4. SEMANTIC CRAWLER AND SEMANTIC CONTENT MARKER

4.1 INTRODUCTION

SCMS is a new trend of CMS, where knowledge extraction plays a significant role in useful mining patterns. The semantic analysis of stored documents implements the knowledge mining. It provides a solution to relevant and accurate retrieval from CMS. The information retrieval from search engines and most CMS are challenging. The keyword-based retrieval may return unrelated data with appropriate knowledge. Most of the existing search engine giants prioritize the results of the search which satisfies the user's requirement [82]. But the generic search engines exhaustively search the web, it retrieves all the separate documents based on the linked keyword to the query. Domain-based segregation and accurate retrieval of the topics from a specific category is highly impossible. There is a need for filtering technique in all the levels of information retrieval.

A user blog focused crawler fetches as many as blogs possible based on the given domain. This crawler is similar to the domain-specific crawler which brings significant services possible for particular users [83]. The harvested URLs are stored in the repository. The crawling should be done periodically whenever the user query is new. It is updated from time to time based on the end user's query [84]. The user utilizes the stored URLs and its content through a semantic user interface [85]. The user can mark the named entities in the interface and map with the links referred. The entities are annotated [86] with the referred links for the created Virtual Document (VD) in the semantic user interface. The high-frequency keywords are picked up from the interface and can be utilized for further search based on the specified topic. The suitable repository is identified by considering the storage of URL and the web page content. There is a need for semantic algorithms which can be utilized for retrieving the already searched queries using the interface. The process of data extraction, transformation, loading, authoring and retrieval are performed by the semantic crawler and Semantic Content Marker (SCM). In the following sections, the design, implementation of the
semantic crawler and SCM are elaborated. The result is evaluated and compared with the existing systems.

4.1.1 Data Extraction, Transformation and Loading using SCM

Authoring template plays a vital role in the CMS. The content should be marked or selected and annotated only through the template. Content marker act as a user interface, authoring template and publishing manager. In this phase, the web data is stored and retrieved with a semantic annotation. Semantic algorithms are proposed for pre-processing, comparing and tracking the changes. Extraction, transformation & loading are three main components of the semantic content marker system. The URL harvesting is the primary work, since suitable data is selected as source documents.

Figure 4.1: SCM for Data Extraction, Transformation and Loading in to DFS

A web crawler is designed which automatically collects the URLs and sources which matches the input query of the user. The content and links harvested from the web are stored in the suitable repository. Loading of data using RDBMS, NoSQL and Distributed Data model into DFS is facilitated. SCM act as a semantic user interface, where the user can create new VD’s, link the source documents, annotate the named entities. Two kinds of end users access the SCM in the DSCMS model. The active user may either key...
in the new query which is not in the store or historical user who can give an existing query. If the query is unique, then semantic crawler is triggered. If the query is old then results from the historical user are retrieved. There are three types of SCM’s developed and implemented in the DSCMS model. Distributed SCM, Dynamic SCM and Static SCM. The comparison among all these SCM’s is evaluated. The working of SCM with all the three types is shown in Figure 4.1. The distributed SCM comprises of a semantic crawler which harvests URL and its content in a parallel manner. The working of proposed semantic crawler along with the distributed SCM is explained in the forthcoming sections. Dynamic SCM is designed and implemented with concept matching technique using search API automatically. The processes involved in the dynamic SCM are also elaborated. Three kinds of SCM are developed as it is very difficult to create a SCM (UI) for DFS directly, but it can be easily interfaced by either RDBMS or NoSQL. Data ingestion is necessary in the process of storage in DFS when batch of real time data is absorbed, hence it is facilitated by the SCMS which are depicted in case study.

The Static SCM extracts the content from the web manually and the structured data is stored into the RDBMS. The transformation and pre-processing techniques are executed in the SCM’s after extraction of data. Loading of data from different database like RDBMS, NoSQL and Data Model into DFS are facilitated for reduction of execution time and facilitation of analysis in DFS.

