots & \ddots & \vdots \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{1n} & w_{2n} & \ldots & w_{tn}
\end{bmatrix}
$$

(2.5)
An entry in the matrix corresponds to the weight of the term in the document, zero means the term has no significance in the document. This weight may be anything that contains the degree of relevance to the document collection. Many variants are proposed for this weight calculation. $tf-idf$ Weights are used most efficiently for calculation of this term weight. Vector Space Model Weight Vector (VWV) is generated from $tf-idf$ values for finding the similarity score of the text to the topic to be crawled.

One of the important applications of VSM is in the calculations of relevance rankings of the documents from a corpus to the query. The deviation of the angles between the documents and the given query are calculated as shown in Fig. 2.4. The documents having the minimum deviation from the query vector are considered as similar to the query. Minimum is the angle deviation larger is the similarity between the document and the query. The cosine of the angle of deviation is used for determining the similarity instead of the angle only.

The document vectors can be plotted as shown in Fig. 2.4. The Doc1 and Doc2 are the two documents of the corpus, Query is the text entered by the user to determine the similarity. $\alpha$ is the angle between Doc1 and the query, $\phi$ is the angle between Doc2 and the Query. Then the cosine similarity between Query and Doc1 is given by

$$cos \ \alpha = \frac{Doc1 \cdot Query}{\|Doc1\| \|Query\|} \quad (2.6)$$
And the cosine similarity between Query and Doc2 is given by

\[
\cos \theta = \frac{\text{Doc2} \cdot \text{Query}}{\|\text{Doc2}\| \|\text{Query}\|}
\]

(2.7)

The dot product determines the cardinality of the set coming after performing intersection of the two vectors. \(|A|\) determines the cardinality of the set A. As all vectors under consideration by this model are element wise nonnegative, a cosine value of zero means that the query and document vector are orthogonal and have no match.

![Fig. 2.4: Vector representation of documents and query](image)

**2.4.1 Vector Space Model Weight Vector Generation**

One of the requisite for any information retrieval system is the measure used for denoting the similarity value between two documents or texts. Vector Space Model Weight Vector (VWV) obtained from the
The VWV generation works as according to the Algorithm 2.2.

The VWV generator algorithm works by initializing the set of seeds with the help of ODP using Algorithm 2.1. The set of seeds so generated are further pre-processed for tokenization and stop words removal. The set of seed pages coming after pre-processing are inserted into the Relevant_Page_Set. Relevant_Page_Set is subjected to Equation 2.1 and Equation 2.2 for calculating \( tf-idf \) values.

The \( tf-idf \) values along with the text term are arranged to form Inverted Index Table. The Inverted Index Table stores \( tf-idf \) score for each word for different documents. After generating Inverted Index Table the sum of the values of \( tf-idf \) for a term over all the documents in Relevant_Page_Set is calculated, and an entry of the term along with the total \( tf-idf \) is made to the VWV table. The values in the VWV table are further subjected to normalization.

Using the VWV table the similarity of a text document is calculated by representing the document in the form of a vector and then by summing the values obtained by each term of the vector from VWV table. This gives a similarity score of the document to the collection of the relevant pages.

### 2.5 Breadth First Search Crawler

The Breadth First Crawler works on the principle of locality of reference on the Web. The principle of locality of reference says that the possibility of reaching to related pages from the links of related pages is
more than the possibility from the links of general pages. The architecture of a Breadth First Search crawler is given is given in Fig. 2.5.

**Algorithm 2.2: Vector Space Model Weight Vector (VWV) Generator**

*Input*: Topic to be crawled.

*Output*: Vector Space Model Weight Vector (VWV)

1. Generate the seed pages for the Topic to be crawled using Algorithm 2.1.
2. Initialize Relevant Page Set to be the set of seed pages coming after pre processing.
3. Generate tf — idf Score Inverted Index Table for all the contents of all the Relevant Page Set.
4. For each term t in the tf-idf Score Inverted Index Table Do
   4.1. Calculate sum of the tf — idf score obtained by t in all documents from tf — idf Score Inverted Index Table, let it be Score.
   4.2. Insert entry <t, Score> into VWV.
   4.3. Normalize the Score values in VWV.

Breadth first crawler starts from the seed pages (pages highly related to the crawler domain) and download them in first come first serve basis. The URLs contained within the document are extracted and inserted at the end of the crawler queue. The crawler functions as according to Algorithm 2.3. It works upon the principle of BFS, which forbid the crawler to go for the next node until all the nodes at a specific level are not completely visited.
The seed URLs are generated from ODP, and are inserted into the crawler queue. The crawler fetches the URL from the queue and downloads the page from the Web. All the links contained within the page are inserted at the end of the crawler queue. The whole process is repeated until the crawler queue is not empty or the crawler limit is not reached.
Algorithm 2.3: Breadth First Search Crawler

Input: Domain topic and Seed pages from Open Directory Project (ODP) database.

Output: The set of pages retrieved by the crawler

1. Initialize the seed pages from the Open Directory Project (ODP) using Algorithm 2.1
2. Put all the seeds into the Crawler Queue.
3. Download all the pages directed by the seed URLs, apply tokenization, and stop word removal and text extraction to each of the URL's downloaded contents.
4. Calculate tf – idf score for each term of the corpus of the pre-processed pages using Equation 2.1 and Equation 2.2.
5. Generate VWV table using Algorithm 2.2.
6. While Crawler Queue is not Empty or the crawler limit is not reached
   a. Fetch the first URL from the Crawler Queue; download the corresponding page from the Web. Calculate page's similarity to the domain using VWV table score, and insert into the Results database. Extract all the links contained within the page and insert them at the end of the Crawler Queue.

