# Modern Information Retrieval

## Latent Semantic Indexing

Hamid Beigy

Sharif university of technology

May 5, 2025

# Table of contents

# Introduction

# Text representation

1. Document Model

   **Words** A word is an element in a vocabulary set.

   **Sentences** A sentence is a collection of words.

   We can get word meaning, POS tags, and word sentiment.

   **Documents** A document is a collection of words (not a sequence).

   **Corpus** A corpus is a collection of documents.

2. Topic Model

   **Topic** A topic is collection of words (subset of vocabulary).

   **Document** A document is represented by (latent) mixture of topics.

3. What is a document?

   - Can we do everything on document level ?
   - Can we treat tweets, short posts, sentences, and queries as a document?
   - How can we measure similarity between two documents / sentences?

4. We want to represent documents and terms as vectors in a lower-dimensional space.

# Topic modeling

1. When there are more information, it becomes more difficult to access what we are looking for.
2. We need new tools for helping to organize information, search, and understand these information.
3. Topic modeling provides methods for automatically organizing, understanding, searching, and summarizing large electronic archives.
   - Uncover the hidden topical patterns that pervade the collection.
   - Annotate the documents according to those topics.
   - Use the annotations to organize, summarize, and search the texts.

# Document representation

1. **Bag of words**, which assumes order of words has no significance. This is a simplifying assumption used in NLP and IR.
2. **Vector space model**, which represents a document by a high-dimensional vector in the space of words.
   - the number of words is huge
   - high dimensionality is noisy and sparse.
   - We want to represent documents and terms as vectors in a lower-dimensional space.

# Term-document matrix

1. Consider the following term-document matrix

| | Anthony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| anthony | 5.25 | 3.18 | 0.0 | 0.0 | 0.0 | 0.35 |
| brutus | 1.21 | 6.10 | 0.0 | 1.0 | 0.0 | 0.0 |
| caesar | 8.59 | 2.54 | 0.0 | 1.51 | 0.25 | 0.0 |
| calpurnia | 0.0 | 1.54 | 0.0 | 0.0 | 0.0 | 0.0 |
| cleopatra | 2.85 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| mercy | 1.51 | 0.0 | 1.90 | 0.12 | 5.25 | 0.88 |
| worser | 1.37 | 0.0 | 0.11 | 4.15 | 0.25 | 1.95 |

2. This matrix is basis for computing the similarity between documents and queries.
3. Can we transform this matrix, so that we get a better measure of similarity between documents and queries?

# Latent semantic indexing

# Latent semantic indexing (an overview)

1. We will decompose the term-document matrix into a product of matrices.
2. We use singular value decomposition (SVD).
3. SVD is $\mathbf{C} = \mathbf{U}\Sigma\mathbf{V}^T$ (where $\mathbf{C}$= term-document matrix)
4. We will then use the SVD to compute a new, improved term-document matrix $\mathbf{C}'$.
5. We'll get better similarity values out of $\mathbf{C}'$ (compared to $\mathbf{C}$).
6. Using SVD for this purpose is called latent semantic indexing or LSI.
7. Idea : Words that co-occur frequently in documents should be similar.

**Theorem (Matrix decomposition)**

*Let* $\mathbf{S}$ *be square real-valued* $M \times M$ *matrix with* $M$ *linearly independent eigenvectors. Then, there exists an eigen decomposition*

$$\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^{-1},$$

*where the columns of* $\mathbf{U}$ *are the eigenvectors of* $\mathbf{S}$ *and* $\Lambda$ *is a diagonal matrix whose diagonal entries are eigenvalues of* $\mathbf{S}$ *in decreasing order* ($\lambda_i \geq \lambda_{i+1}$).
*If the eigenvalues are distinct, then this decomposition is unique.*

Matrix $\mathbf{U}$ has the eigenvectors of $\mathbf{S}$ as columns $\mathbf{U} = (\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_M)$. Then, we have

$$\begin{aligned}\mathbf{SU} &= \mathbf{S}\,(\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_M) = (\lambda_1\mathbf{u}_1\ \lambda_2\mathbf{u}_2\ \ldots\ \lambda_M\mathbf{u}_M) \\ &= (\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_M)\begin{pmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ldots & \\ & & & \lambda_M\end{pmatrix}\end{aligned}$$

Then, we have $\mathbf{SU} = \mathbf{U}\Lambda$ or $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^{-1}$.