4.2. SEMANTIC STRUCTRE AND META-KEYWORD BASED CRAWLER (SSMKC)

The semantic crawler is a part of distributed SCM. The domain expert user triggers the crawler for a new query and updating of the repository. The crawler program ought to crawl only part of the information from the web. Search engines are brilliant in tackling these issues to save the current indices with extensive crawling. Domain knowledge is very much crucial in the user-focused crawling. These focused crawlers retrieve only related URL’s and their content based on domain ontology. The meta-keyword of the
specified topic is preserved and matched with the domain ontology of NLP tool. The architecture of Semantic Meta-Keyword based Crawler (SMKC) is shown in Figure 4.2.

![Figure 4.2: Architecture of SSMKC](image)

### 4.2.1 User-Focused Crawling Process

The user-focused blog crawling process is described below. The information in World Wide Web is vast and hence it is a time-consuming process to crawl and bind the URL based on the domain-specific blogs. A collection of domain-specific blogs can easily compare and map the sites in the repository. The seed URL can impact the results of crawler [87] to a greater extent. The seed URL list is generated based on the list taken from wiki and other blogs based on the relevancy scores.

More than ten popular search engines like Google, Yahoo, Bing, Ask.com, Light Switch and Meta search engine have been used to create seed URLs. Table 4.1 gives the list of the sample seed URLs and relevancy scores. The crawling process in a run is based on a distributed environment which enhances the speed of the crawler to a considerable extent. It is essential to mark the URLs which are unrelated to the domain. Multiple pages with ten downlinks are downloaded from the web. The seed URL is
used for matching the meta-keyword, moreover locking and synchronization is done only using active crawler program. In the proposed thesis the criteria to stop the depth of the crawler is fixed to be ten. The process of extracting links is similar to the growing a node in a graph search problem [88].

Table 4.1 - Sample Seed URLs and semantic relevancy score: IT domain

<table>
<thead>
<tr>
<th>Seed URLS</th>
<th>Relevancy Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://en.wikipedia.org/wiki/Cloud_computing">https://en.wikipedia.org/wiki/Cloud_computing</a></td>
<td>0.753</td>
</tr>
<tr>
<td><a href="https://en.wikipedia.org/wiki/Data_processing">https://en.wikipedia.org/wiki/Data_processing</a></td>
<td>0.665</td>
</tr>
<tr>
<td><a href="https://en.wikipedia.org/wiki/Data_visualization">https://en.wikipedia.org/wiki/Data_visualization</a></td>
<td>0.814</td>
</tr>
<tr>
<td><a href="https://en.wikipedia.org/wiki/Computer_network">https://en.wikipedia.org/wiki/Computer_network</a></td>
<td>0.892</td>
</tr>
<tr>
<td><a href="https://en.wikipedia.org/wiki/Information_technology">https://en.wikipedia.org/wiki/Information_technology</a></td>
<td>0.639</td>
</tr>
<tr>
<td><a href="https://en.wikipedia.org/wiki/Database">https://en.wikipedia.org/wiki/Database</a></td>
<td>0.76</td>
</tr>
<tr>
<td><a href="http://whatis.techtarget.com/definition/system-software">http://whatis.techtarget.com/definition/system-software</a></td>
<td>0.799</td>
</tr>
<tr>
<td><a href="https://www.computerhope.com/jargon/s/systsoft.htm">https://www.computerhope.com/jargon/s/systsoft.htm</a></td>
<td>0.576</td>
</tr>
<tr>
<td><a href="https://en.wikibooks.org/wiki/Introduction_to_Computer_Information_Systems/System_Software">https://en.wikibooks.org/wiki/Introduction_to_Computer_Information_Systems/System_Software</a></td>
<td>0.851</td>
</tr>
</tbody>
</table>

4.2.2 Pre-Processor

Pre-processing is necessary for removing the images, videos and other heterogenous data from the web documents. The retrieved page through crawler is parsed to extract the text separately. Stop word filtering is performed to the content to remove the meaningless words. Stemming is also done to normalize the single root word. The essential keywords are picked up from the page.

