6. HADOOP BASED RECOMMENDER SYSTEM WITH INBUILT CHANGE TRACKER

6.1 INTRODUCTION

In the proposed system, to access the vastly increasing resource more efficiently, a combination of clustering and classification method provides fast results. These techniques allow retrieving of the massive scale of data efficiently using Resource Description Framework (RDF). The RDF is usually a semantic description of the specific metadata in the web. If the retrieved web documents and URLs are marked and frequently visited by the user, then there is a need for change tracker system which detects the changes dynamically [8]. The changes in the source documents are identified automatically by analyzing and updating the RDF document. Moreover, the update to the entire dataset for every change in the resource is complicated and too time-consuming. To solve issues like complexity and time consumption Hadoop based recommender system [120] is proposed. The HBSRS is enhanced using fourfold similarity method and change tracking is semantically implemented with the help of RDF searching and Automatic Breadth-First Search (ABFS).

6.1.1 Recommender system based on enhanced semantic IR

The conventional information retrieval system requires a recommendation approach for selecting the desired result. It is essential for the collaborative web to possess a semantic IR which supports it in many aspects like interaction and analysis. The semantic analysis is performed using web ontology language or RDF. The metadata from RDF is reused and also recommended to the users [121]. <Subject, Predict, Object> are triad which describes the information. It also provides essential relationships and reasoning ability. The ontology layer in the semantic web is improvised by the RDF. The limitations in existing RDF based system are listed below:

- Reasoning over the entire dataset at every update is too time-consuming. Processing time is very high.
- No accurate indexing and summarization for a particular keyword search result.
- There is no inbuilt change detector system in the existing Framework.

6.2 WORK FLOW PROCESS OF HADOOP BASED SEMANTIC RECOMMENDATION SYSTEM (HBSRS)

![Diagram of Hadoop Based Semantic Recommendation System (HBSRS)](image)

Figure 6.1: Hadoop Based Semantic Recommendation System (HBSRS)

The components of HBSRS include two sub-phases such as Inbuilt Change Tracker System (ICTS) and Dynamic Streaming Based Change Detection System (DSBCS). The distributed SCM as mentioned in Chapter 4, are extended with additional functionalities which supports the ICTS and DSBCS. It consists of an RDF query builder, a SPARQL engine, collaborative filter and a change detector. The processing of HBSRS is performed by utilizing the components mentioned above in two stages.

The proposed system consists of two processing stages such as offline processing stage and query processing stage. The workflow of Hadoop based semantic recommendation system (HBSRS) is shown in Figure 6.1. In offline processing stage, the resources are collected from
multiple blogs of the same structure as crawled by SSMKC. From the collected resources RDF dataset is built for different domains. Then these collections are stored in NoSQL [123] database and indexed to generate summaries automatically. Summaries can be quickly updated when the dynamic streaming script triggers an alert to the system. The alerts indicate the dynamic changes either in content or the structure of the web pages.

In query processing stage, the user interacts directly with the system using the set of keywords. When the user provides the keyword, it is automatically parsed using the existing parsing methods and search queries are generated to extract the results from the summaries. Similarity finder and filtering methods are used to provide the acclaimed list of results to the user. Categorization of the documents to specific domain similarity computation methods such as Jaccard and cosine are used. This similarity methods act as the integrated input to semantic clustering algorithm as indicated in Algorithm 5.3. Algorithm 6.1 shows Semantic Similarity Score (SSS) cluster identification.

Algorithm 6.1: Computation of SSS for Identification of Clusters

Input: SD – Incoming web links
Output: Average score to allot to specific cluster
While (SD1!=null){
    DM1= Entitles of (SD1) → SD1-extraction
    for(i=0;i<DT-1;i++){ DM2= getDomainKeywords(DTk); → get all keywords for the domain DTk
        Davgscore=Compare high frequency key terms(DM1, DM2); → Compare keywords in DM1 and DM2
        Obtain the SS[DM1,DM2]
        DM3.Store(DTk,MeanScore) → Store the domain topic and its similarity score }
    Return maxSimScore in (DM3) → Return the max score with Domain topic
    Cluster the source document belong to that domain DTk.

6.2.1 RDF Building and Summarization

RDF building is performed after the data preprocessing. It involves in generating the sequence keywords using the Algorithm 5.5. Finite set of
keywords acts as the input to the Algorithm 6.2 that determines the top 25 keywords periodically used to in every domain.

