Text Analytics
Modeling Information Retrieval 2

Ulf Leser
Content of this Lecture

- IR Models
- Boolean Model
- Vector Space Model
- Relevance Feedback in the VSM
- **Probabilistic Model**
- Latent Semantic Indexing
- Other IR Models
A Probabilistic Interpretation of Relevance

• We want to compute the probability that a doc $d$ is relevant to query $q$

• The probabilistic model determines this probability iteratively using user (or automatic) feedback
  – Similar to VSM with relevance feedback
  – But different (and more principled) way of incorporating feedback

• Assume there is a subset $R \subseteq D$ which contains all (and only) relevant documents for $q$, and $N=D\setminus R$

• For each document, we want to compute the probability $p(R|d)$ that $d$ belongs to $R$ (for $q$)

• This is based on the words in $d$, i.e., we represent $d$ as the set of words contained in $d=\{k_i\}$
A Probabilistic Interpretation of Relevance

- Since words $k_i$ appear both in relevant and in irrelevant docs, we need to look at the influence of both.
- We use odds-scores

\[
rel(d, q) \sim sim(d, q) = \frac{p(R | d)}{p(N | d)}
\]

- Assuming statistical independence of words, we get
  - Clearly wrong

\[
rel(d, q) \sim sim(d, q) = \frac{p(R | k_1, \ldots, k_n)}{p(N | k_1, \ldots, k_n)} = \frac{p(R | k_1) \ast \ldots \ast p(R | k_n)}{p(N | k_1) \ast \ldots \ast p(N | k_n)}
\]
Binary Independence Model I

- **Using Bayes Theorem**

\[
sim(d, q) = \frac{p(R | d)}{p(N | d)} = \frac{p(d | R) \times p(R) \times p(d)}{p(d | N) \times p(N) \times p(d)} \sim \frac{p(d | R)}{p(d | N)}
\]

- \(p(R) (p(N))\): relative frequency of relevant (irrelevant) docs in \(D\)
  - A-Priori probability of a doc to be (ir-)relevant
- \(p(R)\) and \(p(N)\) are independent from \(d\) and \(q\) – thus, both are constant for \(q\) and irrelevant for ranking documents
- \(p(d | R)\) is the probability of drawing the combination of words forming \(d\) when drawing words at random from \(R\)
Binary Independence Model II

- $p(d|R)$: Drawing the words from $d$ means two things
  - Drawing the words from $d$, and
  - not drawing the words not in $d$

- **BIN** considers both plus independence of terms

\[
\text{sim}(d, q) = \frac{p(d \mid R)}{p(d \mid N)} = \frac{\prod_{k \in d} p(k \mid R) \ast \prod_{k \notin d} p(-k \mid R)}{\prod_{k \in d} p(k \mid N) \ast \prod_{k \notin d} p(-k \mid N)}
\]

- Properties
  - Having words that are frequent in $R$ raises the similarity to $q$
  - Not having words that are frequent in $N$ raises the similarity to $q$

- Why is no $q$ in the formula?
Continuation

- Rephrasing using q

\[
\text{sim}(d, q) = \prod_{k \in d \cap q} p(k \mid R) \prod_{k \notin d \cap q \mid d \cap q} p(k \mid N) \prod_{k \notin d \cap q} p(k \mid R) \prod_{k \in d \cap q \mid d \cap q} p(k \mid N) \prod_{k \notin d \cap q} p(k \mid R) \prod_{k \notin d \cap q} p(k \mid N)
\]

- Focusing on the query terms
  - In a real setting we are not sure about R and N – give less weight to terms not in the query
  - Drastically increases performance