4.2.3 Seed URL based Meta-keyword Extractor

The meta-keywords are generated from the seed URLs and matched with the content of the web page. If the meaningful words from the content of the web page are existing or related to the seed URL meta-keyword, then the semantic relevancy score is calculated. The relevancy score is calculated
against ‘n’ URL with the weight of the concept with the domain ontology, the frequency of occurrence of keyword and relationship similarity score with the meta-keyword. A predefined value is fixed from the analysis of 500 URLs. The threshold value is limited to 0.8. The threshold value is limited to 0.8 for seed URL, in the literature of semantic crawler [88], the threshold value is fixed to be 0.5 and more than 0.5, with my experimental analysis it is computed with more than 500 URLs, 0.8 is concluded as the final value. Moreover, the URL extractor is based on 5 values to restrict the URL harvest propositions.

4.2.4 Semantic Relevancy Calculator and Semantic URL Extractor

The semantic relevancy calculator is executed based on the semantic similarity score of the domain ontology. The web page link is extracted based on the following URL strategies:

- **Title based extraction:** The title of the web page is extracted and matched with the domain and score is calculated based on semantic relevancy.
- **Link Extraction:** The URL link extractor comprises of a parser which grabs the HREF attributes and matches with the domain and prunes the unwanted attributes. The incoming and outgoing links on the web page are also harvested with the links.
- **Content-based extraction:** The content is extracted and matched with the domain and the weight of the content based keywords is calculated.
- **Anchor keyword based extraction:** The anchor keywords are matched with the Meta-keywords and the domain to generate relevancy scores.
- **Domain concept based extraction:** If the domain concept and the concept of the content are same, the sum of all the above weights is done to calculate the average semantic score of an URL.

The domain concept based extraction is performed by semantic relevant URL extractor (SRUe) is a factor to pull the seed URLs that are relevant to the given inputs and is given by the following formula in 4.1:
Where \( \text{SRU}_e \) is the semantic relevant URL extraction for the link \( i \),

- \( T_e \) - Title extraction and matching with the domain based semantic score.
- \( L_e \) - The number of incoming and outgoing links with semantic matching with the NLP tool score is considered
- \( C_e \) - Content and keyword based concept matching score
- \( A_e \) - Anchor words based matching with the domain specific similarity score
- \( D_e \) - Domain concept score for the document as a whole

\[
\text{SRU}_e = \frac{1}{5} \left( T_e + L_e + C_e + A_e + D_e \right)
\]  

(4.1)

\( \text{SRU}_e \) is given as input to the URL harvester for the SSMKC. The average of the scores is calculated to check the percentage of the seed URL which are given as input to the crawler. The links that get the value higher than the threshold is termed as relevant. The threshold value of each of the link is calculated by semantic analysis of ‘href’ attributes. The constraints for filtering of URL links are proposed with the following propositions.

**Proposition 1:** URLs are case insensitive to avoid duplication.

**Proposition 2:** URLs ending with .edu, .ac, .gov are avoided and restricted. The blog-based URLs such as wiki, WordPress, Tumblr and weblogs are given priority.

**Proposition 3:** URL differentiation is nullified by commonly used characters like ‘~,’ ‘*,’ ‘+.’ The fragment part is removed.

**Proposition 4:** Incoming and outgoing links inside the blog are also extracted and stored. The list of crawled URLs may have still redundancy this can be excluded by using the list of seeds from different sources like search engines, wikis and question answering systems.
4.2.5 Domain based Meta-Keyword Identifier and Structure based Filter

The domain-based meta-keywords are embedded in the HTML document itself in many websites. In few other sites, the meta-keywords have to be generated based on the query. The structure of the blogs is read by the distributed analyses platform and the content in the web blogs is stored in the repository. The domain of the URL content is considered and semantic relevancy score is calculated with the help of text content. Text content is extracted based on the relevancy score calculated for the domain. The Domain of URL is identified based on the meta-keyword comparison with the already existing seed URL. If the same page is downloaded then, based on the timestamp modification the differences will be identified. Since the website from the repository is deleted in original source. The storage of domain blogs from the identification phase is reserved for the future search and usage.