**Algorithm 6.2: Sequence Keyword Semantic Score (SKSS)**

Input: Limited number of key terms for testing purpose (DM1)
Output: Set of limited Sequence-terms

While(DM1!=null){
    DM2= ObtainDomainTopicTerms(DTk); \( \rightarrow \) extract all the key terms for the specific domain DT
    Compute SS distance value using ontology {
        SSDV \((n1,n2)=\log DM1(n1)/DM1(kp) + DM2(kp)\)
        DM2.Save (k1,SSDV) \( \rightarrow \) Save the topic of the domain with SS value
    }
    Sort (DM3) \( \rightarrow \) Sort DM3 according to SSDV
    return top 25 keywords from DM3 as sequence or co-keyword

RDF data is build using the Jena API with the source URL as subject, its relationship as predicate and abstract source content as literal. Later, RDF data set is used for constructing collections and matching similarity in terms of vertices, subject, predicate, objects and literals. The protégé is used to generate directional graph set for RDF document. Summarization is performed based on frequency of occurrences of keywords and identification of sequence-keywords. It also helps to index the documents based on specific domains effectively.

**6.2.2 Collaborative Filtering (CF)**

The CF is performed by matching the most similar source document got from the recommendation system dependent on the diverse approaches which are initiated by the users. It depends upon the following strategies:

- Significant rank based search depending on number of successful hits.
- The time span that users have visited the source web page.
- Historical user search depending upon the choice of the user for the same kind of query. Historical search repository.
- Fusion search which is the blend of all the kinds of search.
Algorithm 6.3: User Centric Blog Crawling (UCBC)

Input: Choice based result of current active user for specific keyword (AUSR)
Choice based result of historical user for specific keyword (HUSR)
Choice based result based on number of successful hits for specific
Keyword (SHR)
Output: set of relevant of source web documents for specific keyword
Simucbc (AUSR, HUSR, SHR) = |CUSR ∩ HUSR ∩ SHR| ÷ |CUSR ∪ HUSR ∪ SHR|
Return similar source web documents as recommended source documents.

Algorithm 6.2 and Algorithm 6.3 is utilized for summarization and
filtering of the source URLs rendered from RESCC. The filtering process is
enhanced in HBSRS compared to the previous algorithms in terms of
personalization.

6.2.3 Change Tracker Phase

Change tracker [124] traces each source URL on the web and
updates the status periodically. If there is any change tracked to the specific
source web document, it is systematically bookmarked and triples are
segregated for the specific RDF document. The updating of RDF document is
done without altering the whole corpus. Dynamic streamer is used to find the
changes based on scripts which are customized in the change log streamers.
The process for change tracker is shown in the Algorithm 6.4.

Algorithm 6.4: Inbuilt Change Tracking

Input: Source URLs, RDF files and Triples
Output: Backtracked URL's, Alerts through E-Mails
Dynamic streaming algorithm is implemented in DFS
The source URL that has to be monitored is saved as log in the dynamic
streamer.
RDF triples are analyzed. If there is a change in the literal which is identified
by k-depth decision based on conditions then the modification is alerted by a
trigger. K value is iterated, k-depth decisions are taken based on the
conditions such as the filtered URLs are logged in flume and those URLs are
monitored.
Creation of Triples for Source URLs and Comparison of Literals in both
URLs, old and new URLs.
If both are not same then decide based on conditional probability and that the URL change is detected along with the content. The time taken is mentioned and accumulated log data is tested periodically.

### 6.2.4 Alerting the User with Current Status

The dynamic streamer with the change tracker is used to detect the changes in the content of the URL and alert the user. The user is notified by triggering the e-mail for the changed sources in the old web document. The consolidated report is periodically sent to the users. The workflow of Alert module is shown in Figure 6.2.

![Workflow of Alert module](image)

**Figure 6.2: Workflow of Alert module**

The work flow of dynamic streamer is shown in the figure 6.2 a. The dynamic streamer triggers the old source document application and the data source and sink of the dynamic streamer commits its transaction through the channels and stores the output to the HDFS. This kind of streamer is used to check the change log using timestamp, whereas in the proposed research novelty is dynamic streamer is used find the changes stored in the triples via the data source and sink along with the timestamp. The notification header consist of the changes in object, changes in time stamp, old content in source web document whenever the change tracking is triggered by the user [125] to the source. The changes in the content of the source URLs are tracked by means of ABFS algorithm shown in Algorithm 6.5. The RDF is
segregated in to components and stored separately. The literals and vertices are searched and mapped [126].