... \approx \prod_{k \in d \cap q} \frac{p(k \mid R)}{p(k \mid N)} \prod_{k \notin d \cap q} \frac{p(-k \mid R)}{p(-k \mid N)} = \prod_{k \in d \cap q} \frac{p(k \mid R)}{p(k \mid N)} \prod_{k \notin d \cap q} \frac{1 - p(k \mid R)}{1 - p(k \mid N)}
Last Step

\[
\prod_{k \in d \cap q} \frac{p(k \mid R)}{p(k \mid N)} \ast \prod_{k \in q \setminus d} \frac{1 - p(k \mid R)}{1 - p(k \mid N)}
\]

All matching terms \quad All non-matching terms

- Some reformulating (duplicating the terms in q)

\[
= \prod_{k \in d \cap q} \frac{p(k \mid R) \ast (1 - p(k \mid N)) \ast (1 - p(k \mid R)) \ast \prod_{k \in q \setminus d} \frac{1 - p(k \mid R)}{1 - p(k \mid N)}}{p(k \mid N) \ast (1 - p(k \mid R) \ast (1 - p(k \mid N)) \ast \prod_{k \in q \setminus d} \frac{1 - p(k \mid R)}{1 - p(k \mid N)}}
\]

All matching terms \quad All query terms
Continuation 2

- Obviously, the last term is identical for all docs. Thus

\[ \text{sim}(d, q) \approx \prod_{k \in d \cap q} \frac{p(k \mid R) \ast (1 - p(k \mid N))}{p(k \mid N) \ast (1 - p(k \mid R))} \]

- \( \text{sim}(d, q) \) = probability of a document comprising the terms of \( d \) being relative to query \( q \)
- But: Computing \( \text{sim}(d, q) \) requires knowledge of \( R \) and \( N \)
  - If \( R \) and \( N \) were known, we can estimate \( p(k \mid R) / p(k \mid N) \) using maximum likelihood
  - This means: Computing relative frequencies of terms in \( R/N \)
- In reality, we actually want to find \( R \) and \( N \)
Back to Reality

• Idea: Approximation using an iterative process
  – Start with some “educated guess” for R (and set N=D\R)
    • E.g. retrieve all docs containing at least one word from q
  – Compute probabilistic ranking of all docs wrt q based on first guess
    • Here it is important to focus on terms in q
  – Chose relevant docs (by user feedback) or hopefully relevant docs
    (by selecting the top-r docs)
  – This gives new sets R and N
    • If top-r docs are chosen, we may chose to only change probabilities of
      terms in R (and disregard the questionable negative information)
  – Compute new term scores and new ranking
  – Iterate until satisfied

• [Variant of the Expectation Maximization Algorithm (EM)]
Initialization

- **Typical simplifying assumptions** for the start
  - Terms in non-relevant docs are equally distributed: \( p(k|N) \sim \frac{df_k}{|D|} \)
  - \( p(k|R) \) is constant, e.g., \( p(k|R)=0.5 \)
  - Much less computation, less weight to presumably unstable first values

- **Iterations**: Assume we have a new \( R/N \)

\[
P(k \mid R) = \frac{|\{d \mid k \in d, d \in R\}|}{|R|}
\]

\[
P(k \mid N) = \frac{df_k - |\{d \mid k \in d, d \in R\}|}{|D| - |R|}
\]
## Example Data

<table>
<thead>
<tr>
<th>Text</th>
<th>verkauf</th>
<th>haus</th>
<th>italien</th>
<th>gart</th>
<th>miet</th>
<th>blüh</th>
<th>woll</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Wir verkaufen Häuser in Italien</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Häuser mit Gärten zu vermieten</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Häuser: In Italien, um Italien, um Italien herum</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Die italienischen Gärtner sind im Garten</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Um unser italiensches Haus blüht's</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6 Wir verkaufen Blühendes</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Q Wir wollen ein Haus mit Garten in Italien mieten</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Example:
Initialization:

\[ \text{sim}(d,q) \approx \prod_{k \in d \cap q} \frac{p(k \mid R) \ast (1 - p(k \mid N))}{p(k \mid N) \ast (1 - p(k \mid R))} \]