4.2.6 Algorithm for the Semantic Crawler

The crawler used in the thesis is semantic and the semantic score is calculated by customizing the libraries and applying the domain based ontology. The threshold value for the crawler is fixed to be 0.8 in the range of 0-1. The attributes are well defined in the following Algorithm 4.1 based on the seed URL.

Algorithm 4.1: Harvest Source URLs using SSMKC

Input: Seed URL obtained from URL extractor SRUe, Meta-keyword from the seed URL, and Page ranking score.
Output: Crawled relevant URLs
// Seed URL based on Meta-keyword
While (URL tobe seen> getAddrfrom Seed URL) then
If (currep>maxVal) then
CurrURL=URLtobeseen
SetURLseen (currURL)
If allows Crawl (currURL) and currURL not seen then

63
SuccPage=getPage (curURL, curVal)
If (SuccPage==NULL) then
    Log (cannot crawl URL to be Seen)
else
    SDDesc=processText(SuccPage)
    // Semantic Relevancy Calculator
    If similarity Check (SDDesc,ConceptMatch) > threshold then
        Add to rep (SDDesc,curURL)
    URLlist=extractURLs(SuccPage)
end if
free (SuccPage)
end if
// Refine the crawler with seed URL
for each URL in URLList do
    SeedURL.add(URL, metakeywordrelvancy)
    RefineSeedURL(SeedURL.metakeywordrelvancy,curVal+1)
end for
// User focused based on structure and propositions
for each URL and its content
    check the structure and content
    Mark it as successful URL link and store it in repository
end if
end if
end while.

4.3 RESULTS AND OBSERVATION FOR SSMKC

The significance of SSMKC in the technological, social, sports, news and social media domain is evaluated and illustrated in this section. The SSMKC is implemented in Java, Alchemy API and web ontology reasoner with Jena API. It is executed on Hadoop Distributed File system with a Map-Reduce paradigm in Hadoop 2.01 with a machine powered by Intel i3 processor and 2.4GHz with 4 GB RAM and 500 GB Hard disk. Table 4.2 shows statistics of the crawler.
Table 4.2 – SSMKC statistics

<table>
<thead>
<tr>
<th>Detail of links</th>
<th>SSMKC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Links parsed</td>
<td>75,543</td>
</tr>
<tr>
<td>URLs related to domain</td>
<td>39,000 (51.6%)</td>
</tr>
<tr>
<td>URLs Crawled based on structure and domain</td>
<td>9000</td>
</tr>
</tbody>
</table>

The statistics of URL’s parsed in Table 4.2. The crawler parsed 75,543 links are discovered 39,000 URL’s related to the domain specified. But these links include wikis, new, blogs and articles related to technology in computer science, IT, sports, social media, business and politics, history, social science, engineering. There are 9000 links identified are more relevant and valid.

4.3.1 Case Study for the Semantic Crawler

A brief study of the crawler is conducted by comparing the performance of the SSMKC with the traditional keyword-based crawlers and ontology-based crawlers. The seed URLs are given to the three crawlers based on the domains specified. The test is conducted for 3000 sample URLs. The keyword query given is “Big Data”. The keyword-based crawlers are very poor in identifying the domain based links using wrong naming conventions. Therefore, the crawling is accomplished by synonyms of WordNet [89]. Even then, the crawlers are not that perfect to bring in the links which are precisely required for the users. The concepts get narrower and more rooted in the ontology in the lower level so the most relevant URLs can be easily identified [90] for the standard concepts. But in the proposed thesis the structure of the blogs and also the content is read from blogs for tracking and analysis purpose. The implementation is facilitated by SSKMC.

SSKMC works along with the dynamic SCM and so it is termed as distributed SCM. In dynamic SCM, the Virtual Documents are created via Web API, whereas the source documents are pulled from Wikipedia by a simple key word, and it is transformed and stored in MongoDB (NoSQL). Similar kind of interface is used in Distributed SCM, where the source
documents are pulled by the SSKM crawler and the Virtual Documents are created similar to dynamic SCM shown in 4.4.

