CHAPTER 4: SEMANTICS AND LEARNING BASED FOCUSED CRAWLER

4.1 Introduction

In this Chapter the architecture of a focused crawler based upon semantics and learning is proposed. The proposed crawler uses Open Directory Project (ODP) for an automated process of generating the seed URLs. It makes use of three major criterion for guiding the future crawl and accordingly to reorder the Crawler Queue for retrieving the best possible results. The three major criterion used by the proposed crawler are:

a. Content similarity and link similarity using Vector Space Model.

b. Semantic similarity using WordNet semantics.

c. Next phase seed selection based upon Hub scores.

Seed URLs generator, as discussed under Section 2.2 of Chapter 2, is used to generate the set of initial seed pages for the focused crawler from Open Directory Project (ODP).

The Vector Space Model (VSM), discussed under Section 2.4 of Chapter 2, 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. The corpus containing \(n\) number of documents can be represented using \(n \times t\) a matrix. Each of
these $n$ documents along with the query vector can be drawn in space, from where the similarity between them can be calculated.

Vector Space Model Weight Vector (VWV) Generation algorithm, discussed under Section 2.4.1 of Chapter 2, is used to generate the weight vector which is to be used for determining the similarity of documents with the topic domain. The algorithm makes use of $tf - idf$ scores for weighing the document terms.

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.

4.2 WordNet

G. A. Miller [123] started working to combine electronic dictionaries and on-line thesauri, and came with WordNet in 1994. WordNet is a large lexical database of English language. It is a tool that consists of lexical units and relationship between these lexical units in the form of a structured semantic network. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. It is also freely and publicly available for download.
WordNet's structure makes it a useful tool for computational linguistics and natural language processing. It superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions. First, WordNet interlinks not just word forms-strings of letters—but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated. Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus do not follow any explicit pattern other than meaning similarity. It is being used as an ideal tool for information retrieval and semantic tagging, and meaning disambiguation. The rapid appearance of the newer versions and well organized promotion of the WordNet makes it one of the popular tools for Natural Language Processing (NLP).

EuroWordNet [124] based upon the WordNet is a multilingual databases for many European languages. Each WordNet makes use of an independent lexicalization and is interconnected by an inter-linguistic index.

WordNet started with three assumptions. First was the separability hypothesis, according to which the lexical component of every language can be separated and analyzed individually. Hence the lexicon can be considered as an independent entity to the phonology and grammar of the language. This seems to be important by citing that the vocabulary of a
Second assumption was the **patterning hypothesis**, according to which the human being could not master all the lexical knowledge of a language. One could not remember every meaning of a single word; instead, the syntactic patterns and relationship between words make it understood.

Third assumption was the **comprehensiveness hypothesis**, by which the computational linguistics needs a lexical storage like the human being, which is applied to process natural.

The **synsets** came in existence in 1984, when George Miller created a small net of 45 nouns and started working upon the WordNet project with Bellcore in collaboration with Princeton University. Synsets are sets used to represent the lexical concepts by the combination of words and senses. The Grinder is considered as the most important part of the WordNet. Grinder accepts lexical files from the lexicographers and automatically makes a lexical database out of it. First Grinder was developed by Bienkowski [125] in was redeveloped in C by Dan Tiebel [126].

Rewriting Grinder was needed because any change to the WordNet also affects the Grinder. With the growing number of words the need for database distribution was arose. The first distribution of the database came in the form of division in syntactic categories, by making different places for verbs, adjectives, adverbs, and nouns. Word Filter that took a text file as input and produce different kinds of information regarding the words in the text as output was developed. Still the WordNet was unable to filter
plurals. So a lemmatization program that made use of some exceptions to determine words with a regular morphology was developed.

For improvement in coverage of words and meanings in WordNet semantic tagging was performed. For performing semantic tagging ConText was developed. ConText performed word tagging, tokenization, and lemmatization before generating the target word along with the WordNet meaning.

In 1991 first version of WordNet was made publicly available. Its current version 3.1 is available online at WordNet website: http://wordnet.princeton.edu/. It can be downloaded and used freely.

Disambiguation is one of the important applications of WordNet in information retrieval systems. It can be used in traditional information retrieval systems to enhance the efficiency. It may be used with a classifier for efficiently classifying the documents. It can be combined with neural network to process local contexts. WordNet can also be combined with Bayesian network to establish lexical relations as source of knowledge. It is used for statistical and symbolic information integration [127].

