Chapter – VIII

Fast and Enhanced Context Based Information Retrieval in Big Data using Hadoop and Spark
CHAPTER 8

FAST AND ENHANCED CONTEXT BASED INFORMATION RETRIEVAL IN BIG DATA USING HADOOP AND SPARK

8.1. Introduction

Involving context in any text retrieval process requires the addition of semantic knowledge into the application. Knowledge incorporated applications tend to be process intensive. Further, this process needs to be applied on huge data hence our application uses Hadoop for data storage and Spark to perform in memory processing to provide faster analysis.

The process of context based information retrieval is carried out in three major phases; TF-IDF based word ranking, context based feature vector creation and classifier training. Figure 8.1 shows the sequence of process to be carried out in performing effective information retrieval process.

![Diagram of Context Based Information Retrieval in Big Data](image)

Figure 8.1: Context Based Information Retrieval in Big Data
8.2. TF-IDF based Word Ranking

Term Frequency / Inverted Document Frequency (TF-IDF) is a numerical statistic that provides the importance of a particular word in a text document in a huge collection of documents. It is used as a weighing factor that determines the importance of the word. The advantage of this method is that it is not just frequency based, instead this method weights a word on the basis of its frequency compared with its frequency in the documents contained in the corpus. This helps isolate common words from words of importance.

\[
tfidf(t, d, D) = tf(t, d) \times idf(t, D) \quad -----(8.1)
\]

Term Frequency (TF) and the Inverted Document Frequency (IDF) are calculated using (8.1) and (2) [YAT, 99], [SAL, 98].

\[
tf(t, d) = \frac{f(t, d)}{count(w, d)} \quad -----(8.2)
\]

where \(f(t,d)\) refers to the number of times the word \(t\) is present in the document \(d\) and \(count(w,d)\) refers to the number of words in the document \(d\).

\[
idf(t, D) = \frac{\log N}{\|d \in D : t \in d\|} \quad -----(8.3)
\]

where, \(N\) is the total number of documents in the corpus, and \(\|d \in D : t \in d\|\) is the number of documents where the word \(t\) is present. If the term is not in the corpus, then it will lead to a divide-by-zero error, hence it is also common to adjust the denominator to \(1 + \|d \in D : t \in d\|\).

The words falling below the required ranks are eliminated and the pruned text is taken to the next phase for creating the feature vectors. This phase eliminates all the common words, feature vectors being matrices tends to get large as the number of
words in the text increase. This phase helps in maintaining a check on the size of the analysis vector.

**8.3. Context based Feature Vector Creation**

The next phase of processing is the creation of the feature vectors. A feature vector is a matrix that contains a word, its frequency of occurrence and other information such as its semantic meaning, polarity etc. Semantics of a text play a major role in identifying the meaning hidden in a sentence. Word based vector creation leads to more emphasis being applied on words rather than the actual meaning of the sentence leading to noisy data being added to the analysis vectors. This will in turn a lot of unassociated results being added to the result set. Inclusion of semantics in this phase deals with the sentence on the basis of its meaning leading to creation of feature vectors that are more associated with the meaning of the text under consideration.

After the creation of feature vectors, semantically similar records are merged replacing the actual item with its semantic counterpart and increasing its importance level. The final vector obtained after this phase contains the minimal number of vectors representing the document semantically. Similar process is carried out on the corpus containing training documents. These feature vectors are stored in Hadoop Distributed File System (HDFS).

**8.4. Classifier Training in Hadoop**

The training and testing phase of this application is performed on Hadoop. Classification is performed using parallelized Naïve Bayes Classifier (NBC) programmed on Spark, so as to perform in-memory processing in Hadoop for faster results. Spark is a general engine for large-scale data processing. The advantage of
using Spark is that the programs written in Spark can run up to 100x faster than Hadoop MapReduce in memory. NBC is a combination of a supervised learning method and a classifier. Due to its embarrassingly parallel nature and its ability to handle data sets of mixed types had influenced our decision to choose NBC.

According to NBC, prediction is performed as follows

$$C^{predict} = \arg \max_c P(C = c) \prod_{j=1}^{m} P(X_j = u_j | C = c) \quad (8.4)$$

Where $C^{predict} \in \{C_1, C_2, ..., C_k\}$ for a given $x$ and $X = (X_1 = u_1, ..., X_m = u_m)$ is the instance of the data to be classified.

A MapReduce implementation of NBC was used with parallelized code to operate on Resilient Distributed Dataset (RDD) on Spark. Both training and the testing components were implemented on Spark environment. Hence a reduction in terms of both training and testing times were observed.

8.5. Conclusion

A Hadoop based Big Data architecture to perform information retrieval in real time is presented in this paper. The advantage of this method is that it can analyze the entire data corpus in fast and effective manner to perform efficient retrieval of information. The initial process of this architecture plays a role in ranking data, while the semantic analysis is carried out in the next phase. These two methods help in reducing the data and in associating the semantic components with the data. The retrieval process is carried out using the NBC implemented in the MapReduce framework incorporated with Spark for faster data processing.