Text Analytics

Modeling Information Retrieval 2

Ulf Leser
IR Models

- Modeling: How is relevance judged?
  - What model of relevancy is assumed?
Boolean Model

- Simple model based on set theory
- Queries are specified as **Boolean expressions**
  - Terms are atoms
  - Terms are connected by AND, OR, NOT (XOR, ...)
- There are no real weights; $w_{ij} \in \{0,1\}$
  - All terms contribute equally to the content of a doc
- Relevance of a document is either 0 or 1
  - Computation by Boolean operations on the doc and query vector
  - Represent $q$ as **set $S$ of vectors over all terms**
    - Example: $q = k_1 \land (k_2 \lor \neg k_3)$ $\Rightarrow$ $S = \{(1,1,1),(1,1,0),(1,0,0)\}$
    - $S$ describes all possible combinations of keywords that make a doc relevant
  - Document $d$ is relevant for $q$ iff $v_d \in S$
Properties

• Simple, clear semantics, widely used in (early) systems

• Disadvantages
  – **No partial matching** (expression must evaluate to true)
    • Suppose query \( k_1 \land k_2 \land \ldots \land k_9 \)
    • A doc \( d \) with \( k_1 \land k_2 \ldots k_8 \) is as much refused as one with none of the terms
  – **No ranking** (extensions exist, but not satisfactory)
  – Query terms cannot be weighted
  – Average users don’t like Boolean expressions
    • Search engines: “+bill +clinton –lewinsky tax”

• Results: Often unsatisfactory
  – Too many documents (too few restrictions)
    • Without ranking
  – Too few documents (too many restrictions)
Vector Space Model

  - Historically, a breakthrough in IR
  - Still most popular model today

- General idea
  - Fix a vocabulary $K$
    - Typically the set of all different terms in $D$
  - View each doc and each query as a point in a $|K|$-dim space
  - Rank docs according to distance from the query

- Main advantages
  - Ability to rank docs (according to relevance)
  - Natural capability for partial matching
Preliminaries 1

• Definition

Let $D$ be a document collection, $K$ be the set of all terms in $D$, $d \in D$ and $k \in K$.

- The term frequency $tf_{dk}$ is the frequency of $k$ in $d$
- The document frequency $df_k$ is the number of docs in $D$ containing $k$
  - Or sometimes the number of occurrences of $k$ in $D$
- The inverse document frequency $idf_k = |D| / df_k$
  - Actually, it should rather be called the inverse relative document frequency
  - In practice, one usually uses $idf_k = \log(|D| / df_k)$

• Remarks

- $idf$ is a popular weighting term, therefore a proper definition
- Clearly, $tf_{dk}$ is a natural measure for the weight $w_{dk}$
  - But not the only or best one
The Simplest Approach

- Co-ordinates are set as **term frequency**
- Distance is measured as the **cosine of the angle** between doc \( d \) and query \( q \)
  - Wait for Euclidean distance

\[
sim(d, q) = \cos(v_d, v_q) = \frac{v_d \circ v_q}{|v_d| \cdot |v_q|} = \frac{\sum (v_d[i] \cdot v_q[i])}{\sqrt{\sum v_q[i]^2} \cdot \sqrt{\sum v_d[i]^2}}
\]

- Properties
  - Including term weights increases retrieval performance
  - Computes a (seemingly reasonable) ranking
  - Also returns partial matches

Can be dropped for ranking
Example Data

- Assume stop word removal (wir, in, ...) and stemming (verkaufen -> kauf, wollen -> woll, ...)

<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>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>Häuser mit Gärten zu vermieten</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Häuser: In Italien, um Italien, um Italien herum</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Die italienschen Gärtner sind im Garten</td>
<td></td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Der Garten in unserem italienschen Haus blüht</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>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>
Ranking with Length Normalization

\[ sim(d, q) = \frac{\sum (v_q[i] \cdot v_d[i])}{\sqrt{\sum v_d[i]^2}} \]