# Symmetric matrix decomposition

**Theorem (Symmetric matrix decomposition)**

*Let* $\mathbf{S}$ *be square, symmetric real-valued* $M \times M$ *matrix with* $M$ *linearly independent eigenvectors. Then, there exists a symmetric diagonal decomposition*

$$\mathbf{S} = \mathbf{Q}\Lambda\mathbf{Q}^\top,$$

*where the columns of* $\mathbf{U}$ *are the orthogonal and normalized (unit length, real) eigenvectors of* $\mathbf{S}$ *and* $\Lambda$ *is a diagonal matrix whose diagonal entries are eigenvalues of* $\mathbf{S}$*. Further, all entries of* $\mathbf{Q}$ *are real and we have* $\mathbf{Q}^{-1} = \mathbf{Q}^\top$*.*

**Theorem (Singular value decomposition)**

*Let $r$ be the rank of the $M \times N$ matrix $\mathbf{C}$. Then, there is a singular value decomposition (SVD) of $\mathbf{C}$ of the form*

$$\mathbf{C} = \mathbf{U}\Sigma\mathbf{V}^\top,$$

*where*

- *The eigenvectors $\lambda_1, \ldots, \lambda_r$ of $\mathbf{C}\mathbf{C}^\top$ are the same as the eigenvectors of $\mathbf{C}^\top\mathbf{C}$.*
- *For $1 \leq i \leq r$, let $\sigma_i = \sqrt{\lambda_i}$, with $\lambda_i \geq \lambda_{i+1}$. Then, $M \times N$ matrix $\Sigma$ is composed by setting $\Sigma_{ii} = \sigma_i$ for $1 \leq i \leq r$, and zero otherwise.*

We have

- Matrix $\mathbf{U}$ is of dimensions $M \times r$.
- Matrix $\Sigma$ is of dimensions $r \times r$.
- Matrix $\mathbf{V}^\top$ is of dimensions $r \times N$.
- $\mathbf{U}$ and $\mathbf{V}$ are orthogonal: $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$.

# Example of $C = U\Sigma V^T$

1. Consider the following term-document matrix

| $C$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| ship | 1 | 0 | 1 | 0 | 0 | 0 |
| boat | 0 | 1 | 0 | 0 | 0 | 0 |
| ocean | 1 | 1 | 0 | 0 | 0 | 0 |
| wood | 1 | 0 | 0 | 1 | 1 | 0 |
| tree | 0 | 0 | 0 | 1 | 0 | 1 |

2. This is a standard term-document matrix.
3. Actually, we use a non-weighted matrix here to simplify the example.

# Example of $C = U\Sigma V^T$

1. The matrix $U$ equals to

| $U$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| ship | −0.44 | −0.30 | 0.57 | 0.58 | 0.25 |
| boat | −0.13 | −0.33 | −0.59 | 0.00 | 0.73 |
| ocean | −0.48 | −0.51 | −0.37 | 0.00 | −0.61 |
| wood | −0.70 | 0.35 | 0.15 | −0.58 | 0.16 |
| tree | −0.26 | 0.65 | −0.41 | 0.58 | −0.09 |

2. One row per term, one column per $\min(M, N)$ where $M$ is the number of terms and $N$ is the number of documents.
3. This is an orthonormal matrix:
   - Row vectors have unit length
   - Any two distinct row vectors are orthogonal to each other
4. Think of the dimensions as semantic dimensions that capture distinct topics like politics, sports, economics.
5. Each number $u_{ij}$ in the matrix indicates how strongly related term $i$ is to the topic represented by semantic dimension $j$.

