# Latent Semantic Indexing

Narayana Shanmukha Venkat
Computer Science and Engineering [B.Tech]
Roll no : 17075036

# Overview

**Latent semantic analysis** (**LSA**) is a technique in natural language processing, mainly used for analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.

- Dependencies
- Singular Value Decomposition
- Latent Semantic Indexing via the SVD
- Query Representation
- Similarities
- References

# Dependencies

NumPy is the fundamental package for scientific computing with Python.

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- useful linear algebra, Fourier transform, and random number capabilities

Gensim is a Python library for topic modelling, document indexing and similarity retrieval with large corpora. Target audience is the natural language processing (NLP) and information retrieval (IR) community.

NLTK(Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. It provides with a suite of text processing libraries for, tokenization, stemming.

# Singular Value Decomposition

Suppose M is a m × n matrix whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a factorization, called a 'singular value decomposition' of M, of the form

$$\mathbf{M = U\ \Sigma\ V}*$$

M X N = MXM MXN NXN

where

- U is an m × m unitary matrix over K (if K = , unitary matrices are orthogonal matrices),
- Σ is a diagonal m × n matrix with non-negative real numbers on the diagonal,
- V is an n × n unitary matrix over K, and V∗ is the conjugate transpose of V.

The diagonal entries σi of Σ are known as the singular values of M

# Truncated SVD

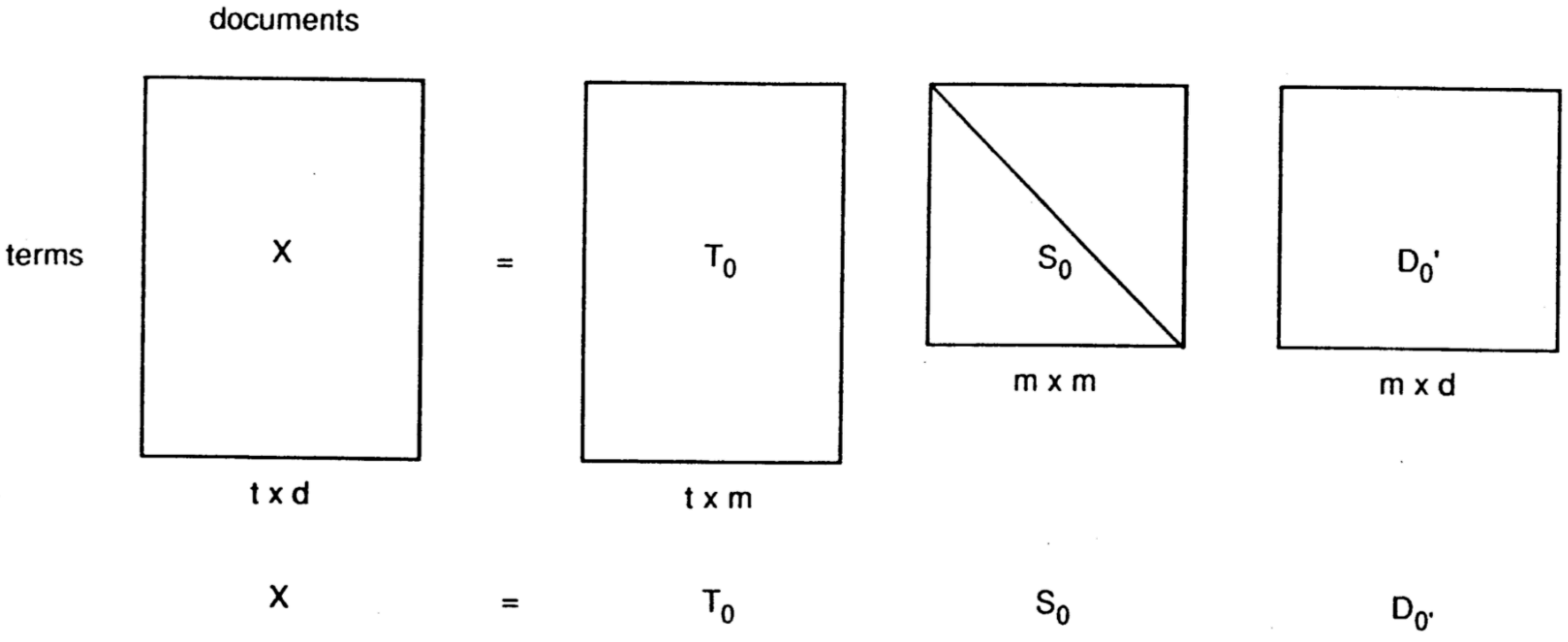
**$M = U_t \Sigma_t V_t^*$**

M X N = MXT TXT TXN

Only the t column vectors of U and t row vectors of V* corresponding to the t largest singular values $\Sigma_t$ are calculated. The rest of the matrix is discarded. This can be much quicker and more economical than the compact SVD if $t \ll r$. The matrix $U_t$ is thus $m \times t$, $\Sigma_t$ is $t \times t$ diagonal, and $V_t^*$ is $t \times n$.

Of course the truncated SVD is no longer an exact decomposition of the original matrix M, but as discussed above, the approximate matrix  is in a very useful sense the closest approximation to M that can be achieved by a matrix of rank t.