WordNet lexical relations are used for statistical classifier development which can identify the meanings of the words from the combination of context and local signs [128]. It is also used as support for the development of a computational similarity model to add on-line semantic representation to the statistical corpus. WordNet has, therefore, proved its worth as an ideal methodological element to disambiguate the meaning of words in information extraction systems [129]. As a result,
projects have been launched to disambiguate nouns in English language texts using specification marks deriving from WordNet taxonomies as a knowledge base, as well as to reduce polysemy in verbs, classified by their meanings via predicate associations, with a view to optimizing information retrieval.

The proposed crawler makes use of WordNet to find the words which are semantically similar to the words present in the VWV, which in turn came from the documents highly related to the crawler domain (after pre-processing). Now the VWV contains the old term and weights as well as the words semantically similar to the existing terms with score equal to that of the original term, this new vector is named as Semantic VSM Weight Vector (SVVV). This new vector is used by the proposed crawler to deduce the semantic similarity, content and link similarity of the candidate page for possible inclusion into the future crawl page category.

4.3 Hyperlink-Induced Topic Search (HITS)

Kleinberg [38] in 1999 observed that the in-link count is not the only parameter for deciding the relevancy of a page, but the pages pointing to it should contain some large overlapping pattern as they all belong to some specific Web community. This observation led to the existence of hubs and authorities. Pages which are pointed to by many pages are termed as authorities. These pages are supposed to contain the information related to a particular domain. Hubs are the pages which are pointing to many other pages. For retrieving the good quality results a focused crawler should
target the hubs which are pointing to the authority pages related to the domain. The hubs and authorities exhibit the mutual reinforcement property as shown in Fig. 4.1.

Figure 4.1 shows an interlinked portion of the Web. The pages \{A,B,C,D,E\} are contributing towards the authority score for the page P, the page P is further contributing towards the hub score for pages \{F,G,H\}. This reinforcement relationship between the hubs and authority is used by Kleinberg to determine the hub score and authority score of a page on the subset of the Web.

HITS is a link analysis algorithm that rates Web pages. The first step of the HITS algorithm is to retrieve the most relevant pages to the search query. This set is called the root set and can be obtained by taking the top n pages returned by a text-based search algorithm. A base set is generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it. The web pages in the base set and all
hyperlinks among those pages form a focused sub-graph. The HITS computation is performed only on this focused sub-graph. The 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.

The hub score for the page \( P \) in \( R \) is given by

\[
HUB_P = \sum_{Q \in R} \sum_{\text{Link}(P \rightarrow Q) \in R} \text{AUTHORITY}_Q
\]

(4.1)

And authority score of \( P \) is given by

\[
\text{AUTHORITY}_P = \sum_{Q \in R} \sum_{\text{Link}(Q \rightarrow P) \in R} \text{HUB}_Q
\]

(4.2)

### 4.4 Proposed Crawler’s First Phase

First phase of the crawler is depicted in Fig. 4.2. Seed URL Generator from Open Directory Project generates a set of Seed URLs belonging to the topic to be crawled. The URL downloader fetches the seed URL from the Web and sends it to the text extractor, tokenizer, and stop words remover. Text extractor retrieves the content of Web page in the form of text using...
HTML parsing techniques. The contents so generated are passed for tokenization, which is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens.

The list of tokens generated by the tokenizer is processed for removing the stop words. Stop words like is, am, are, able etc. are some extremely common words which would appear to be of little value in helping select documents matching a user need and hence are to be removed to save execution time and run time space consumed.

The \( tf - idf \) (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. \( tf - idf \) for the corpus containing the seed pages is generated using Equations 2.1 and 2.2 from Chapter 2.

Vector Space Model Weight Vector Generator works upon the \( tf - idf \) values calculated for the set of seed pages, which are coming after pre processing.
Semantics and Learning Based Focused Crawler

Seed URL Generator

Topic to be crawled

Text extractor
Tokenizer
Stop Words Remover

Pre-processors

TF-IDF score generator for contents of Seed URLs

Semantic VSM Weight Vector (SVWV) generator from WordNet dictionary and VWV

Semantic Similarity (SS) Score Calculator for Seed pages contents using SVWV

Vector Space Model (VSM) Weight Vector (VWV) Generator

Crawl Database

Links relevancy score generator based upon the parent’s similarity, link text similarity, and anchor text similarity score

Next URL downloader

Document’s domain relevancy decider based upon SVWV score of vector

Fig. 4.2: Proposed focused crawler’s first phase

URL: http://www.wqq
SS Score: 1.0876

97
Algorithm 4.1: SVWV Generator

Input: WordNet database and VWV table.
Output: SVWV

1. For each term $t$ present in VWV Do
   a. Insert $t$ in the SVWV along with its score from VWV.
   b. From WordNet dictionary retrieve all words which are semantically similar to $t$ and assign them the score same as that of $t$ and put in the SVWV along with its score, if it is not already there.