- All docs with at least one word from q
  - \( R = \{1, 2, 3, 4, 5\}, \ N = \{6\} \)
- Start with initial estimations
  - \( p(k \mid R) = 0.5, \ p(k \mid N) = \frac{df_k}{|D|} \rightarrow p(\text{verkauf} \mid N) = p(\text{blüh} \mid N) = 2/6 \)
  - **Smoothing**: If \( p(k \mid X) = 0 \), set \( p(k \mid X) = 0.01 \)
- Compute initial ranking
  - \( \text{sim}(1,q) = \frac{p(\text{haus} \mid R) \ast (1 - p(\text{haus} \mid N)) \ast p(\text{italien} \mid R) \ast (1 - p(\text{italien} \mid N))}{p(\text{haus} \mid N) \ast (1 - p(\text{haus} \mid R)) \ast p(\text{italien} \mid N) \ast (1 - p(\text{italien} \mid R))} = 0.01 \ast (1 - 0.5) \ast 0.01 \ast (1 - 0.5) = 9801 \)
  - \( \text{sim}(2,q) = 970299 \)
  - \( \text{sim}(3,q) = \text{sim}(4,q) = \text{sim}(5,q) = 9801 \)
  - \( \text{sim}(6,q) = 0 \)
Example: Adjustment

- Let’s use the **top-2 docs** as new R
  - Second chosen arbitrarily among 1,3,4,5
  - R={1,2}, N={3,4,5,6}

- Adjust scores
  - \( p(\text{verkauf}|R) = .5 \), \( p(\text{verkauf}|N) = (2-1)/(6-2) = 1/4 \)
  - \( p(\text{haus}|R) = 1\sim.99 \), \( p(\text{haus}|N) = (4-2)/(6-2) = 2/4 \)
  - \( p(\text{italien}|R) = .5 \), \( p(\text{italien}|N) = (4-1)/(6-2) = 3/4 \)
  - \( p(\text{gart}|R) = .5 \), \( p(\text{gart}|N) = (2-1)/(6-2) = 1/4 \)
  - \( p(\text{miet}|R) = .5 \), \( p(\text{miet}|N) = (1-1)/(6-2) = 0\sim0.01 \)
Example: Re-Ranking

\[ \text{sim}(d,q) \approx \prod_{k \in d \cap q} \frac{p(k \mid R) \cdot (1 - p(k \mid N))}{p(k \mid N) \cdot (1 - p(k \mid R))} \]

- New ranking
  - \( \text{sim}(1,q) = p(\text{haus} \mid R) \cdot (1 - p(\text{haus} \mid N)) \cdot p(\text{italien} \mid R) \cdot (1 - p(\text{italien} \mid N)) \cdot p(\text{haus} \mid N) \cdot (1 - p(\text{haus} \mid R)) \cdot p(\text{italien} \mid N) \cdot (1 - p(\text{italien} \mid R)) \)
    
    \[ = \ldots \]
  - \( \text{sim}(2,q) = \ldots \)
  - \( \text{sim}(3,q) = \ldots \)
  - \( \ldots \)
Pros and Cons

• Advantages
  – **Sound probabilistic framework**
    • Note that VSM is strictly heuristic – what is the justification for those distance measures?
  – Results *converge* to most probable docs
    • Under the assumption that relevant docs are similar by sharing term distributions that are different from distributions in irrelevant docs

• Disadvantages
  – First guesses often are pretty bad – slow convergence
  – Terms cannot be weighted ($w_{ij} \in \{0,1\}$)
  – Assumes statistical independence of terms (as most methods)
  – “Has *never worked convincingly better* in practice” [MS07]
Probabilistic Model versus VSM with Rel. Feedback

- Published 1990 by Salton & Buckley
- **Comparison** based on various corpora
- Improvement after 1 feedback iteration
- Probabilistic model (BIR) in general **worse than VSM+rel feedback (IDE)**
  - Probabilistic model does not weight terms in documents
  - Probabilistic model does not allow to weight terms in queries
Content of this Lecture