The significant feature is the extraction of content from particular blog type based word press which has the same structure. In the case study, it is performed with the help of PIG an analysis platform in Hadoop. The content is read and filtered and stored in the data model called Hbase in Hadoop. The extraction from the blog and storage is performed by PIG and Hbase in the Hadoop. The PIG is used to pull and read the unstructured data. The specific group of blogs has the same structure as built by the Hypertext Markup Language document object model.

The distributed repository in the DFS stores the content from the blog in the form of a comma, separated values. The parsing of the structure of HTML page has developed a script in the crawler. It also helps in identifying the meta-keywords in the web page.

Before the URLs are extracted to Hadoop, a crawler is designed to extract the URLs from different search engines. Five to ten different search engines like Google search API, wiki API, Mashable URLs from BBC API, Yahoo, Askme.com are some the search engine. The structure of the blog URL is read before extracting using Pig in Hadoop. The domain of the URL content is considered and semantic relevancy score is calculated with the help of text content. The title of the website or blog content is extracted. The crawler is used for extracting links in the blogs such as incoming and outgoing links.

The meta-keywords of seed URL are accounted for semantic analysis. Seed URLs are identified based on the domain for the crawler. In the case study, 75,543 URLs are discovered in various domains such as computer science, sports, news, electronics, medical applications and mobile computing. 39000 URLs are focused and stored by the crawler in priority basis from the total links discovered.
4.3.2 Evaluation Metrics

For performance evaluation, harvest rate is considered [91] and it is used to identify the integrity of the crawler. Since the consistent set of the domain specific links is unknown, it is hard to measure recall. Harvest rate indicates the rate at which the essential pages are crawled and the rate at which the unrelated links are taken out from the crawl.

\[
\text{Harvest Rate} = \frac{R}{N}
\]  

(4.2)

Where \( R \) is the number of relevant web pages crawled and \( N \) is the total number of web pages crawled.

High harvest rate is the sign of proper crawling. Harvest Rate indicates the rate of significant pages crawled and successfully taking out unrelated URLs. When compared to techniques and model applied in Keyword based crawler [89] the average harvest rate of is (36%), According to [90] the Ontology-based crawler is implemented and the average rate is (54%) and Semantic relevancy is 58% which is highest compared to former methods. SSMKC has improved the harvest rate and pruned the unrelated URLs based on the knowledge provided to it.

Figure 4.3: Comparison of harvest rate of different crawlers
Figure 4.3 shows the analysis of the crawler harvest rate. The threshold value is fixed to be 0.8 after several iterations performed various crawlers for the same set of documents.

The 1000 sample URLs are considered for testing purpose. The harvest rate decreases corresponding to the increasing number of URLs. There is a steady decrease in harvest rate for the URLs more than 1000 as shown in the above graph. There may be variations when the size and content of URL changes. Keyword-based crawler crawls the pages on the web for submitted keywords in the user query without considering users requirement and priority hence harvest rate is very less, whereas the semantic–relevancy crawler examines logical meaning to get relevant services. In addition to this SSMKC focusses on the user required domain and structure of the web document as the content is read and validated.

4.4 DYNAMIC SCM USING NOSQL STORAGE

The semantic user interface is part of SCMS which extracts useful information from the web [92]. Dynamic SCM has also performed the work of semantic user interface in proposed DSCMS. The detailed working of Dynamic SCM is shown in Figure 4.4. Search API pull the source URL in the web and utilize it for indexing and analysis.