Semantic VSM Weight Vector (SVWV) generator makes use of WordNet dictionary and Vector Space Model Weight Vector to generate SVWV scores.

The SVWV score acts as the semantic similarity score between the focused domain and other text. It works as according to the Algorithm 4.1. The SVWV algorithm is called with WordNet and VWV table as arguments. The VWV table is processed on term by term basis and the words which are semantically similar are retrieved from the WordNet and are inserted into the SVWV table.

The Crawler Queue is a priority queue maintained upon the Score numeric field, and contains the unseen URLs along with their Scores. Vector Convertor generates the vectors corresponding to each text.
representation of the Web page, so that it could be compared to the crawling domain to generate the similarity score.

The crawler picks the first URL from the crawler queue, fetches it from the Web, and the fetched page is sent for preprocessing to the parsers. In addition to the other parsers there is one more that constructs the vectors out of the page's HTML contents. These vectors are used by the relevancy decider to decide which page is relevant and which is not. The relevant pages are further sent to the links relevancy score generator and to the crawl database which contains the relevant pages along with their relevancy scores, the irrelevant pages are discarded. After generating the relevant page the relevancy scores for the links contained within the page are determined and are inserted into the crawler queue along with their relevancy scores. This phase of the proposed crawler works as according to the Algorithm 4.2.

Algorithm 2.1 from Chapter 2 is used in Step 1 to initialize the set of seed pages with the help of Open Directory Project database, the topic to be crawled is given as input and the set of seed pages is received.

Step 2 of the algorithm applies various preprocessing techniques to the contents of the seed pages. \( tf-idf \) Values for the contents of preprocessed seed pages are generated by Step 3 of the Algorithm.

VSM Weight Vector is generated by Step 4 of the proposed algorithm by using \( tf-idf \) values as the term weights.
Algorithm 4.2: Proposed crawler’s first phase

**Input:** Open Directory Project (ODP) database, domain topic for focused crawler, WordNet.

**Output:** The set of pages, R, having Semantic VSM Weight Vector (SVWV) score is amongst the highest.

1. Initialize the seed pages from the Open Directory Project (ODP) using Algorithm 2.1 from Chapter 2.
2. Apply pre-processing techniques to each individual document in the seeds.
3. Generate $tf-idf$ weight score matrix for the seed pages using Equation 2.2 from Chapter 2.
4. Generate a single vector comprising term and the sum of $tf-idf$ score of the term from all the documents vectors, VSM Weight Vector (VWV), using Algorithm 2.2 from Chapter 2.
5. For each term present in the VSM Weight Vector
   a. Find the words which are semantically similar to the term by using WordNet semantic relations, and assign them the score as that of the original term from the VSM Weight Vector and insert `<term, weight>` into the Semantic VSM Weight Vector (SVWV) as according to the Algorithm 4.1.
6. For each seed page
   a. Calculate similarity score of the page’s contents from the Semantic VSM Weight Vector (SVWV).
   b. Insert `<page, similarity score>` in the Crawler Queue, which is a priority queue maintained upon the score numeric field.
7. While Crawler Queue, which is a priority queue, is not Empty or the crawler limit is not reached
   a. Retrieve the topmost URL from the Crawler Queue, and fetch it from the Web.
   b. Calculate its content’s similarity score from the Semantic VSM Weight Vector (SVWV), and insert the page into the results database R, if its score is greater than the relevancy threshold.
   c. For each link present in the page calculate score as $score = \frac{\text{parent.content.similarity}}{7} + \text{link.anchor.similarity}$.
   d. Insert the links into the Crawler Queue as according to their score's
The semantically similar words for each word from the VWV are found by using WordNet semantic relations, and are assigned same score as that of the word from VWV. WordNet's semantically similar words along with VWV words form a new vector called Semantic VSM Weight Vector (SVWV) and is generated by using Step 5 of the algorithm.

