About Latent Semantic Indexing (LSI)

Peter Wullinger

December 21, 2018

Latent Semantic Indexing (LSI) natural language processing technique that is used for a dual purpose:

- during the analysis phase (i.e. Latent Semantic Analysis), features expressed as weighted linear combination of term co-occurences are extracted.
- during the retrieval phase, the extracted features are used as index entries for similarity-based document retrieval. To achieve improved performance and reduce storage requirements, less relevant features are removed, so that the indexing becomes approximate. The resulting approximation is optimal with regard to the the least-squares residual error.

1 Definitions

Let $m \in \mathbb{N}$, $n \in \mathbb{N}$ be natural numbers and let $p = \min(m, n)$.

D an indexed family of size n of documents. The document ith document from D is indicated as $D[i]$. We also call D the *corpus*.

T an indexed family of size m of terms. The jth term from T is indicated as $T[j]$.

d_i a document vector, expressed a vector of size m of term frequencies. Each element of the vector indicates the (relative) frequency of the numbered term within $D[i]$.

t_j a term vector, expressed as a vector of size n of document frequencies. Each element of the vector indicates the (relative) frequency that term $T[j]$ appears within document $D[i]$.

X An $m \times n$ matrix containing the cross-tabulation of all d_is and t_j, the *document-term matrix.*

X is of the form

$$X = \begin{array}{cc} & d_i \\ & \downarrow \\ t_j \rightarrow & \begin{pmatrix} x_{1,1} & \dots & x_{i,n} \\ \vdots & & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{pmatrix} \end{array}$$

Its columns are the n document vectors and its rows are the m term vectors.

Consequently, by this definition

- XX^T contains the dot product of all document vectors, giving the co-occurence of the terms. This is a possible representation of the similarity between all terms in the corpus (using the co-occurence from the corpus documents).
- X^TX contains the dot product of all term vectors, giving the co-occurence of documents over each term. This is a possible representation for the similarity between all documents in the corpus.

2 Similarity Queries

One question to ask from a document repository is to retrieve all documents that are relevant to a certain query q, with with q being a conjuctive query for a set of terms $T_q \sqsubseteq T$.

A simple way of doing this is calculating the pseudo-document d_q which contains exactly the terms that appear in q and calculate

$$\mathrm{sim}_c = X^T \cdot d_q$$

This yields the cosine similarity of all documents from the corpus with the query document. We can then retrieve the set of documents with the highest similarity.

However, this approach has at least two drawbacks:

1. Since $m = |T|$ is most likely large, we have to store a large number of values per document, which is undesirable when we also have a large corpus.
2. The direct approach is only able to consider direct occurence of a term and has no concept of semantic relationship between terms. This means documents that do not contain exactly the words from the query get a low similarity score, even if they contain related terms.

LSI approaches both problems using a corpus-based approach for determining semantic relationship and a linear algebra-based approach for reducing index size.

3 Singular Value Decomposition

For every real (or complex) matrix X, there exists a decomposition

$$M = U\Sigma V^*$$

such that

U is an orthogonal (or unitary) $m \times m$ matrix, i.e. $U^TU = UU^T = I$ $(U^*U = UU^* = I)$.

V is an orthogonal (or unitary) $n \times n$ matrix, i.e. $V^TV = VV^T = I$ $(V^*V = VV^* = I)$.

Σ is a positiv semidefinite rectangular mxn matrix. The non-zero (diagonal) elements of Σ are called the *singular values* of X.

Hence,

$$M = \begin{pmatrix} u_{1,1} & \dots & u_{1,m} \\ \vdots & & \vdots \\ u_{m,1} & \dots & u_{m,m} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & & \dots & 0 \\ 0 & \ddots & & \dots & 0 \\ 0 & 0 & \sigma_p & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_{1,1} & \dots & v_{1,n} \\ \vdots & & \vdots \\ v_{n,1} & \dots & v_{n,n} \end{pmatrix}$$

is called the *singular value decomposition* of M.

We will assume that we are dealing exclusively with real-values matrices, hence we will only use ^{T} for the remainder of this document. If dealing with complex matrices, the conjugate transpose * needs to be used.