- \( sim(d_1, q) = \frac{(1*0+1*1+1*1+0*1+0*1+0*0+0*1)}{\sqrt{3}} \approx 1.15 \)
- \( sim(d_2, q) = \frac{(1+1+1)}{\sqrt{3}} \approx 1.73 \)
- \( sim(d_3, q) = \frac{(1+3)}{\sqrt{10}} \approx 1.26 \)
- \( sim(d_4, q) = \frac{(1+2)}{\sqrt{5}} \approx 1.34 \)
- \( sim(d_5, q) = \frac{(1+1+1)}{\sqrt{4}} \approx 1.5 \)

Q: Wir wollen ein Haus mit Garten in Italien mieten

No good?
Improved Scoring: TF*IDF

- One obvious problem: The longer the document, the higher the chances of being ranked high
  - Solution: Normalize on document length (also yields $0 \leq w_{ij} \leq 1$)

$$w_{ij} = \frac{tf_{ij}}{|d_i|} = \frac{tf_{ij}}{\sum_{k=1..|d_i|} tf_{ik}}$$

- Second obvious problem: Some terms are just everywhere in D and should be scored less
  - Solution: Use IDF scores

$$w_{ij} = \frac{tf_{ij}}{|d_i|} * idf_j$$
Metaphor: Comparison to Clustering

- **Clustering**: Find groups of objects that are close to each other and far apart from all others

- Rephrase the IR problem
  - A query partitions \( D \) in two clusters: relevant \( R \) / not relevant \( N \)
  - \( q \) should be in the heard of \( R \): similarity
  - Docs in \( R \) should be close to each other: large \( w_{ij}q_i \) values
  - Docs in \( R \) should be far apart from other points: idf
  - Goal of scoring: Balance between intra- and inter-cluster similarity
Shortcomings

- We assumed that all terms are independent, i.e., that their vectors are orthogonal
  - Clearly wrong: terms are semantically close or not
    - The appearance of “red” in a doc with “wine” doesn’t mean much
    - But “wine” is a non-zero match for “drinks”
  - Extension: Topic-based Vector Space Model
- No treatment homonyms (as for most other methods)
  - We would need to split words into their senses
  - See word disambiguation (later)
- Order independent
  - But order carries semantic meaning (object? subject?)
  - Order important for disambiguation
Relevance Feedback

• User judges the current answers

• Basic assumptions
  – Relevant docs are somewhat similar to each other – the common core should be emphasized
  – Irrelevant docs are different from relevant docs – the differences should be deemphasized

• System adapts to feedback by
  – Query expansion: Add new terms to the query
    • From the relevant documents
  – Term re-weighting: Assign new weights to terms
    • From the relevant documents
Rocchio Algorithm

- Usually we do not know \( D_r \)
- Let \( R (N) \) be the set docs marked as relevant (irrelevant)
- **Adapt query vector** as follows
  \[
  v_{q_{\text{new}}} = \alpha * v(q) + \beta * \frac{1}{|R|} \sum_{d \in R} d - \gamma * \frac{1}{|N|} \sum_{d \in N} d
  \]
- This implies query expansion and term re-weighting
  - How to choose \( \alpha, \beta, \gamma \)?
- **Alternative**
  \[
  v_{q_{\text{new}}} = \alpha * v(q) + \beta * \frac{1}{|R|} \sum_{d \in R} d - \gamma \{d \mid d = \text{arg max} \{sim(q, d)\} \}
  \]
  - Intuition: Non-relevant docs are heterogeneous and tear in every direction – better to only take the worst
Results

- **Advantages**
  - **Improved results** (many positive studies) compared to single queries
  - Users need not generate new queries themselves
  - Iterative process converging to the best possible answer
  - Especially helpful for increasing recall (due to query expansion)

- **Disadvantages**
  - Requires some work by the user
    - Excite: Only 4% of users used relevance feedback (“more of this” button)
  - Writing a new query based on returned results might be faster (and easier and more successful) than classifying results
  - Based on the assumption that relevant docs are similar
    - What if user searches for all meanings of “jaguar”?  
  - Query is very long already after one iteration – makes retrieval slow (why?)

