DATE PAGE

ASSIGNMENT-2

FOR

WEB DATA MINING

TOPIC: Latex Semantic Indexing with SVD.

Submitted to

Gopeekrishnan R

Associate Professor

Dept: of Computer Applications

SCE, Kottayam

Submitted by

Ribin Sara John

S5-RMCA

Roll No: 19

SCE, Kottayam.

Submitted on

14/12/2021

LATENT SEMANTIC INDEXING (LSI)

Inorder to understand the concept of latent semantic indexing, it is very important to look closel into the working of a search engine. A search engine like google use complex algorithms to understand 2 things:

(i) A site's content and its context ie, what purpose is the webpage/site served for.

(ii) A user's reach intent and its relationship to specific keywords.

The LSI helps in delivering the most accurate results by identifying related keywords and process synonyms

Eg: consider the word 'aviator' in a query. The search engine returns some e-commerce web pages that featuring the 'aviator' a brand of the popular Rayban glasses, pages of fan websites featuring the hollywood film 'The aviator'.

One of the best and earliest way to find LSI keywords is Google itself, it automatically suggests some keywords related to the searched query. To provide a better search result, the search engines use LSI keywords to add context to pages.

The LSI adds an important step to the document indexing process. In addition to the keywords, a document also contains the method examines the document collection as a whole. It then, sees which other documents contain some of those same words. LSI considers documents that have many words is common to be semantically close and ones with few words in common to be semantically distant.

In LSI, it is assumed that, there is some underlying latent semantic structure in the data. It then uses a statistical technique called singular value Decomposition(SVD) to estimate this latent structure. This structure is also called hidden concept space. This sounds syntactically different but semantically similar terms and documents. Additionally, the query ~~retrieval~~ is transformed into the concept space before retrieval.

Let 'D' be the document collection

Let the no: of distinctive words in D be 'm'

Let 'n' be the no: of documents in D.

LSI states with a mxn term-document matrix, A where each row of A represents a term and, each column represents a document. The matrix may be computed in various ways.

Eg: using term-frequency or TF-IDF values.

Each entry or cell of the matrix, denoted by A_{ij}, is the no: of times that term 'i' occurs in document 'j'.

<u>Example</u>:

Consider 3 documents,

d1: shipment of gold damaged in a fire

d2: Delivery of silver arrived in a silver truck

d3: shipment of gold arrived in a truck

Suppose that we use the term-frequency as term weights and query weights, the document-indexing rules used as:

- stopwords were not ignored.
- text was tokenized and lowercased.
- no stemming was used.
- terms were sorted alphabetically

Let the query be:

q : gold silver truck

The term-document matrix A and query matrix q are:

$$
\begin{array}{l}
\text{Terms} \\
\text{a} \\
\text{arrived} \\
\text{damaged} \\
\text{delivery} \\
\text{fire} \\
\text{gold} \\
\text{in} \\
\text{of} \\
\text{shipment} \\
\text{silver} \\
\text{truck}
\end{array}
\quad
\begin{array}{c}
\begin{array}{ccc} d_1 & d_2 & d_3 \end{array} \\
A = \begin{bmatrix}
1 & 1 & 1 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 2 & 0 \\
0 & 1 & 1
\end{bmatrix}
\end{array}
\quad
\begin{array}{c}
q \\
q = \begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1
\end{bmatrix}
\end{array}
$$

Singular Value Decomposition

This is another form of matrix analysis that leads to a low-dimensional representation of a high-dimensional matrix. This approach allows an exact representation of any matrix. It also makes us easy to eliminate the less important parts of that matrix representation to produce an approximate representation with any desired no: of dimensions. The fewer the dimensions choosen, the less accurate will be the approximation.

SVD factors matrix A into the product of 3 matrices U, V^T and Σ

ie, $A = U \cdot \Sigma \cdot V^T$

where $U \rightarrow$ a '$m \times r$' matrix & its coloumns are called left singular vectors (Eigen vectors of $A \cdot A^T$)

$\Sigma \rightarrow$ an '$r \times r$' diagonal matrix

$\Sigma = \text{diag}(\sigma_1, \sigma_2, \sigma_3 \ldots \sigma_r)$ such that $\sigma_i > 0$:

$\sigma_1, \sigma_2, \ldots \sigma_r$ are singular values.

$V \rightarrow$ an '$n \times r$' matrix & its columns are called right singular vectors (Eigen vectors of $A^T A$).

An important feature of SVD is that, we can delete some unsignificant dimensions in the concept space to optimally approximate the matrix A. In IR the insignificant dimensions may represent "noise" in the data and be removed.

Let us use only the 'k' largest singular values in Σ and set the remaining small ones to zero

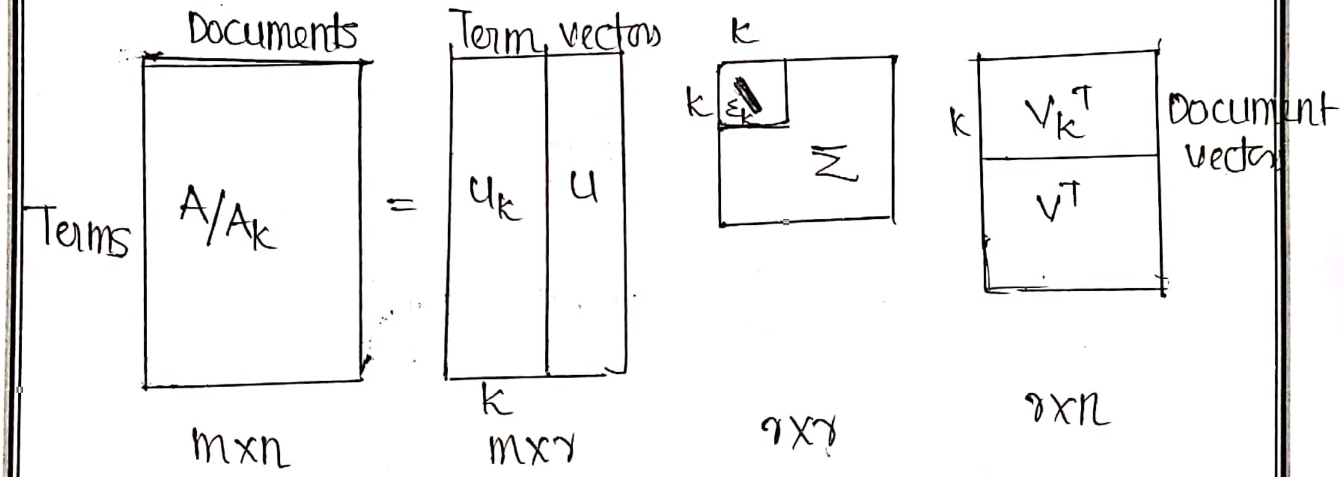
The approximated matrix of A is denoted by A_k. We can also reduce the size of the matrices Σ, U and V be be deleting the last r-k rows and columns from Σ, the last r-k columns in U and the last r-k columns in V.

we thus obtain,

$$A_k = U_k \Sigma_k V_k$$

which means that the use k-largest singular triplets be approximate the original term-document matrix A.

This row space is called the k-concept space. The original matrices and the reduced matrices are



The SVD method for LSI doesn't reconstruct the original term-document matrix 'A' perfectly.