Chapter – IX

Real-Time context based Information Retrieval using Semantic and Features on Streaming Content
CHAPTER 9

REAL-TIME CONTEXT BASED INFORMATION RETRIEVAL USING SEMANTIC AND FEATURES ON STREAMING CONTENT

9.1. Introduction

Context based retrieval of information has become the major need for the current information retrieval systems. The major reason for this requirement is that analyzing the content might not necessarily provide the required results. This leads to inappropriate content presented to the user. However, analyzing the text based on the context rather than the content can provide effective correlations and hence better semantically close content [THA, 14]. The proposed information retrieval architecture is presented in figure 9.1.

![Figure 9.1: Context based Information Retrieval Architecture](image-url)
Input query is presented to the query creator, which creates queries based on the data sources. The queries are applied and the corresponding documents are retrieved from the data sources. These contain text, images, multimedia content, XML and HTML content. All textual contents are considered directly and the multimedia data are analyzed with the help of the metadata associated with them. This corresponds to the retrieved documents.

9.2. Document Preprocessing

The retrieved documents contain raw textual data. The data is considered to be in its raw format, because it is laden with several tokens and characters that are not required for the current processing architecture. The data pre-processing phase operates on these impurities to provide cleaned usable data that can be used in the actual processing module. The data pre-processing phase is divided into three sections; tokenization, stemming and normalization.

Tokenization is the process of dividing the data into distinct entities called tokens. This is a heuristic process, performed by setting the splitting constraints, which includes symbols and white spaces. Output from this phase corresponds to individual entities representing tokens.

Stemming is the process of identifying affixed words and eliminating the affixes to provide the root word. The initial stemming algorithm was proposed by Lovins in [REM, 15]. Next predecessor is the Porter Stemmer algorithm [ILY, 04]. This is one of the most widely used stemming techniques. The second version of Porter Stemmer is updated and maintained [DUZ, 07]. This work parallelizes Porter Stemmer 2 and is used as the stemmer. The tokens are first analyzed individually. If
the token is identified as a stop-word, it is eliminated from the list, otherwise it is passed to the stemmer for identifying the root word. Stop word identification is carried out using the WordNet 3.0 repository [SAL, 13].

Normalization is the final phase of the pre-processing module. Stemmer operates on the basis of regular expressions. Hence it strips off any token that matches the patterns, leading to ripping off several proper words. Further, data input by human users tend to contain missed spellings and inflections. Relating semantics to such words is not possible. Hence it necessitates normalization. Normalization compares the tokens with a word lexicon to provide its corrected or synonymous form of the word.

9.3. Feature Vector Creation

A feature set is created using all the shortlisted tokens as the components. Each document is composed of a feature vector. The feature vectors are combined to form a feature matrix. This creates an \( n \times m \) matrix, where \( n \) refers to the number of documents and \( m \) refers to the components of the document.

\[
\begin{align*}
   a_{ij} = \begin{cases} 
   0 \quad \text{if word}_j \not\in \text{review}_i \\
   \text{count}(\text{word}_j) \quad \text{if word}_j \in \text{review}_i
   \end{cases}
\end{align*}
\]

The feature matrix is created with all the reviews components/ tokens. If the token is contained in the document, then the number of occurrence of the component is added to the matrix, otherwise 0 is added and the feature matrix is created. The feature matrix is passed to the next phase for filtering.
9.4. Query based Vector Filtering

Semantic based analysis and filtering is carried out in this phase on the basis of the input query. Semantic based analysis is carried out by first identifying the polarity of the input query and then matching it with the polarity of the results to eliminate semantically uncorrelated results. Semantic analysis is carried out using the SentiWordNet 3.0 repository [BAL, 15]. The human annotated nature of the dataset makes it more reliable for processing. Polarity of documents are calculated by identifying individual polarity of the components and aggregating them to identify the final polarity.

\[
Polarity_d = \sum_{i=1}^{n} a_{id} \left( Polarity_{(pos,i)} - Polarity_{(neg,i)} \right) \quad ------ (9.3)
\]

Where \( n \) is the number of tokens in the document \( Polarity_{(pos,i)} \) refers to the positive polarity associated with the term \( i \) and \( Polarity_{(neg,i)} \) refers to the negative polarity associated with the term \( i \) and \( a_{id} \) refers to the number of occurrence of the token \( i \) in the document \( d \).

Polarity identification is carried out for the query as well as the retrieved documents. All the documents matching the polarity of the query are shortlisted for the next phase. This phase is carried out to match the semantic nature of the query with the result sets. Documents are retrieved on the basis of content similarity, providing a first level filtration, this phase acts as a second level filtration leading to results with high semantic correlation.

9.5. Vector Ranking and Content Delivery

Polarity identified in the previous module not only corresponds to the magnitude, but also the intensity of the polarity level in the document. Rank of a
document is identified by finding the absolute difference of the polarity of the vector with the polarity level of the query.

\[
Rank_d = \text{abs}(\text{Polarity}_d - \text{Polarity}_{\text{Query}}) \quad \text{-------- (9.4)}
\]

Vectors that correspond to the lowest difference are considered to have higher correlation with the input query. Vectors are ranked on the basis of their difference and presented to the user.

9.6. Conclusion

This chapter presents an effective real-time processing technique that can be used to retrieve information effectively from heterogeneous data. Spark based implementations are used to guarantee faster processing. Efficiency of the proposed technique can be observed from the plots. However, the proposed technique exhibits low TNR levels. Future extensions of this technique will be extended to address this issue. Further, it was also observed that the proposed technique does not handle sarcasm. This is attributed to the low TNR levels exhibited by the architecture. Future extensions of this architecture will be programmed to identify sarcasm to make the sentiment identification module more robust.