- IR Models
- Boolean Model
- Vector Space Model
- Relevance Feedback in the VSM
- Probabilistic Model
- Latent Semantic Indexing
- Other IR Models
Latent Semantic Indexing

- We so-far ignored **semantic relationships** between terms
  - Homonyms: bank (money, river)
  - Synonyms: House, building, hut, villa, ...
  - Hyperonyms: officer – lieutenant

- **Idea of Latent Semantic Indexing (LSI)**
    - >5000 citations!
  - Map (many) terms into (fewer) **semantic concepts**
    - Which are hidden (or “latent”) in the docs
  - Compare docs and query in **concept space** instead of term space

- One **big advantage**: Can find docs that don’t even contain the query terms
Terms and Concepts

- Concepts are **more abstract** than terms
- Concepts are (more or less) related to terms and to docs
- LSI finds “concepts” automatically by **matrix manipulations**
  - A concept will be a set of frequently co-occurring terms
  - Concepts from LSI cannot be “spelled out”, but are matrix columns

Quelle: K. Aberer, IR
Term-Document Matrix

- Definition

*The term-document matrix* $M$ for docs $D$ and terms $K$ has $n=|D|$ columns and $m=|K|$ rows. $M[i,j]=1$ iff document $d_j$ contains term $k_i$.

- Works equally well for TF or TF*IDF values

<table>
<thead>
<tr>
<th>Begriff</th>
<th>Dokument 1</th>
<th>Dokument 2</th>
<th>Dokument 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Document</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Retrieval</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Information</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Theory</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Database</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Indexing</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Computer</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Term-Document Matrix and VSM

- The matrix we used in VSM was a transposed document-term matrix \((=M^t)\)
- Having \(M\), we can compute the vector \(v\) containing the VSM-scores of all docs given \(q\) as \(v=M^t \cdot q\)
  - Ignoring score normalization
What to do with a Term-Document Matrix

• M is not just a comfortable way of representing the term vectors of all documents
  – M is a matrix
  – Linear Algebra offers many ways to manipulate matrices

• In the following, we approximate M by a M’
  – M’ should be smaller than M (in a certain sense)
    • Less dimensions; faster computations
  – M’ should abstract from terms to concepts
    • The less dimensions capture the least frequent co-occurrences
  – M’ should be such that \( M^t * q \approx M' * q \)
    • Produce the least error among all M’ of the same dimension

• Note: We only sketch LSI
Term and Document Correlation

- $M \cdot M^t$ is called the term correlation matrix
  - Has $|K|$ columns and $|K|$ rows
  - "Similarity" of terms: how often do they co-occur in a doc?

- $M^t \cdot M$ is called the document correlation matrix
  - Has $|D|$ columns and $|D|$ rows
  - "Similarity" of docs: how many terms do they share?

- Example

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

$M$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

$M^t$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Term correlation matrix
Some Lineare Algebra to Remember

- Let $M$ be a matrix
- The **rank** of $M$ ($r$) is the maximal number of linear independent rows of $M$ (its dimension)
- If we have $M\lambda - \lambda x = 0$ for $x \neq 0$, then $\lambda$ is called an **Eigenwert** of $M$ and $x$ is his associated **Eigenvector**
  - Eigenvectors/-werte are useful for many things
  - In particular, one can show that a matrix $M$ can be transformed into a **diagonal matrix** $L$ with $L=U^{-1}M*U$ with $U$ formed from the Eigenvectors of $M$, but only iff $M$ has “enough” Eigenvectors
  - Such $L$ is called **similar to** $M$; $L$ represents $M$ in another vector space, based on another basis
  - $L$ can be used in many cases instead of $M$ and is easier to handle
  - However, our $M$ usually will **not have** “enough” Eigenvectors
Singular Value Decomposition (SVD)

• SVD is a method to decompose any matrix in the following way: \( M = X \cdot S \cdot Y^t \)
  