# Example of $C = U\Sigma V^T$

1. The matrix $\Sigma$ equals to

| $\Sigma$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 2.16 | 0.00 | 0.00 | 0.00 | 0.00 |
| 2 | 0.00 | 1.59 | 0.00 | 0.00 | 0.00 |
| 3 | 0.00 | 0.00 | 1.28 | 0.00 | 0.00 |
| 4 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 |
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.39 |

2. This is a square, diagonal matrix of dimensionality $min(M, N) \times min(M, N)$.
3. The diagonal consists of the singular values of $C$.
4. The magnitude of the singular value measures the importance of the corresponding semantic dimension.
5. We'll make use of this by omitting unimportant dimensions.

# Example of $C = U\Sigma V^T$

1. The matrix $V^T$ equals to

| $V^T$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| 1 | −0.75 | −0.28 | −0.20 | −0.45 | −0.33 | −0.12 |
| 2 | −0.29 | −0.53 | −0.19 | 0.63 | 0.22 | 0.41 |
| 3 | 0.28 | −0.75 | 0.45 | −0.20 | 0.12 | −0.33 |
| 4 | 0.00 | 0.00 | 0.58 | 0.00 | −0.58 | 0.58 |
| 5 | −0.53 | 0.29 | 0.63 | 0.19 | 0.41 | −0.22 |

2. One column per document, one row per $\min(M, N)$ where $M$ is the number of terms and $N$ is the number of documents.
3. This is an orthonormal matrix:
   - Column vectors have unit length
   - Any two distinct column vectors are orthogonal to each other.
4. These are again the semantic dimensions from matrices $U$ and $\Sigma$ that capture distinct topics like politics, sports, economics.
5. Each number $v_{ij}$ in the matrix indicates how strongly related document $i$ is to the topic represented by semantic dimension $j$.

1. Given an $M \times N$ matrix $\mathbf{C}$ and a positive integer $k$, we wish to find an $M \times N$ matrix $\mathbf{C}_k$ of rank at most $k$, so as to minimize the **Frobenius norm** of the matrix difference $\mathbf{X}_F = \mathbf{C} - \mathbf{C}_k$, defined as

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} X_{ij}^2}.$$

2. Thus, the **Frobenius norm** of $\mathbf{X}$ measures the discrepancy between $\mathbf{C}_k$ and $\mathbf{C}$.
3. Let $k$ be the rank of $\mathbf{C}_k$ and $r$ be the rank of $\mathbf{C}$. When $k < r$, we refer to $\mathbf{C}_k$ as a **low-rank approximation**.

**Theorem (Low-rank approximation)**

*Let $C_k$ be an approximation of matrix $\mathbf{C}$ whose rank is at most $k$. Then,*

$$\min_{\mathbf{Z} \mid \mathit{rank}(\mathbf{Z})=k} \{\|\mathbf{C} - \mathbf{Z}\|_F\} = \|\mathbf{C} - \mathbf{C}_k\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2},$$

*where $\sigma_1 \geq \sigma_2 \geq \ldots$.*

# Low-rank approximation

1. SVD can be used to solve the low-rank approximation problem.
2. We use the following three-step procedure
   - Given $\mathbf{C}$, construct its SVD as $\mathbf{C}_k = \mathbf{U}\Sigma\mathbf{V}^\top$.
   - Drive matrix $\Sigma_k$ from $\Sigma$ by replacing $r - k$ smallest singular values of $\Sigma$ to **zero**.
   - Compute and output $\mathbf{C}_k = \mathbf{U}_k\Sigma_k\mathbf{V}_k^\top$ as the rank-$k$ approximation to $k$.

# Latent semantic analysis

1. In IR, it is sometimes called Latent Semantic Indexing (LSI).
2. LSI is a method for discovering hidden concepts in document data.
3. Each document and term is then expressed as a vector with elements corresponding to these concepts.
4. Each element in a vector gives the degree of participation of the document or term in the corresponding concept.
5. Goal is not to describe the concepts verbally, but representing documents and terms in a unified way for exposing document-document, document-term, and term-term similarities or semantic relationship which are otherwise hidden.