The automatic extraction of the web documents through search API from CMS like the wiki is possible for semantic analysis and relevant retrieval [93]. The pulled documents are preprocessed using stemmer and parser. The Virtual Documents (VD) are created and stored in NoSQL [94] like a document-oriented database and the corresponding source documents are stored as semantically linked data with concept matching. The group of words are annotated and highlighted in the semantic interface provided by dynamic SCM. The concept matching based semantic annotation is implemented for matching the source document SD with the virtual document. The content in the virtual document are parsed using the MCCA.
Figure 4.4: Dynamic Semantic Content Marker

When the user selects the group of words/phrases in the VD then recommended URLs are retrieved from the repository it displays the source links. The precision of retrieval is improved by proposed MCCA. The dynamic content marker consists of the components such as source extractor, Virtual document creator, a linker which links both source and virtual documents.

4.4.1 Creation of Virtual Documents

VDs are newly created using already existing blogs and source URLs.

Figure 4.5: Structure of the URL storage with detail
These VDs are stored automatically with the detailed representation as shown in Figure 4.5. The stored URLs are fetched with the domain, title, content and links, time stamp 'ts' are appropriately stored in the scalable repository. Creation of VD is implemented using randomly generated sources for a particular topic automatically via search API.

4.4.2 Preprocessing of Documents

The source documents and its URL are given fed to the system. The documents are cleaned and saved in the data store. The unwanted information such as audio files, video files and internal CSS are removed. The advertisements are also removed and the following steps are implemented:

- Stemming
- Removal of stop words
- Parser is used for removing the unwanted noise in web page
- Cumulative documents using Meta tags

![Diagram](image)

Figure 4.6: Internal noise removal process

The process first checks whether SD is present. Secondly, it gets the URLs and checks whether more than one URL is present in the field. Thirdly, it parses the URLs using comma as a delimiter. In the fourth step, for each separated URL, it parses the web page into HTML document using HTML parsing tool. In the fifth step, for each HTML document, it removes all the
navigation links, images and styles. The text is filtered is used for analysis. Figure 4.6 shows the detailed steps involved in the noised removal process.

4.4.3 Algorithm for Dynamic SCM

The semantic analysis is accomplished using concept matching technique. The content in the form of the paragraph in VD is segregated, categorized and linked with the source documents based on the high-frequency keywords. This algorithm is typically used in the page ranking techniques. The step by step procedure for MCCA is shown in Algorithm 4.2.

Algorithm 4.2: MCCA for Linking and Displaying Source Documents

Input: Virtual Document (VD) and Source Document (SD)

Process: Annotating the source links with the high frequency terms

Output: Display the top ten annotated source links.

If URLs extracted from search API
{
    Check the content of the page
    If count (VD == N) {
        Initiate pre-processing {
            for ‘N’ VD {
                Remove stop words
                Performing stemming and finding root words
            }
        }
        for ‘N’ links of VD{
            Iterate the above process using parser
        }
        If (Relationship of ontology concept ∩ Source content)
        {
            Calculate similarity score for SDₙ using concept matching
        } Else If (Relationship of ontology concept ∩ VD content)
        Calculate similarity score of VDₙ
        Compare the Score (VD) & Score (SD) If (Threshold value > 0.5)
        Link the SD with VD for a specific domain.
        Sort the score of high frequency terms (VD)
Identify top ten (VD) and annotated content
Display the source links of SD for the user
}
End if
}

4.5 STATIC SCM USING RDBMS STORAGE

Static SCM system consists of five main modules. The user directly browses the content and the content is inserted via an interface to the data store. This model utilizing the concept of semantic analysis is used to retrieve a relevant document from the web [95].

![Static Semantic Content Marker](image)

Figure 4.7: Static Semantic Content Marker

The processes in static SCM starts with the user gives a query based on his requirement. The content from the differentUrls are used to create the new virtual document. Figure 4.7 shows the working of static semantic content marker system. The pre-processing techniques which are used to clean the CSS styles and unwanted information as written in the previous
section are carried out. The key entities are identified and their significance is marked. The semantic annotation is performed to highlight the key entities. The domain based concept matching is used to map the key entities with the title of the documents. When user is willing to refer the SD of new web article created in future the source documents are retrieved based on priority.

4.5.1 Entity Relationship Identification

The relation between entities is mapped among the source and newly created documents. The metadata stored in the article are tree like structure by which the relationships are established. In the proposed system, the recognition of entities and their relations are measured by similarity scores from Alchemy API.