  – S is the diagonal matrix of the singular values of M in descending order and has size \( r \times r \)
  
  – X is the matrix of Eigenvectors of \( M \cdot M^t \)
  
  – Y is the matrix of Eigenvectors of \( M^t \cdot M \)
  
  – This decomposition is unique and can be computed in \( O(r^3) \)
Example

- Assume for now $M$ is quadratic and has full rank
  - Example for $r = |K| = |D| = 3$

$$
\begin{bmatrix}
M_{11} & M_{12} & M_{13} \\
M_{21} & \ldots & \ldots \\
M_{31} & \ldots & M_{33}
\end{bmatrix} = 
\begin{bmatrix}
x_{11} & \ldots & \ldots \\
\ldots & \ldots & \ldots \\
\ldots & \ldots & x_{33}
\end{bmatrix} 
\cdot 
\begin{bmatrix}
s_{11} & 0 & 0 \\
0 & s_{22} & 0 \\
0 & 0 & s_{33}
\end{bmatrix} 
\cdot 
\begin{bmatrix}
y_{11} & \ldots & \ldots \\
\ldots & \ldots & \ldots \\
\ldots & \ldots & y_{33}
\end{bmatrix}
$$

- $M_{11} = (x_{11} * s_{11} + x_{12} * s_{12} + x_{13} * s_{13}) * y_{11} +$
  $(x_{11} * s_{21} + x_{12} * s_{22} + x_{13} * s_{23}) * y_{21} +$
  $(x_{11} * s_{31} + x_{12} * s_{32} + x_{13} * s_{33}) * y_{31}$
  $= x_{11} * s_{11} * y_{11} + x_{12} * s_{22} * y_{21} + x_{13} * s_{33} * y_{31}$
- $M_{12} = \ldots$
General Case

- $M$ not quadratic; $r < \min(|K|, |R|)$
  - All sums range from 1 to $r$

- LSI idea: What if we stop the sums earlier, at some $s<r$?
  - $s_{ii}$ are sorted by descending value
  - Aggregating only over the first $s$ $s_{ii}$-values captures “most” of $M$
Approximating $M$

- $S$ can be used to approximate $M$
- Fix some $s<r$; Compute $M_s = X_s \cdot S_s \cdot Y_s^t$
  - $X_s$: First $s$ columns in $X$
  - $S_s$: First $s$ columns and first $s$ rows in $S$
  - $Y_s$: First $s$ rows in $Y$
- $M_s$ has the same size as $M$, but different values
  - For LSI, we don’t need to compute $M_s$, but only need $X_s$, $S_s$ and $Y_s$
s-Approximations

- Since the $s_{ii}$ are sorted in decreasing order
  - The approximation is the better, the larger $s$
  - The computation is the faster, the smaller $s$

- LSI: Only consider the top-$s$ singular values
  - $s$ must be small enough to filter out noise and to provide “semantic reduction”
  - $s$ must be large enough to represent the diversity in the documents
  - Typical value: 200-500

- Optimality: After SVD, $M'$ is the matrix where $||M-M'||_2$ is the smallest
Geometric Interpretation of SVD

- M is a linear transposition in a vector space
- X,Y can be seen as coordination transformations, S is a linear scaling
- X transforms M into a vector space where its transposition can be represented as a linear scaling (and Y transforms it back into the vector space of M)
- s-approximation
  - M is transformed into a vector space of lower dimension such that the new dimensions capture the most important variations in M
  - Distances between vectors are preserved as much as possible
- Universal method: LSI has many more applications than IR
LRI for Information Retrieval