# Latent semantic analysis

1. Matrix $\mathbf{B} = \mathbf{A}^\top \mathbf{A}$ is the document-document matrix. If documents $i$ and $j$ have $b$ words in common then $\mathbf{B}[i,j] = b$.
2. Matrix $\mathbf{C} = \mathbf{A}\mathbf{A}^\top$ is the term-term matrix. If terms $i$ and $j$ occur together in $c$ documents, then $\mathbf{C}[i,j] = c$.
3. Terms and documents have new representations in terms of these hidden concepts.
4. The terms are represented by the row vectors of $M \times k$ matrix $\mathbf{U}_k \Sigma_k$.
5. The documents by the column vectors the $k \times n$ matrix $\Sigma_k \mathbf{V}_k^\top$.

# Latent semantic analysis

1. General idea
   - Map documents (and terms) to a low-dimensional representation
   - Design a mapping such that the low-dimensional space reflects semantic association (latent semantic space)
   - Compute document similarity based on inner product in latent semantic space
2. Goals
   - Similar terms map to similar location in low dimensional space
   - LSI projects queries and documents into a space with latent semantic dimensions.
   - Noise reduction by dimension reduction
3. Limitations
   - No probabilistic model of term occurrences
   - Results are difficult to interpret
   - Assumes that words and documents form a joint Gaussian model
   - Arbitrary selection of the number of dimensions k
   - Cannot account for polysemy
   - No generative model

# Latent semantic indexing (Summary)

1. We decompose term-document matrix $C$ into a product of three matrices: $U\Sigma V^\top$.

   **The term matrix** $U$ consists of one row vector for each term.

   **The document matrix** $V^\top$ consists of one column vector for each document.

   **The singular value matrix** $\Sigma$ diagonal matrix with singular values, reflecting importance of each dimension

2. Transform query vector using transformation $\mathbf{q}_k = \mathbf{q}\Sigma_k^{-1}\mathbf{U}_k^\top$.

3. This is obtained using the fact that every document is represented by a column vector of $V^\top$. Hence, $\mathbf{V}_k = \mathbf{C}_k^\top \Sigma_k^{-1}\mathbf{U}_k$.

4. Compute the similarity two terms or two documents using cosine similarity.

# Dimensionality reduction

# How we use the SVD in LSI

1. Key property: Each singular value tells us how important its dimension is.
2. By setting less important dimensions to zero, we keep the important information, but get rid of the "details".
3. These details may
   - be noise – in that case, reduced LSI is a better representation because it is less noisy.
   - make things dissimilar that should be similar – again, the reduced LSI representation is a better representation because it represents similarity better.
4. Analogy for "fewer details is better"
   - Image of a blue flower
   - Image of a yellow flower
   - Omitting color makes is easier to see the similarity

# Reducing the dimensionality to 2

| $U$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| ship | −0.44 | −0.30 | 0.00 | 0.00 | 0.00 |
| boat | −0.13 | −0.33 | 0.00 | 0.00 | 0.00 |
| ocean | −0.48 | −0.51 | 0.00 | 0.00 | 0.00 |
| wood | −0.70 | 0.35 | 0.00 | 0.00 | 0.00 |
| tree | −0.26 | 0.65 | 0.00 | 0.00 | 0.00 |

| $\Sigma_2$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 2.16 | 0.00 | 0.00 | 0.00 | 0.00 |
| 2 | 0.00 | 1.59 | 0.00 | 0.00 | 0.00 |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

| $V^T$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| 1 | −0.75 | −0.28 | −0.20 | −0.45 | −0.33 | −0.12 |
| 2 | −0.29 | −0.53 | −0.19 | 0.63 | 0.22 | 0.41 |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

- Actually, we only zero out singular values in $\Sigma$.
- This has the effect of setting the corresponding dimensions in $U$ and $V^T$ to zero when computing the product $C = U\Sigma V^T$.