2.6 Vector Space Model (VSM) based Best First Crawler

Vector space model based best first crawler makes use of the Vector Space Model Weight Vector (VWV) table score derived from the vector representation of the corpus documents and the tf – idf scores to guide the
focused crawler. The VWV table score act as heuristic for deciding the next URL to be crawled. The crawler architecture is given in Fig. 2.6.

The crawler starts from the seeds generated with the help of ODP. The seed pages are pre-processed for tokenization, stop words removal and text extraction. The extracted contents are passed to the next module for score generation which is further used to generate the VWV table.

Fig. 2.6: Vector Space Model Based Best First Crawler
The score table so generated is used to find the similarity score between the domain and the Web page. The Crawler Queue is a priority queue maintained upon the similarity score numeric field. The crawling algorithm for the crawler is given as Algorithm 2.4.

**Algorithm 2.4: VSM Best First Search Crawler**  
*Input:* Domain topic and Open Directory Project (ODP) database.  
*Output:* The set of pages retrieved by the crawler based upon VSM score

1. Initialize the seed pages from the Open Directory Project (ODP) using Algorithm 2.1.  
2. Pre-process the contents of the Web page corresponding to each seed page after downloading it from the Web.  
3. Generate $tf-idf$ weight score matrix for the seed pages using Equation 2.1 and Equation 2.2.  
4. Generate VWV using Algorithm 2.2.  
5. Insert all the seed URLs into the crawler queue as according to their content's similarity total score from the VWV.  
6. While Crawler Queue is not Empty or the crawler limit is not reached  
   - a. Pop topmost URL from the Crawler Queue, and fetch it from the Web.  
   - b. Apply pre-processing operations on the downloaded document.  
   - c. Calculate the content's similarity score of the page from the VWV table.  
      1. If Similarity Score $>$ Threshold Value then  
         i. Insert the page into the results database along with the similarity score.  
         ii. For each link present in the page calculate score as  
             \[
             \text{score} = \frac{\text{parent content similarity}}{7} + \text{link anchor similarity}.  
             \]  
         iii. Insert the links into the Crawler Queue as according to its score's location.
The algorithm uses ODP to initialize the set of seeds using Algorithm 2.1. The seed pages are pre-processed as according to Step 2.

The VWV table is generated using the Algorithm 2.2 in Step 4 of the crawling algorithm. All the seed URLs along with their similarity score are inserted into the Crawler Queue as shown by Step 5. Step 6 denotes the beginning of the main crawling loop. The first step of the crawling loop fetches the URL from the Crawler Queue which is having the largest similarity score. The page corresponding to the fetched URL is downloaded from the Web. Different pre-processing steps are applied to the downloaded page. The VWV table based similarity score for the contents of the page is calculated. If the similarity score is greater than the threshold value then the page is inserted into the result database along with its similarity score.

All the links contained in a relevant page are extracted and a score derived from VWV table, score of the parent page, score of the anchor text is calculated. The links are inserted into the Crawler Queue as according to the similarity score. The irrelevant page is discarded at this point of time only and is not explored for any process further. The whole process is re-iterated until the Crawler Queue is not empty or the crawler limit is not reached.

2.6 Results

Both the crawlers are implemented in Java using a Windows 7 setup on an Intel(R) Core(TM) i5 Processor 3.20 GHz and 4.00 GB of RAM, and
Breadth First and Vector Space Model Based Best First Crawlers

using MySQL as the backend database. An automated process for gathering seed pages is used. Open Directory Project (ODP) provides the categorical collection of URLs that are manually edited and not biased by any commercial user. From here one can find individual category links. One can find maximum number of URLs related to the concerned topic by just browsing the http://www.dmoz.org from root to the topic of interest.

For analysis and other purposes http://www.dmoz.org also provides the whole database free of cost, which can be used as according to the needs of the individual. This database was downloaded and its contents were extracted to MySQL using PERL (Practical Extraction and Report
Language) scripts. The URLs belonging to the categories which are pointing to “Animation”, “Cricket”, “Science” and “Computer” at the last level were retrieved.

A total of 323, 199, 250 and 563 such URLs were found through a module written in Java for each of the “Animation”, “Cricket”, “Science” and “Computer” domain respectively. The set of these URLs acted as the initial seed pages for both the crawlers. The precision graphs for different similarity values for different topic domains are shown from Fig. 2.7 to Fig. 2.10.

![Fig. 2.8: Precision graph for "Cricket" domain](image-url)
Breadth First and Vector Space Model Based Best First Crawlers

Fig. 2.9: Precision graph for "Science" domain

Fig. 2.10: Precision graph for "Computer" domain
2.8 Conclusion

In this chapter the architecture and implementation details for Breadth First Crawler and Vector Space Model Based Best First Focused Crawler were given. All the pre-requisite like initial seed page generation from Open Directory Project, Vector Space Model Weight Vector generation and domain topic-page similarity calculations were mentioned. Results for retrieving pages related to four crawler domain were plotted as precision graphs.