![Figure 6.2a: Workflow of Dynamic Streamer](image)

**Algorithm 6.5: Automatic Breadth First Search (ABFS) for identifying the content change**

//Check the triple sets for the filtered URLs
function Search (u, Af )

init ← (m, q)
init.parent ← null

if q=0, q ∈ Fe then return initial value

Initialize an empty Queue (BFS)
    Initialize Seen as an empty set
    Insert init into both Open and Seen
    While Open is not empty do
        Pull node S = (n, q) from Neighbors
        for each t = (n0, q0) in Neighbors(s) that is not in Seen do
            t.parent ← s
            if q0 ∈ Fe then return t or add t to solutions
        //Insert t into both Open and Seen
        Method Neighbors ((n, q))
        Initialize Succ as an empty set and RDF graph Gtemp as an empty graph
        Rtemp ← adhoc (v)
        for each IRI a ∈ I and state q0
            s.t. (q, a, q0) is in δw do
                for each triple (v, a, v0) in Rtemp do Insert (q0, v0) into Succ
//Collect the triple set with same author or subject and date based on relationship score.
for each IR a− ∈ I−and state q0
s.t. (q, a−, q0) is in δe do for each triple (v0, a, v) in Gtemp do Insert (q0, v0) in to Succ, return Succ.

6.3 RESULTS AND DISCUSSION

To evaluate the HBSRS with KASR [63], the experimental setup with 9000 URLs and 10 clusters which are generated and optimized in chapter 5 are considered. Figure 6.3 shows the extraction of blog content using Pig with the help of Hadoop and it is stored in the form of comma separated values in Mongo DB and processing with clustering and classification. The system provides the efficiency similar to the conventional systems but if the size exceeds above 100 to 1000 MB the traditional system slags and requires excess time to analyze and process the data.

The proposed HBSRS is best suited for handling huge number of dataset, whereas the existing system requires more handling time and the querying engine is utilized for saturating the time.

Figure 6.3: Workflow of Tracker module
The change detection and identification of the content change is compared with the existing literature [64]. The Table 6.1 shows the comparison and analysis of identification of changes in URLs ranging from 100 to 500 taken for testing purpose for the regular interval of five 24 hours.

From the Table 6.1, it is concluded that there is 6% to 32% increase in percentage of detection compared to existing method. This is due to the identification of semantic change in the content using ABFS dynamically. When the number of documents increases the percentage of identification may increase or may not change.

Table 6.1 - Change detection rate using the proposed and existing algorithms

<table>
<thead>
<tr>
<th>Algorithms used</th>
<th>Total number of URLs</th>
<th>Number of changes detected</th>
<th>% of detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change detection using focused crawl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>30</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>95</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>173</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>238</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>275</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>Proposed ICTS with dynamic streamer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>60</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>142</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>218</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>247</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>305</td>
<td>61</td>
<td></td>
</tr>
</tbody>
</table>

It depends upon the size of the URL and number of changes made in the URL and its content for a particular period. Exquisitely, change detection depends upon the size of the URL and the changes made in the URL content and objects.

The proposed Algorithm 6.4 shows a linear increase or spikes in the curve depending upon the nature of source documents. The nature of document is determined by the time stamp change, change in literal, vertices and then content change of the old URL.
Evaluation of the recommendation system which is a significant component of DSCMS is performed by microanalysis which is implemented for 750 URLs and evaluated using the existing system. Keyword Aware Service Recommender (KASR) retrieves 138 source web documents for sample query “Cloud Computing Services” unlike the HBSRS system using UCBC algorithm gives only 50 URLs to the uses out of 750 URLs. The count of the number of dynamic websites is maintained in the change tracker. The system is designed in such a way that only 150 changes are saved. If the counter crosses 150, the metadata created for the 150 source web document files is flushed. The execution of the algorithm is implemented both in a single node, two nodes and three nodes. The performance is evaluated by measuring the time for the following categories:

- Triples searching time
- Uploading of triple time
- Source document retrieval time results are exhibited in Table 6.2.

Table 6.2 shows that the results are being executed in single and multi-node environments. It is an evident fact that the proposed HBSR system proved to be best when matched with the existing system. The querying engine used in HBSRS avoids the overhead of the existing change detector.

Table 6.2 - Time comparison of Single Node vs Multi-node in DFS

<table>
<thead>
<tr>
<th></th>
<th>KASR</th>
<th>HBSRS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single Node</td>
<td>Multi Node</td>
</tr>
<tr>
<td>Upload Triples</td>
<td>6.51 seconds</td>
<td>4.58 seconds</td>
</tr>
<tr>
<td>Search Document</td>
<td>3.87 seconds</td>
<td>1.73 seconds</td>
</tr>
<tr>
<td>Retrieve Document</td>
<td>1.38 seconds</td>
<td>1.32 seconds</td>
</tr>
</tbody>
</table>

The constructed summaries are updated and performed with the least effort which is complex in the traditional systems which consumes more rebuilding time. The 10 number of queries are selected for testing purpose as
shown in Figure 6.4. The query is executed both for HBSR and KASR. The graph shows there is a constant increase in pruning unrelated URLs.