Is is customary to list the singular values σ_i (in Σ) in descending order. If this requires re-ordering, the columns of U and V (rows of V^T) also need to be re-arranged, accordingly. For example for the case $m = 2, n = 2$, the following holds:

$$\underbrace{\begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}}_{\Sigma} \underbrace{\begin{pmatrix} v_{1,1} & v_{2,1} \\ v_{1,2} & v_{2,2} \end{pmatrix}}_{V^T} = \underbrace{\begin{pmatrix} u_{1,2} & u_{1,1} \\ u_{2,2} & u_{2,1} \end{pmatrix}}_{U'} \underbrace{\begin{pmatrix} \sigma_2 & 0 \\ 0 & \sigma_1 \end{pmatrix}}_{\Sigma'} \underbrace{\begin{pmatrix} v_{2,1} & v_{2,2} \\ v_{1,1} & v_{1,2} \end{pmatrix}}_{V'^T}$$

Truncated SVD

If we write out the example

$$\begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_{1,1} & v_{2,1} \\ v_{1,2} & v_{2,2} \end{pmatrix} = \begin{pmatrix} u_{1,1}\sigma_1 v_{1,1} + u_{1,2}\sigma_2 v_{1,2} & u_{1,1}\sigma_1 v_{2,1} + u_{1,2}\sigma_2 v_{2,2} \\ u_{2,1}\sigma_1 v_{1,1} + u_{2,2}\sigma_2 v_{1,2} & u_{2,1}\sigma_1 v_{1,2} + u_{2,2}\sigma_2 v_{2,2} \end{pmatrix}$$

we can also see, that X is the dot product between all the rows of U and the columns of V^T, with the ith summand = row of U = column of V^T) scaled by σ_i. Hence, if

Σ is in descending order, each subsequent row of U and each subsequent column of V contribute less and less to the final value.

This makes it possible to turn X into an approximation X_k represented by the *truncated SVD* $U_k \Sigma_k V_k^T$, which is calculated as follows:

For some value $k < p$,

- truncate U to an $m \times k$-matrix U_k,
- truncate Σ to an $k \times k$-matrix Σ_k, and
- truncate V to an $m \times k$-matrix V_k (making V_k^T into a $k \times m$-matrix).

It can be shown, that –if Σ is in order of descending values– this truncation results in a least-squares approximation of X by X_k on the residual errors.

4 Latent Semantic Index Space

Remember from section 1, that XX^T correlates the co-occurence of all terms over the documents. When expressed in SVD-form, this becomes

$$\begin{aligned} X^T X &= (U\Sigma V^T)^T (U\Sigma V^T) && (1) \\ &= V\Sigma^T U^T (U\Sigma V^T) && (2) \\ &= V\Sigma\Sigma V^T && (3) \\ &= V\Sigma^2 V^T && (4) \\ &= (V\Sigma)(V\Sigma)^T && (5) \end{aligned}$$

This means, the correspondence between two documents can be expressed completely using V and Σ, after an initial transformation (via U).

Since Σ is a constant factor, the columns of V alone can be interpreted as a "hidden factor representation" of the documents in the corpus. And thus, if we can transform a query document d_q into its hidden representation $\hat{d}_q$, we can compare its hidden representations $\Sigma\hat{d}_q$ with the columns of ΣV^T (or some other document $\Sigma\hat{d}'$.

5 Obtaining the hidden representation

How do we obtain the hidden representation $\hat{d}$ for some document vector d? Remember that

$$X \approx U_k \Sigma_k {V_k}^T$$

Now, since X actually contains the document vectors in its columns and as noted in section 3, every column of X maps to a column in V_k^T, we consider the columns of V_k^T the hidden representations of the columns of X.

$$\begin{aligned}
X_k &= U_k \Sigma_k {V_k}^T \\
X_k &= U_k \Sigma_k {V_k}^T && |U^{-1} \cdot \\
U_k^{-1} X_k &= \Sigma_k {V_k}^T && |\Sigma_k^{-1} \cdot \\
\Sigma_k^{-1} U_k^{-1} X_k &= {V_k}^T \\
\Sigma_k^{-1} U_k^T X_k &= V_k^T
\end{aligned}$$

Replacing V_k^T with $\hat{d}$ and X_k with d then yields

$$\hat{d} \quad = \Sigma_k^{-1} {U_k}^T d$$

We then can compare the hidden representations of two documents d and d'

$$\begin{aligned}
\hat{d_1} &= \Sigma_k^{-1} U_k^T d_1 \Sigma_k \\
\hat{d'} &= \Sigma_k^{-1} U_k^T d' \Sigma_k
\end{aligned}$$

using apppropriate similarity measures (e.g. cosine similarity).