- We map document vectors from a m-dimensional space into a s-dimensional space
  - Approximated docs (still) are represented by columns in $Y_s^t$
- Variations between document vectors are determined by the number of terms they have in common
  - The more terms in common, the smaller the distance
- SVD tries to preserve these distances
- To this end, it (in a way) maps frequently co-occurring terms to the same dimensions
  - Because frequently co-occurring terms have little impact on distance
- Frequently co-occurring terms can be interpreted as concepts
  - But they cannot easily be “named”
  - Also, we cannot simply determine the terms that are mapped into a new dimension – it is always a bit of everything (a linear combination)
Query Evaluation

- After LSI, docs are represented by columns in $Y_s^t$
- How can we compute the distance between a query and a doc in concept space?
  - We first need to represent q in concept space
  - Assume q as a new column in M
    - Of course, we can transform M offline, but need to transform q online
  - This would generate a new column in $Y_s^t$
  - To only compute this column, we apply the same transformations to q as we did to all other columns of M
  - With a little algebra, we get: $q' = q^t \cdot X_s \cdot S_s^{-1}$
  - This vector is compared to the doc vectors as usual
Example: Term-Document Matrix

- Taken from Mi Islita: “Tutorials on SVD & LSI”
  - Who took it from the Grossman and Frieder book

<table>
<thead>
<tr>
<th>Terms</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>arrived</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>damaged</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>delivery</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fire</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gold</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>in</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>of</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>shipment</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>silver</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>truck</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Query: „gold silver truck“
Singular Value Decomposition

\[ M = X \cdot S \cdot Y^t \]

\[ X = \begin{bmatrix}
-0.4201 & 0.0748 & -0.0460 \\
-0.2995 & -0.2001 & 0.4078 \\
-0.1206 & 0.2749 & -0.4538 \\
-0.1576 & -0.3046 & -0.2006 \\
-0.1206 & 0.2749 & -0.4538 \\
-0.2626 & 0.3794 & 0.1547 \\
-0.4201 & 0.0748 & -0.0460 \\
-0.4201 & 0.0748 & -0.0460 \\
-0.2626 & 0.3794 & 0.1547 \\
-0.3151 & -0.6093 & -0.4013 \\
-0.2995 & -0.2001 & 0.4078
\end{bmatrix} \]

\[ S = \begin{bmatrix}
4.0989 & 0.0000 & 0.0000 \\
0.0000 & 2.3616 & 0.0000 \\
0.0000 & 0.0000 & 1.2737
\end{bmatrix} \]

\[ Y = \begin{bmatrix}
-0.4945 & 0.6492 & -0.5780 \\
-0.6458 & -0.7194 & -0.2556 \\
-0.5817 & 0.2469 & 0.7750
\end{bmatrix} \]

\[ Y^t = \begin{bmatrix}
-0.4945 & -0.6458 & -0.5817 \\
0.6492 & -0.7194 & 0.2469 \\
-0.5780 & -0.2556 & 0.7750
\end{bmatrix} \]
A Two-Approximation (s=2)

\[
X_2 = \begin{bmatrix}
-0.4201 & 0.0748 \\
-0.2996 & -0.2001 \\
-0.1206 & 0.2749 \\
-0.1576 & -0.3046 \\
-0.1206 & 0.2749 \\
-0.2626 & 0.3794 \\
-0.4201 & 0.0748 \\
-0.4201 & 0.0748 \\
-0.2626 & 0.3794 \\
-0.3151 & -0.6093 \\
-0.2996 & -0.2001
\end{bmatrix}
\]

\[
S_2 = \begin{bmatrix}
4.0989 & 0.0000 \\
0.0000 & 2.3616
\end{bmatrix}
\]

\[
Y_2 = \begin{bmatrix}
-0.4945 & 0.6492 \\
-0.6458 & 0.7194 \\
-0.5817 & 0.2469
\end{bmatrix}
\]

\[
Y_2^t = \begin{bmatrix}
-0.4945 & -0.6458 & -0.5817 \\
0.6492 & -0.7194 & 0.2469
\end{bmatrix}
\]

\[\uparrow \quad \uparrow \quad \uparrow \]