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.
  - Best compared to VSM with relevance feedback.
- Assume there is a subset $R \subseteq D$ which contains all (and only) relevant documents for q.
- For each document, we want to compute the probability $p(R|d)$ that d belongs to R (for q).
- Then, we use odds-scores

$$\text{rel}(d, q) \sim \text{sim}(d, q) = \frac{p(R | d)}{p(D \setminus R | d)}$$
Binary Independence Model

• Bayes (with $N=D\setminus R$)

$$sim(d,q) = \frac{p(R \mid d)}{p(N \mid d)} = \frac{p(d \mid R) \cdot p(R)}{p(d \mid N) \cdot p(N)} \sim \frac{p(d \mid R)}{p(d \mid N)}$$

- $p(R)$ is the relative frequency of relevant docs in $D$
- $p(d \mid R)$ is the random experiment of drawing $d$ when drawing from $R$
- $p(R)$ and $p(N)$ are independent from $d$ – thus, both are constant for $q$ and irrelevant for ranking documents

• Representing docs by their terms and assuming term independence

$$sim(d,q) = \frac{\prod_{k \in d} p(k \mid R) \cdot \prod_{k \not\in d} p(-k \mid R)}{\prod_{k \in d} p(k \mid N) \cdot \prod_{k \not\in d} p(-k \mid N)}$$
Continuation

- Rephrasing using terms in q

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

- Focusing on the query terms
  - All others are not relevant for ranking docs wrt to q

\[
\ldots \approx \prod_{k \in d \cap q} \frac{p(k \mid R)}{p(k \mid N)} \prod_{k \in q \setminus d} \frac{p(\neg k \mid R)}{p(\neg k \mid N)} = \prod_{k \in d \cap q} \frac{p(k \mid R)}{p(k \mid N)} \prod_{k \in q \setminus d} \frac{1 - p(k \mid R)}{1 - p(k \mid N)}
\]
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**  **All non-matching terms**

- Some clever reformulating (duplicating the terms in q)

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

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

**All matching terms**  **All query terms**
Continuation 2

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

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

• \( sim(d, q) = \) probability of a document comprising the terms of \( d \) being relative to query \( q \)

• If we **knew** \( R \) and \( N \)
  – Life would be easy
  – Using max likelihood, we approximate all probabilities by counting term occurrences
Back to Reality

- But we don’t know R and N
- Idea: Approximation using an iterative process
  - Start with some “educated guess” for R (and set N=D\R)
  - Compute probabilistic ranking of all docs wrt q based on first guess
  - 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

• Simplifying assumptions for the start
  – All terms in non-relevant docs are equally distributed in all docs. Thus, \( p(k|N) \sim \frac{df_k}{|D|} \)
  – \( p(k|R) \) is constant, e.g., \( p(k|R) = 0.5 \)

• Start process
  – Retrieve all docs containing at least one term from \( q \)
  – Compute probabilistic ranking using the simplified assumptions defined above
  – Determine new \( R \)
  – Adapt scores by counting occurrences of \( k \)'s in \( R \) and \( N \)

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

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

<table>
<thead>
<tr>
<th></th>
<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</td>
<td>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</td>
<td>Häuser mit Gärten zu vermieten</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Häuser: In Italien, um Italien, um Italien herum</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Die italienschen 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</td>
<td>Um unser italiensches Haus blüht's</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Wir verkaufen Blühendes</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Q</td>
<td>Wir wollen ein Haus mit Garten in Italien mieten</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Example: Initialization

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

- All docs containing at least one term from q
  - \( R = \{1,2,3,4,5\}, N = \{6\} \)
- Start with initial estimations
  - \( p(k \mid R) = 0.5, p(k \mid N) = \frac{\text{df}_k}{|D|} \rightarrow p(\text{verkauf} \mid N) = p(\text{blüh} \mid N) = \frac{2}{5} \)
  - Smoothing: If \( p(k \mid X) = 0 \), set \( p(k \mid X) = 0.01 \)
- Compute initial ranking
  - \( \text{sim}(1,q) = \frac{p(\text{haus} \mid R) \cdot (1 - p(\text{haus} \mid N)) \cdot p(\text{italien} \mid R) \cdot (1 - p(\text{italien} \mid N))}{p(\text{haus} \mid N) \cdot (1 - p(\text{haus} \mid R)) \cdot p(\text{italien} \mid N) \cdot (1 - p(\text{italien} \mid R))} = \frac{.5 \cdot (1 - 0.01) \cdot .5 \cdot (1 - 0.01)}{(0.01 \cdot (1 - 0.5) \cdot 0.01 \cdot (1 - 0.5))} = 9801 \)
  - \( \text{sim}(2,q) = 970299 \)
  - \( \text{sim}(3,q) = \text{sim}(4,q) = \text{sim}(5,q) = 9801 \)
  - \( \text{sim}(6,q) = 0 \)
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
  - \( p(\text{verkauf}|R) = 0.5 \)
  - \( p(\text{verkauf}|N) = (2-1)/(6-2) = 1/4 \)
  - \( p(\text{haus}|R) = 1 \sim 0.99 \)
  - \( p(\text{haus}|N) = (4-2)/(6-2) = 2/4 \)
  - \( p(\text{italien}|R) = 0.5 \)
  - \( p(\text{italien}|N) = (4-1)/(6-2) = 3/4 \)
  - \( p(\text{gart}|R) = 0.5 \)
  - \( p(\text{gart}|N) = (2-1)/(6-2) = 1/4 \)
  - \( p(\text{miet}|R) = 0.5 \)
  - \( p(\text{miet}|N) = (1-1)/(6-2) = 0 \sim 0.01 \)