# Reducing the dimensionality to 2

| $C_2$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| ship | 0.85 | 0.52 | 0.28 | 0.13 | 0.21 | −0.08 |
| boat | 0.36 | 0.36 | 0.16 | −0.20 | −0.02 | −0.18 |
| ocean | 1.01 | 0.72 | 0.36 | −0.04 | 0.16 | −0.21 |
| wood | 0.97 | 0.12 | 0.20 | 1.03 | 0.62 | 0.41 |
| tree | 0.12 | −0.39 | −0.08 | 0.90 | 0.41 | 0.49 |

=

| $U$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| ship | −0.44 | −0.30 | 0.57 | 0.58 | 0.25 |
| boat | −0.13 | −0.33 | −0.59 | 0.00 | 0.73 |
| ocean | −0.48 | −0.51 | −0.37 | 0.00 | −0.61 |
| wood | −0.70 | 0.35 | 0.15 | −0.58 | 0.16 |
| tree | −0.26 | 0.65 | −0.41 | 0.58 | −0.09 |

×

| $\Sigma_2$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 2.16 | 0.00 | 0.00 | 0.00 | 0.00 |
| 2 | 0.00 | 1.59 | 0.00 | 0.00 | 0.00 |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

×

| $V^T$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| 1 | −0.75 | −0.28 | −0.20 | −0.45 | −0.33 | −0.12 |
| 2 | −0.29 | −0.53 | −0.19 | 0.63 | 0.22 | 0.41 |
| 3 | 0.28 | −0.75 | 0.45 | −0.20 | 0.12 | −0.33 |
| 4 | 0.00 | 0.00 | 0.58 | 0.00 | −0.58 | 0.58 |
| 5 | −0.53 | 0.29 | 0.63 | 0.19 | 0.41 | −0.22 |

# Original matrix $C$ vs. reduced $C_2 = U\Sigma_2 V^T$

| $C$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| ship | 1 | 0 | 1 | 0 | 0 | 0 |
| boat | 0 | 1 | 0 | 0 | 0 | 0 |
| ocean | 1 | 1 | 0 | 0 | 0 | 0 |
| wood | 1 | 0 | 0 | 1 | 1 | 0 |
| tree | 0 | 0 | 0 | 1 | 0 | 1 |

| $C_2$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| ship | 0.85 | 0.52 | 0.28 | 0.13 | 0.21 | −0.08 |
| boat | 0.36 | 0.36 | 0.16 | −0.20 | −0.02 | −0.18 |
| ocean | 1.01 | 0.72 | 0.36 | −0.04 | 0.16 | −0.21 |
| wood | 0.97 | 0.12 | 0.20 | 1.03 | 0.62 | 0.41 |
| tree | 0.12 | −0.39 | −0.08 | 0.90 | 0.41 | 0.49 |

We can view $C_2$ as a two-dimensional representation of the matrix $C$. We have performed a dimensionality reduction to two dimensions.

# Why the reduced matrix $C_2$ is better than $C$

| $C$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| ship | 1 | 0 | 1 | 0 | 0 | 0 |
| boat | 0 | 1 | 0 | 0 | 0 | 0 |
| ocean | 1 | 1 | 0 | 0 | 0 | 0 |
| wood | 1 | 0 | 0 | 1 | 1 | 0 |
| tree | 0 | 0 | 0 | 1 | 0 | 1 |

- Similarity of $d_2$ and $d_3$ in the original space: 0.
- Similarity of $d_2$ and $d_3$ in the reduced space: $\approx 0.52$
- boat and ship are semantically similar.
- The reduced similarity measure reflects this.
- What property of the SVD reduction is responsible for improved similarity? (Do it as an exercise.)

| $C_2$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ |
|---|---|---|---|---|---|---|
| ship | 0.85 | 0.52 | 0.28 | 0.13 | 0.21 | −0.08 |
| boat | 0.36 | 0.36 | 0.16 | −0.20 | −0.02 | −0.18 |
| ocean | 1.01 | 0.72 | 0.36 | −0.04 | 0.16 | −0.21 |
| wood | 0.97 | 0.12 | 0.20 | 1.03 | 0.62 | 0.41 |
| tree | 0.12 | −0.39 | −0.08 | 0.90 | 0.41 | 0.49 |

# LSI in information retrieval

# Why we use LSI in information retrieval?

1. LSI takes documents that are semantically similar (=talk about the same topics),
2. But these documents are not similar in the vector space (because they use different words)
3. LSI re-represents them in a reduced vector space in which they have higher similarity.
4. Thus, LSI addresses the problems of synonymy and semantic relatedness.
5. Standard vector space: Synonyms contribute nothing to document similarity.
6. Desired effect of LSI: Synonyms contribute strongly to document similarity.