Natural language processing and machine learning with statistics are utilized for analysis in terms of person, topics, languages, and facts. Keyword extraction, concept tagging and text extraction are the processes used in the proposed research for enriching semantic analysis performed by the NLP tools. The source code in the tool is customized for identifying entities and the metadata is stored in RDF / XML format.

4.5.2 Comparison and Location of Source Web Pages

When group of words are marked in the newly created article, the text related source documents are retrieved by matching the source documents and marked text semantically. The SD in the data store are mapped and checked for similarity. The comparison of source web pages with marked query is done by domain based concept matching and pattern matching.

4.5.3 Algorithm for Static SCM

The location of the text in the source document based on the semantic analysis is explored. The text in the marked virtual document is matched with the source document. Firstly the presence of the URL is checked and secondly the marked text concept in SD is matched, thirdly the URL of the source content with the location of matched content is returned. It is
accomplished by NER-based concept matching technique shown in Algorithm 4.3.

**Algorithm 4.3: NER based Concept Matching for Linking and Retrieving**

Input: Marked text content.
Output: SD based URL
Variables: marked text portion by the user
Begin
for every “Created Article”
    if “userinput !=NULL” then
        Compare “userinput” with all text documents in DB;
        end if
        return “SourceURL of the matched text document”;
    End for
If the returned SD based web pages loaded to interface {
Begin
If “returnedURL! = NULL” then
for every “returnedURL” do
    compare “userinput” and “text document”; 
    Yield “position of userinput in text document”; 
end for
While every “returned position” do 
    locate the corresponding to the URL;
    identify the marked text and return position in webpage;
end while

4.5.4 Result Analysis for Static and Dynamic SCMs

The proposed concept of dynamic SCM has been experimented with Wiki documents. The dynamic pulling is implemented for specific documents using Wiki API and Mongo DB. The heuristic algorithm MCCA is used to link the high-frequency words with the source content. Java and Mongo DB is used with Alchemy API and reasoner. The concept matching technique is utilized to run the code.

The time taken for retrieval through MongoDB is reduced when compared with MySQL in static SCM. The static content marker is realized by constructing web API with server side scripting in Java and open source database with Tomcat and Alchemy API. Alchemy API is a named entity
based tool which can be used for semantic analysis [94] in all the SCMs. To evaluate the performance of the Alchemy API the concept matching based MCCA is applied to three more tools in that context and analyzed [95]. The metrics considered for evaluation is precision and relevancy score.

The relevancy score and precision are estimated by the following formulas and the comparison is plotted for graphical analysis.

\[
Overall \ Precision = \frac{\text{relevant entities} \cap \text{retrieved entities}}{\text{retrieved entities}}
\]

\[\text{(4.3)}\]

\[
Relevancy \ Score = \frac{\text{relevant entities} \cap \text{retrieved entities}}{\text{relevant entities}}
\]

\[\text{(4.4)}\]

The relevancy score and precision scores of the NER tools are compared by implementing MCCA algorithm. It is evident that the Alchemy API tool shows highest recall and precision from Figure 4.8. The overall scores by a testing set of 100 documents show that Alchemy API is the best tool for performing semantic analysis with customized algorithms and statistical methods compared to other popular tools. If the number of documents is increased further the score also steadily increases as shown in the graph below.

![Figure 4.8: Comparison of NER Tools](image)
4.5.5 Comparative Study of SCMs

Extraction, Transforming and Loading of web documents are performed in all the three methods. The comparative study is conducted with various features.

Extracting the web documents from Wiki API and Google API automatically. Transforming refers to preprocessing and cleaning the documents by removing unwanted content. The pre-processed documents in text format are ingested in databases. The text documents are loaded to Hadoop Distributed File System (HDFS) based on the requirement. Table 4.3 shows the comparison of experimental setup and algorithms used for execution.