![Figure 6.4: Comparison of KASR Vs HBSRS](image)

The triple loading time in the conventional KASR system and HBSRS increases exponentially. The loading time of the triples is up to 48 seconds for 3000 triples whereas in multi-node the time is decreased 34 seconds. There is approximately 18% decrease in the triples uploading time.

![Figure 6.5: Triples uploading time vs number of Triples](image)
The loading and processing of triples are handled by the distributed framework. The map reduce paradigm executes the logic for handling the triples simultaneously. So that the large number of triples are loaded and processed quickly. In the case of multi-node environment, a compression script is introduced to reduce the shift time from mapper to reducer.

![Image](image.png)

Figure 6.6: Triples uploading time vs. Size of the dataset

Figure 6.5 and Figure 6.6 shows the graph between the size of the triples and time of the triple upload.

The system provides the same performance like the existing system but when the size that crosses above 100MB the existing system takes more time to process the data than conventional systems. When the number of documents increases above 3000, then the curve raises exponentially. The uploading time increases drastically if the number of words in the content of the document increases rapidly. There is also a steady increase in time exponentially when there is a drastic increase in number of triples generated.

**6.4 MACRO ANALYSIS FOR HBSRS**

The micro analysis is performed using HBSRS with 750 URLs and the time taken for handling triples are illustrated in the previous section. The model is evaluated with crawled 39000 URLs and filtered 9000 URLs by SSKMC. Macro analysis is conducted with 12500 URLs by adding 3500
crawled specific related URLs in the system. The system shows the semantic source web document retrieval is better when compared to KASR system. The four fold similarity combines the semantic analysis of lexical similarities with the domain ontology similarity and increase the accuracy level by pruning the number of unrelated documents.

6.4.1 Case Study – E-Learning System

The case study explained here is based on the recommendation system which combines the previous modules like semantic clustering and semantic classification. The given case study and the work flow of the semantic Hadoop Based Semantic Recommendation System (HBSRS) is shown in Figure 6.1. In the proposed study, the content extracted from web is indexed and stored in Mongo DB. The indexed storage is analyzed with the help of a knowledge base which is enabled by distributed clusters. The clusters are optimized for the number of URLs considered. 10 clusters are fixed for the domains such as medical electronics, big data, cloud computing, sports, politics, mobile computing, security, business and electronics. Existing system offers approaches like keyword search with automatic depth decisions. The depth decisions are finalized using the relational and semantic similarities. These depth decisions are incorporated in the RDF graph set searches in proposed work. It requires unsupervised techniques such as clustering and classification. These techniques allow retrieving of large scale of data efficiently using RDF documents.

The RDF is normally semantic description of the specific metadata in the web. There is no inbuilt change tracker system which tracks the changes dynamically in the changing content. Moreover the update to the entire dataset for every change in the resource is difficult and too time consuming.

To solve issues like complexity and time consumption hadoop based recommender system is proposed which uses user centric crawling algorithms. In the proposed system, the recommendation engine is enhanced by means of fourfold similarity method. The inbuilt change tracking is semantically implemented with the help of RDF searching and ABFS. The
precision of the proposed system is improved by rate from 16% to 49% with existing keyword based search when conducted for 25 input queries categorized under simple, moderate and complex levels.

The macro analysis is conducted for 25 user queries and 12500 URLs. The comparison in retrieval of documents using HBSRS and KASR are shown in the Figure 6.7. The total number of documents in mixed proportion is loaded in the repository for testing purpose. For example, the total number of documents for the query “cloud bursting” is 500 but KASR system retrieves 610 URLs, instead HBSRS retrieves 510 URLs. Therefore it is concluded that KASR is 78% accurate instead the HBSRS is 98% accurate. Comparative analysis proved that the true positives retrieved by HBSRS is more, whereas the false positives are more in the case of KASR method. The results show the DSCMS with the HBSRS is proved to best both in micro analysis and macro analysis.