The semantic score for each seed page is calculated from SVWV using Step 6 (a) of the algorithm by adding the score of each individual term present in the seed page from the SVWV table.

All the seed URLs are inserted into the crawler queue as according to their semantic score's position using Step 6 (b) of the proposed algorithm.

Step 7 represents the main focused crawling loop. The topmost URL from the crawler queue is fetched from the Web using Step 7 (a), and its content's semantic similarity score is determined from the SVWV table using Step 7 (b). If the semantic score of the page comes out to be greater than the threshold value, the page is inserted into the results database and the page is considered to lead to other relevant pages as given in Step 7 (b). If the page under consideration is found to be relevant then all the links contained within the page are also to be explored further. The semantic score of the links contained within the relevant page is obtained by addition of the SVWV score for its anchor text and one seventh part of SVWV score of the parent page as mentioned in Step 7 (c). One seventh part of the parent's score is added to predict the similarity score of the
child page because the average number of out-links from a Web page is 7 as given by Kumar et. al [130]. The links contained within the page are inserted into the crawler queue as according to their semantic score’s location as mentioned in Step 7 (d).

4.5 Proposed Crawler’s Consecutive Phase

Output result set R coming after execution of the previous crawling phase is passed to the next phase. Architecture for the consecutive phase of the proposed crawler is given in Figure 4.3. The next phase finds the best hubs out of pages coming from the previous phase. Hub URL is the one which is pointing to many other URLs and the 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. HITS algorithm is used to calculate the hub and authority score of all the URLs present in set R. Hubs and authorities exhibit mutually reinforcing relationship. The hub scores and authority scores are calculated as according to the Algorithm 4.3. Hub and authority score calculation algorithm takes the set, S, as input and calculates hub and authority score for each page present, as accordingly to their mutual-reinforcement relationship.

Initially hub and authority score for each page in S is initialized to 1 by Step 1. The hub score of a page, P, is calculated to be the sum of authority score of all the pages, Q, in S for which there is a link from P to Q as given by Step 2(a). The authority score of a page, P, is calculated to be
the sum of hub score of all the pages, Q, in S for which there is a link from Q to P as given by Step 2(b).

The sum of squares of the hub score and sum of squares of authority scores of all the pages in S is calculated by Step 4. Step 5 calculates the square root of the sum of hub score and the square root of the sum of authority score obtained in Step 4. Step 5 calculates the square root of the sum of squares of hub and authority scores calculated in Step 4. The normalized hub and authority scores are obtained using Step 6 of the algorithm.

After generating the normalized hub and authority scores, a semantic similarity score is calculated by combining the normalized hub score and SVWW score. All the URLs are inserted into the crawler queue in order of their semantic similarity scores.

The Crawler Queue is a priority queue maintained upon the Score numeric field, and contains the unseen URL along with their Scores. Vector Convertor generates the vectors corresponding to each text representation of the Web page, so that it could be compared to the crawling domain to generate the similarity score.

The crawler picks the first URL from the crawler queue, fetches it from the Web, and the fetched page is sent for preprocessing. In addition to the other pre-processing techniques there is one more that constructs the vectors out of the page’s HTML contents. These vectors are used by the relevancy decider to decide which page is relevant and which is not.
The relevant pages are further sent to the links relevancy score generator and to the crawl database which contains the relevant pages along with their relevancy scores, the irrelevant pages are discarded.

Figure 4.3: Proposed focused crawler's consecutive phase
After generating the relevant page the relevancy scores for the links contained within the page are determined and are inserted into the crawler queue along with their relevancy scores.

**Algorithm 4.3: Hub Authority Calculation**

**Input:** Set of pages, S

**Output:** Hub and Authority score for each page in S

1. For all pages in S
   \[ HUB_p = AUTHORITY_p = 1 \]

2. For each page p in S
   a. \[ HUB_p = \sum_{q \in S \text{ Link}(p\rightarrow q) \in S} AUTHORITY_q \]
   b. \[ AUTHORITY_p = \sum_{q \in S \text{ Link}(q\rightarrow p) \in S} HUB_q \]

3. Repeat Step 2 until HUB and AUTHORITY vectors do not converge.

4. Calculate \( SqHUB \) and \( SqAuthority \) as
   \[ SqHUB = \sum_{p \in S} HUB_p^2 \]
   \[ SqAuthority = \sum_{p \in S} AUTHORITY_p^2 \]

5. Calculate \( SqrtHUB \) and \( SqrtAuthority \) as
   \[ SqrtHUB = \sqrt{SqHUB} \]
   \[ SqrtAuthority = \sqrt{SqAuthority} \]

6. For each page p in S
   \[ HUB_p = \frac{HUB_p}{SqrtHUB} \]
   \[ AUTHORITY_p = \frac{AUTHORITY_p}{SqrtAuthority} \]
This phase of the proposed crawler works as according to the
Algorithm 4.4.