# How LSI addresses synonymy and semantic relatedness?

1. The dimensionality reduction forces us to omit a lot of detail.
2. We have to map different words to the same dimension in the reduced space.
3. The cost of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words.
4. SVD selects the least costly mapping (see below).
5. Thus, it will map synonyms to the same dimension.
6. But it will avoid doing that for unrelated words.

# Comparison to other approaches

1. Recap: Relevance feedback and query expansion are used to increase recall in information retrieval – if query and documents have no terms in common.

   or, more commonly, too few terms in common for a high similarity score

2. LSI increases recall and decreases precision. (why? do as an exercise.)

3. Thus, it addresses the same problems as (pseudo) relevance feedback and query expansion and it has the same problems.

# LSI vs PCA

1. Like PCA, LSI aims at finding the directions of high correlations between words called principal directions.
2. Like PCA, it retains the projection of the data on a number k of these principal directions, which are called the principal components.
3. Unlike PCA, LSI does not center the data (no specific reason).
4. LSI is typically combined with TF-IDF

# Optimality

1. SVD is optimal in the following sense.
2. Keeping the $k$ largest singular values and setting all others to zero gives you the optimal approximation of the original matrix $\mathbf{C}$ (Eckart-Young theorem)
3. Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates $\mathbf{C}$ better.
4. Measure of approximation is Frobenius norm: $\|\mathbf{C}\|_F = \sqrt{\sum_i \sum_j c_{ij}^2}$
5. So LSI uses the best possible matrix.
6. There is only one best possible matrix – unique solution
7. There is only a tenuous relationship between the Frobenius norm and cosine similarity between documents. (describe it as an exercise.)

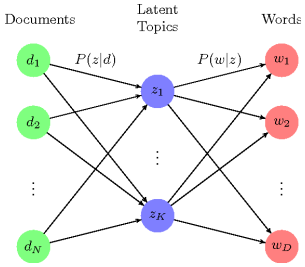
# Probabilistic Latent semantic indexing

# Probabilistic Latent semantic indexing (pLSI)

1. Difference between topics and words?
   - Words are observable.
   - Topics are not, they are latent.
2. probabilistic LSI (pLSI) model, also known as the aspect model, we (Hofmann 1999),
   - Associates an unobserved latent variable $z \in Z = \{z_1, \ldots, z_k\}$ with each observation.
   - Defines a joint probability model over documents and words
   - Assumes $w$ is independent of $d$ conditioned on $z$.
   - Cardinality of $z$ should be much less than than $M$ and $N$.

1. Basic generative model
   - Select document $d$ with probability $p(d)$.
   - Select a latent class $z$ with probability $p(z|d)$.
   - Generate a word $w$ with probability $p(w|z)$.
2. Joint probability model

$$p(d, w) = p(d)p(w|d)$$
$$p(w|d) = \sum_z p(w|z)p(z|d)$$




The goal is to minimize KL divergence (cross-entropy) between the empirical distribution of words and the model distribution $p(w|d)$ as given below.

$$\mathcal{L} = \sum_{d \in D} \sum_{w \in W} n(d, w) \log p(d, w)$$

The above objective function is optimized using EM algorithm.

# Probabilistic Latent semantic indexing (summary)

1. pLSA strengths
   - Models word-document co-occurrences as a mixture of conditionally independent multinomial distributions
   - A mixture model, not a clustering model
   - Results have a clear probabilistic interpretation
   - Allows for model combination
   - Problem of polysemy is better addressed
2. pLSA Limitations
   - Potentially higher computational complexity than LSI
   - EM algorithm gives local maximum
   - Prone to overfitting. As a solution, we can use Tempered EM.
   - Not a well defined generative model for new documents. As a solution, we can use Latent Dirichlet Allocation (Blei, Ng, and Jordan 2003).

# References

# Reading

1. Chapter 18 of Introduction to Information Retrieval[1]

[1] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

# References


- Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3, pp. 993–1022.
- Hofmann, Thomas (1999). "Probabilistic Latent Semantic Analysis". In: *Proc. of the 15 Conf. on Uncertainty in Artificial Intelligence (UAI)*.
- Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

# Questions?