![Figure 6.7: Macro analysis for HBSRS vs KASR](image)

The Table 6.3 give details about number of users, queries and the number of source documents retrieved from KASR and HBSRS. The macro analysis is performed using apache spark in hadoop 2.0 with Java API. The visual analysis of the retrieval of documents are shown in Figure 6.7
<table>
<thead>
<tr>
<th>Number of Users</th>
<th>Queries (Q1-Q25)</th>
<th>KASRS</th>
<th>HBSRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>Cloud Bursting in Virtual Machines</td>
<td>610</td>
<td>510</td>
</tr>
<tr>
<td>User 2</td>
<td>Typical deployments require SDN and Cloud</td>
<td>894</td>
<td>610</td>
</tr>
<tr>
<td>User 3</td>
<td>service-oriented architecture style of software design</td>
<td>1085</td>
<td>690</td>
</tr>
<tr>
<td>User 4</td>
<td>Wireless electronic devices and health</td>
<td>1886</td>
<td>913</td>
</tr>
<tr>
<td>User 6</td>
<td>Distributed systems are groups of networked computers</td>
<td>2167</td>
<td>880</td>
</tr>
<tr>
<td>User 7</td>
<td>Internet is the global system of interconnected computer networks</td>
<td>2288</td>
<td>910</td>
</tr>
<tr>
<td>User 8</td>
<td>hardware support for database management</td>
<td>2329</td>
<td>1120</td>
</tr>
<tr>
<td>User 9</td>
<td>international cricketers based on their recent performances</td>
<td>2372</td>
<td>1518</td>
</tr>
<tr>
<td>User 10</td>
<td>Information Security is the process of protecting</td>
<td>2710</td>
<td>1529</td>
</tr>
<tr>
<td>User 11</td>
<td>Federation International Football Association</td>
<td>2845</td>
<td>1501</td>
</tr>
<tr>
<td>User 12</td>
<td>Institute of Electrical and Electronics Engineers</td>
<td>2938</td>
<td>1510</td>
</tr>
<tr>
<td>User 13</td>
<td>Version of an electronic mailing list or newsgroup</td>
<td>2959</td>
<td>1518</td>
</tr>
<tr>
<td>User 14</td>
<td>software and hardware on the mobile devices</td>
<td>3216</td>
<td>1552</td>
</tr>
<tr>
<td>User 15</td>
<td>global system of interconnected computer networks</td>
<td>3346</td>
<td>1632</td>
</tr>
<tr>
<td>User 16</td>
<td>computer hardware, shared memory refers to a block</td>
<td>3340</td>
<td>1827</td>
</tr>
<tr>
<td>User 17</td>
<td>IT components the foundation of IT service</td>
<td>3383</td>
<td>1825</td>
</tr>
<tr>
<td>Number of Users</td>
<td>Queries (Q1-Q25)</td>
<td>KASRS</td>
<td>HBSRS</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>User 18</td>
<td>massively distributed databases with high partition tolerance</td>
<td>3382</td>
<td>1820</td>
</tr>
<tr>
<td>User 19</td>
<td>Smartphones, handheld mobile devices</td>
<td>3491</td>
<td>2064</td>
</tr>
<tr>
<td>User 20</td>
<td>Databases have been largely secured against hackers</td>
<td>3997</td>
<td>2163</td>
</tr>
<tr>
<td>User 21</td>
<td>Cloud computing is an information technology (IT) paradigm.</td>
<td>4054</td>
<td>2154</td>
</tr>
<tr>
<td>User 22</td>
<td>Distributed computing is a field of computer science</td>
<td>4122</td>
<td>2509</td>
</tr>
<tr>
<td>User 23</td>
<td>cloud to refer platforms for distributed computing</td>
<td>4295</td>
<td>2513</td>
</tr>
<tr>
<td>User 24</td>
<td>computer science, distributed memory refers to multiprocessor</td>
<td>4480</td>
<td>2554</td>
</tr>
<tr>
<td>User 25</td>
<td>all modern computers is the Von Neumann architecture</td>
<td>5001</td>
<td>3201</td>
</tr>
</tbody>
</table>
6.5 SUMMARY

- In this chapter, design and deployment of HBSRS which recommends the personalized retrieval of web documents by RDF summarization and collaborative filtering technique is presented in distributed framework.

- Inbuilt change tracker is developed along with the HBSRS. It detects the content based changes with the help of DSBC scripts and ABFS algorithms.

- The HBSRS is also deployed in single node and multi node distributed framework. The time taken for searching, uploading and retrieval shows that efficiency is improved in the multi-node environment.

- The HBSRS plays a significant role in coordinating the work of all the components of DSCMS model. To ensure the same, the micro analysis of evaluation with 750 URLs is performed for the queries given by the user. Comparison with conventional key-word based model shows that the accuracy is 40% greater in the proposed system.

- The overall macro analysis is performed by considering 12500 URLs and 25 users. The macro analysis is visualized by comparing with the existing key-word based recommendation systems.

The chapter 7 presents query processing and retrieval. This chapter showcases the effectiveness of DSCMS model by taking the web documents from the stored repository.