\[d_1 \quad d_2 \quad d_3\]
Transforming the Query

\[ q' = q^t \cdot X_2 \cdot S_2^{-1} \]

\[
q' = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
-0.4201 & 0.0748 \\
-0.2995 & -0.2001 \\
-0.1206 & 0.2749 \\
-0.1576 & -0.3046 \\
-0.1206 & 0.2749 \\
-0.2626 & 0.3794 \\
-0.4201 & 0.0740 \\
-0.4201 & 0.0746 \\
-0.2626 & 0.3794 \\
-0.3151 & -0.6083 \\
-0.2995 & -0.2001
\end{bmatrix}
\begin{bmatrix}
1 \\
\frac{1}{0.0000} \\
\frac{1}{2.3616}
\end{bmatrix}
\]

\[ = \begin{bmatrix}
-0.2140 \\
-0.1021
\end{bmatrix} \]
Computing the Cosine of the Angle

\[
\text{sim}(q, d) = \frac{q \cdot d}{\|q\| \cdot \|d\|}
\]

\[
\text{sim}(q, d_1) = \frac{(-0.2140) \cdot (-0.4945) + (-0.1821) \cdot (0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \cdot \sqrt{(-0.4945)^2 + (0.6492)^2}} = 0.0541
\]

\[
\text{sim}(q, d_2) = \frac{(-0.2140) \cdot (-0.6458) + (-0.1821) \cdot (-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \cdot \sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910
\]

\[
\text{sim}(q, d_3) = \frac{(-0.2140) \cdot (-0.5817) + (-0.1821) \cdot (0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \cdot \sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478
\]
Visualization of Results in 2D

<table>
<thead>
<tr>
<th>Terms</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>arrived</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>damaged</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>delivery</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fire</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gold</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>in</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>of</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>shipment</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>silver</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>truck</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

\[ M = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad q = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \]

![Diagram showing visualization of results in 2D](image-url)
Pros and Cons

• **Pro**
  – *Made it into practice*, used by real search engines
  – Speed-up through computation with less dimensions
  – Increases recall (and usually decreases precision)

• **Contra**
  – Computing SVD is expensive
    • *Fast approximations* exist, especially for extremely sparse matrices
    • Use stemming, stop-word removal etc. to shrink the original matrix
  – Ranking requires less dimensions than |D|, but more than |q|
    • Every query needs to be mapped first – turns a few keywords into a s-dimensional vector
    • We **cannot simply index** the “concepts” of Ms using inverted files etc.
    • Thus, LSI needs other techniques than indexing (read: *lots of memory*)
Content of this Lecture

- IR Models
- Boolean Model
- Vector Space Model
- Relevance Feedback in the VSM
- Probabilistic Model
- Latent Semantic Indexing
- Other IR Models
Extended Boolean Model

• One critique to the Boolean Model: If one term out of 10 is missing, the result is the same as if 10 were missing

• Idea: Measure “distance” for each conjunctive / disjunctive subterm of the query expression to the document
  – Example: X-ary AND: use a projection into x-dim space
  – Query expression is \((1,1,1,...,1)\)
  – Doc is \((a_1,a_2,...,a_x)=(0/1?,0/1?,...)\)
  – Similarity is distance between these two points
  – Similar formulas for OR and NOT

• Using the appropriate definition of distance, the extended Boolean model may mimic both the Boolean and the VSM
Generalized Vector Space Model

• One critique to the VSM: Terms are not independent
• Thus, term vectors cannot be assumed to be orthogonal
• Generalized Vector Space Model
  – Build a much larger vector space with $2^{|K|}$ dimensions
  – Each dimension (“minterm”) stands for all docs containing a particular set of terms
  – Minterms are not orthogonal but correlated by term co-occurrences
  – Convert query and docs into minterm space
  – Finally, $\text{sim}(q, d)$ is the cosine of the angel in minterm space
• Nice theory, includes term co-occurrence, much more complex than ordinary VSM, no proven advantage