# Latent Semantic Indexing via the SVD



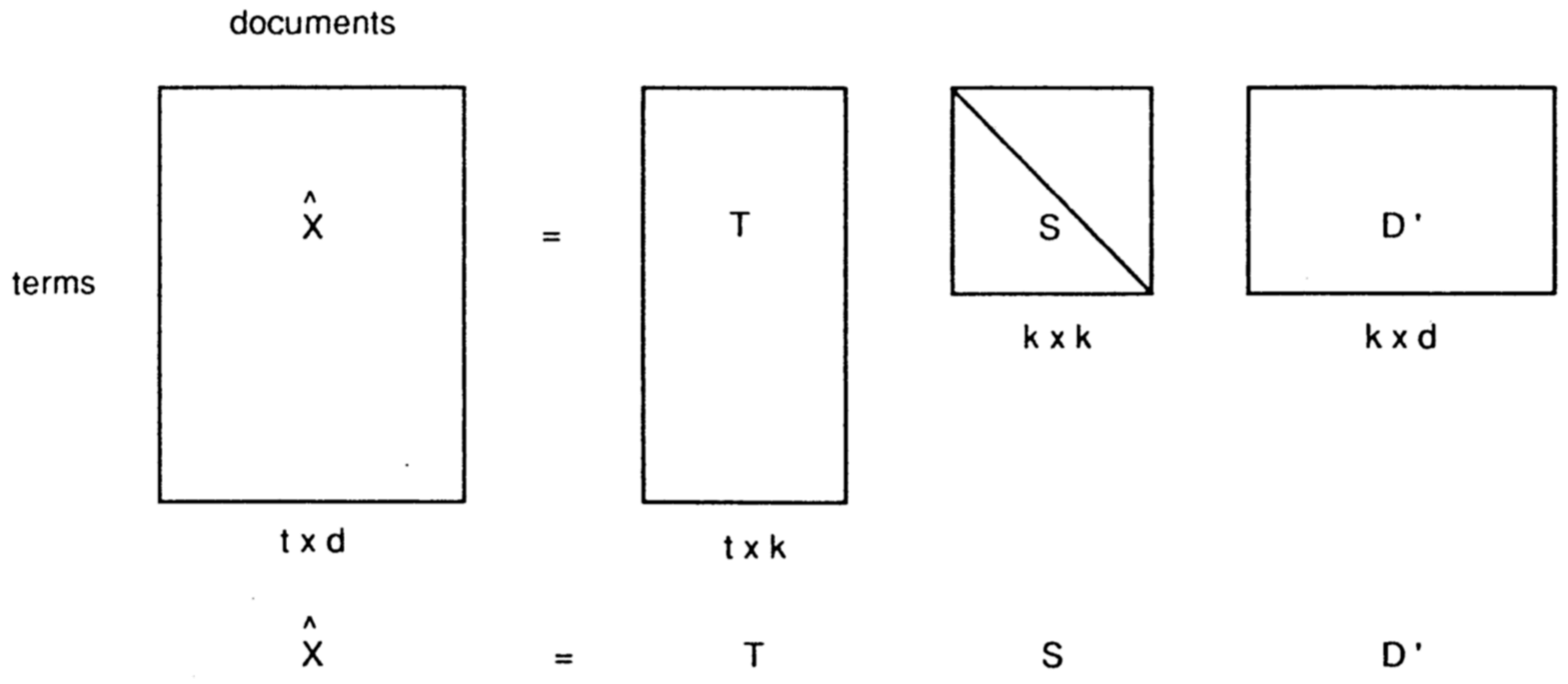

Singular value decomposition of the term x document matrix, X. Where:

$T_0$ has orthogonal, unit-length columns ($T_0' T_0 = I$)
$D_0$ has orthogonal, unit-length columns ($D_0' D_0 = I$)
$S_0$ is the diagonal matrix of singular values
t is the number of rows of X
d is the number of columns of X
m is the rank of X ($\leq \min(t,d)$)

FIG. Schematic of the Singular Value Decomposition (SVD) of a rectangular term by document matrix. The original term by document matrix is decomposed into three matrices each with linearly independent components.

# K- Rank Approximation

If singular values are ordered by size, the first k largest maybe kept and remaining smaller ones are set to zero. The product of the resulting matrices is approximately equal to original one, but obviously of rank k less than the original one.




**Reduced** singular value decomposition of the term x document matrix, X. Where:

T has orthogonal, unit-length columns (T' T = I)
D has orthogonal, unit-length columns (D' D = I)
S is the diagonal matrix of singular values
t is the number of rows of X
d is the number of columns of X
m is the rank of X ($\leq$ min(t,d))
k is the chosen number of dimensions in the reduced model (k $\leq$ m)

FIG. Schematic of the *reduced* Singular Value Decomposition (SVD) of a term by document matrix. The original term by document matrix is *approximated* using the *k* largest singular values and their corresponding singular vectors.

# Query Representation

$$q_k = q^T U_k \Sigma_k^{-1}$$

1 X T = 1XM MXT TXT

- Any query *q* is also mapped into this space,
- Query is NOT a sparse vector.

# Similarities

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

Given two vectors of attributes, A and B, the cosine similarity, cos(θ), is represented using a dot product and magnitude as
where  and  are components of vector  and  respectively.where  and  are components of vector  and  respectively.

*  *Here the query is converted to the lsi space initially by QUERY REPRESENTATION and then be compared with document to topic vectors, i.e V*

*  *Although there might many other ways of finding the similarities, cosine similarity has produced good results*

# References

Deerwester, Scott; Dumais, Susan T.; Furnas, George W.; Landauer, Thomas K.; Harshman, Richard (1990). "Indexing by Latent Semantic Analysis" [ *Journal of the American Society for Information Science* ]