**Text Mining and Information Retrieval using SVD in LSI (Latent Semantic Indexing)**

**Course : Applied Linear Algebra (MAT 248)**

**Group no. 18**

| AU2140080 | Patel Divykumar |
| --- | --- |
| AU2140057 | Sanket Shah |
| AU2140166 | Nisarg Patel |
| AU2140141 | Krutharth Trivedi |
| AU2140220 | Shubham Apat |

**Introduction**

**Text Mining :**

Text mining, also known as text data mining, is the process of transforming unstructured text into a structured format to identify meaningful patterns and new insights.Text is one of the most common data types within databases. Depending on the database, this data can be organized as:

**Information Retrieval (IR)**

Information retrieval (IR) returns relevant information or documents based on a predefined set of queries or phrases. IR systems utilize algorithms to track user behaviors and identify relevant data. Information retrieval is commonly used in library catalog systems and popular search engines, like Google.

**Latent Semantic Indexing (LSI) :**

Latent semantic indexing (LSI) or Latent Semantic Analysis(LSA) is an indexing and information retrieval method. It is one of the major analysis approaches in the field of text mining. It helps in finding out the documents which are most relative with the specified keyword. Similarly it also helps the search engines to give the most appropriate results for the search query.

**Concepts used in the project:-**

**Singular Value Decomposition :**

In combination with text mining, SVD offers the statistical foundation for classification techniques typically referred to as latent semantic indexing. In SVD, matrix A typically consists of word x documents, representing the paper and text as a high-dimensional vector space model (also known as a hyperspace document representation).

* Papers are shown as V rows, and an analysis of the VS rows may be used to determine how similar two documents are.
* Words are shown in U rows.
* By examining the rows of the US matrix, it is possible to determine how similar the phrases are.

**Frobenius Norm** :

The square root of the sum of the squares of a matrix's elements is all that a matrix's Frobenius norm is. With regard to LSI, it compares how unlike two matrices are. Let's imagine that we wish to compare the matrices A1 and A2. Calculating the Frobenius norm of (A1 - A2)(F1) will enable you to do this . Next, we determine A1(F2)'s Frobenius norm . We divide both of them into F1/F2, and the result allows us to determine how dissimilar the two matrices are. This response will always range from 0 to 1. The difference between A1 and A2 will be smaller the more simplistic the response. ( For better understanding: as if A1 = A2, our answer will be 0 since F1 will be equal to 0).

**Jacobi eigenvalue Theorem:-**

Finding the eigenvalues and eigenvectors of symmetric matrices is an iterative process. It is based on the rotational series. In this case, we apply similarity transformations to a matrix in order to change it into a diagonal matrix. The resultant diagonal matrix's diagonal elements will roughly correspond to the eigenvalues of the original supplied matrix.

**Approach:-**

In order to perform LS1, we preprocess the text data. We remove punctuations and store it in a python list, tokenizing each string. We then have to make a wordset which is a set of all unique words of all documents.Now, we will find the frequency of all words in each document one by one. We measured term frequency by (count of words/total number of words in document)for all the documents. Then we store IDF(Inverse Document Frequency) of all words. And then multiply term frequency with the IDF value of each word.

We store these values in a dataframe called X. Rows of X transpose are words and columns are document numbers. Convert this dataframe in a matrix and perform SVD on it. We iterate from 1 to min(m,n) where m = number of rows, n = number of columns singular values and choose the first i (iterator) number of singular values. As we reduce the count of singular values the matrix gets more and more compressed.

After deciding a threshold for the Frobenius norm, we compared the original matrix and the reconstructed matrix. If the value returned by the frobenius function of two matrices is less than this threshold, we will use the reconstructed matrix for the search. For the search function, we ask for an input. If a word is found, we calculate the score of each document otherwise we say no matching documents found. This is done by using the reconstructed matrix.

**Result:-** To compare the weightage of different files, the program computes the values of U,S and V from the SVD approach and after that program runs smoothly.