<table>
<thead>
<tr>
<th>Storage Repository</th>
<th>Static SCM</th>
<th>Dynamic SCM</th>
<th>Distributed SCM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MYSQL 5.1</td>
<td>MongoDB</td>
<td>H-Base on Hadoop 1.01</td>
</tr>
<tr>
<td>Java &amp; IDE</td>
<td>JDK 1.7 &amp; Netbeans 6.1</td>
<td>JDK 1.7 &amp; Eclipse-Kepler</td>
<td>JDK 1.7 &amp; Eclipse–Kepler</td>
</tr>
<tr>
<td>NLP Tool</td>
<td>Alchemy API 2013</td>
<td>Alchemy API 2013</td>
<td>Alchemy API 2014</td>
</tr>
<tr>
<td>Algorithms Implemented</td>
<td>NER based concept matching</td>
<td>MCCA based on concept matching</td>
<td>Semantic similarity using concept matching</td>
</tr>
</tbody>
</table>

Various experiments are performed in all the data storage based SCM using different semantic techniques such as semantic annotation and concept matching [96]. The time taken for execution of all the three SCMs are calculated and shown in the Figure 4.9.
The number of documents is increased from 100 to 1000 steadily and the time taken for each method is calculated. The time is taken for retrieval using distributed SCM is the least when compared to dynamic and static SCM. Time is directly proportional to the number of documents, in case of static SCM. Whereas, in dynamic SCM the time taken is lesser when compared to static due to scalable and sharding property of data model. But in the distributed SCM the time taken for the document retrieval is saturated and quick. It is achieved by the parallel search and retrieval of web documents with multi data nodes in a distributed environment. It is also enabled by storing the logic in cache blocks of the data node. When the new documents are accumulated, it takes very less time just greater or within the time taken for previous search and retrieval. The metrics for evaluation of SCMs are analyzed based on Recall and Precision. The relevancy score and overall accuracy is calculated by the following formulas shown in 4.5 & 4.6.

\[
\text{Relavancy Score} = \frac{\text{relevant documents} \cap \text{retrieved documents}}{\text{relevant documents}} \tag{4.5}
\]

\[
\text{Overall Precision} = \frac{\text{relevant documents} \cap \text{retrieved documents}}{\text{retrieved documents}} \tag{4.6}
\]

Figure 4.9: Comparison of time for all the SCMs
The relevant document retrieval is accomplished by the recall and precision scores. The three types of SCMs are compared for the former and later scores as shown in the Figure 4.10. These scores are calculated for 10 to 100 set of documents and the average score is considered as the overall score in the above graph, this overall score is best in the case of distributed SCM when compared to static and dynamic SCMs. It is succeeded by implementing the algorithm which uses refined concept matching method. The method used here is more semantic and user-focused. In the case of static and dynamic SCM, preliminary algorithms are used which gives reduced overall score. Moreover the overall precision decreases when increases number of documents.
4.6 SUMMARY

- In this chapter, design and deployment of SCM with different data models are performed. It ensures data extraction, transformation and loading of web documents such as virtual and source documents to distributed file systems.
- The algorithm for user focused semantic crawler SSMKC is developed. The evaluation and comparison of the crawler with existing ones are illustrated.
- Static and dynamic content markers with concept matching algorithms such as MCCA and NER based concept matching are implemented and evaluated.
- It is demonstrated and proved that the best semantic NLP tool is Alchemy API which is more suitable for semantic analysis when compared with the existing tools.
- The experimental results of the different SCMs prove that concept matching method is more accurate, distributed and user focused in the case of distributed SCM when compared to static and dynamic SCMs.
- The performance analysis is also conducted with respect to time of searching and retrieval for all the three SCMs. In the case of distributed SCM, the map reduce paradigm model implements search and retrieval simultaneously. Hence the time taken is least when compared others.
- The qualitative and quantitative comparisons are accomplished for evaluating all the three SCMs. Both comparisons proved that distributed SCM with semantic user interface is best when compared to other designs.

The Chapter 5 presents the design and development of rendering engine which comprises of combination of semantic clustering and classification techniques in the distributed file system.