Algorithm 4.4: Consecutive phase crawl for the proposed crawler
Input: Set of page, R, generated from the previous crawling attempt.
Output: Set of pages, T, related to the domain in some greater degree than
from the previous crawl.

1. Generate normalized HUB and AUTHORITY vectors for all the pages in
   R using Algorithm 4.3.
2. For each page p in R
   a. Calculate \( RANK_p = HUB_p + p.\text{content}.\text{Semantic VSM Weight}
      \quad \text{Vector (SVWW).score} \)
3. Insert M number of pages p from R, which are having highest \( RANK_p \)
   score, in the Crawler Queue, which is a priority queue maintained upon
   the score numeric field.
4. While Crawler Queue, which is a priority 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. Calculate its content's similarity score from the Semantic VSM
      Weight Vector (SVWW), and insert the page into the results
      database, T, if its score is greater than the relevancy threshold.
   c. For each link present in the page calculate score as
      \[
      \text{score} = \frac{\text{parent.content.similarity} + \text{link.anchor.similarity}}{7}
      \]
   d. Insert the link into the Crawler Queue as according to its score's
      location.
Step 1 generates the normalized hub and authority score for the URLs coming after the execution of the previous phase of the crawler using Algorithm 4.3. Step 2 calculates the semantic similarity score for the URLs by combining the hub score and the SVWV score for the contents of the resultant pages from the previous phase.

The URLs with their respective scores are inserted into the crawler queue, which is again a priority queue maintained upon the score numeric field. The crawler picks the first URL from the crawler queue, fetches it from the Web, and the fetched page is sent for preprocessing. Vectors for the HTML contents are created. These vectors are used by the relevancy decider to decide which page is relevant and which is not. The relevant pages are further sent to the links relevancy score generator and to the crawl database which contains the relevant pages along with their relevancy scores, the irrelevant pages are discarded.

After generating the links relevancy score for the links within the relevant page, the links are inserted into the crawler queue along with their relevancy scores.

4.6 Results Showing Effect of Consecutive Crawling

The consecutive phase of the proposed crawler starts by learning the new seeds to start from the results of the previous crawling phase. The new seeds are generated by considering the hub score and semantic similarity
score of the pages from the previous crawl results. A fixed number of highest scoring URLs are chosen to act as seed pages for the next crawling phase. To study the consecutive phase effect four consecutive runs of the proposed crawler were made for collecting related pages for “Animation”, “Cricket”, “Science” and “Computer” domain. After each crawl phase 200 new seeds were generated for the next crawl.

The proposed crawler is 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 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.
Semantics and Learning Based Focused Crawler

- Proposed Crawler Phase-1
- Proposed Crawler Phase-2
- Proposed Crawler Phase-3
- Proposed Crawler Phase-4

**Fig. 4.4:** Precision graph for "Animation" domain for consecutive phases of the proposed crawler

**Fig. 4.5:** Precision graph for "Computer" domain for consecutive phases of the proposed crawler
Proposed Crawler Phase-1
Proposed Crawler Phase-2
Proposed Crawler Phase-3
Proposed Crawler Phase-4

Fig. 4.6: Precision graph for “Science” domain for consecutive phases of the proposed crawler

Fig. 4.7: Precision graph for “Cricket” domain for consecutive phases of the proposed crawler
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 the first phase of the proposed crawler. The precision graphs for different similarity values for different topic domains for different phases of proposed crawler are shown in Fig. 4.4 to Fig. 4.7.

The learning effect of the consecutive crawling can be observed by looking at the improvement in precision value for all values of the similarity scores for all the crawling domains under observations. Number of documents retrieved by the crawler having relevancy score greater than the fixed value is considered to be relevant to the individual domain. The results are plotted as graphs between the precision value (vertical axis) and the similarity score (horizontal axis).

The results show that the precision value tends to increase with the increasing number of the crawling attempt for almost all similarity values for the downloaded pages.