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

\[
P(k | N) = \frac{df_k - |\{d | k \in d, d \in N\}|}{|D| - |R|}
\]
Example: Re-Ranking

Re-Ranking

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

- New ranking
  - \( \text{sim}(1,q) = p(\text{haus}\mid R) \ast (1 - p(\text{haus}\mid N)) \ast p(\text{italien}\mid R) \ast (1 - p(\text{italien}\mid N)) \)
    
    \[= \cdots\]
  - \( \text{sim}(2,q) = \cdots\)
  - \( \text{sim}(3,q) = \cdots\)
  - \( \text{sim}(4,q) = \cdots\)
  - \( \text{sim}(5,q) = \cdots\)
  - \( \text{sim}(6,q) = \cdots\)
  - \( \text{sim}(Q,q) = 1 \)
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 are pretty bad – slow convergence
  – Terms are not weighted ($w_{ij} \in \{0,1\}$), as in the Boolean model
  – Assumes statistical independence of terms (as most methods)
  – Efficient implementation?
  – “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

<table>
<thead>
<tr>
<th>eingesetzte Methode</th>
<th>CACM</th>
<th>CISO</th>
<th>CRAN</th>
<th>INSPEC</th>
<th>MED</th>
</tr>
</thead>
<tbody>
<tr>
<td>initiale Anfrage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.1459</td>
<td>0.1184</td>
<td>0.1156</td>
<td>0.1368</td>
<td>0.3346</td>
</tr>
<tr>
<td>IDE (dec hi)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.2704</td>
<td>0.1742</td>
<td>0.3011</td>
<td>0.2140</td>
<td>0.6305</td>
</tr>
<tr>
<td>Verbesserung</td>
<td>+86%</td>
<td>+47%</td>
<td>+160%</td>
<td>+56%</td>
<td>+88%</td>
</tr>
<tr>
<td>ausgewählte Termene</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.2479</td>
<td>0.1924</td>
<td>0.2498</td>
<td>0.1976</td>
<td>0.6218</td>
</tr>
<tr>
<td>Verbesserung</td>
<td>+70%</td>
<td>+63%</td>
<td>+116%</td>
<td>+44%</td>
<td>+86%</td>
</tr>
<tr>
<td>BIR-Modell</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mit allen Termene</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.2289</td>
<td>0.1436</td>
<td>0.3108</td>
<td>0.1621</td>
<td>0.5972</td>
</tr>
<tr>
<td>Verbesserung</td>
<td>+57%</td>
<td>+21%</td>
<td>+169%</td>
<td>+19%</td>
<td>+78%</td>
</tr>
<tr>
<td>ausgewählte Termene</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.2224</td>
<td>0.1634</td>
<td>0.2120</td>
<td>0.1876</td>
<td>0.5643</td>
</tr>
<tr>
<td>Verbesserung</td>
<td>+52%</td>
<td>+38%</td>
<td>+83%</td>
<td>+37%</td>
<td>+69%</td>
</tr>
</tbody>
</table>

- Probabilistic mod. in general worse than VSM+rel feedback
  - 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 (Sketch with little Math)

- Until now, we were comparing terms by using equality
- Ignores **semantic relationships** between terms
  - Homonyms: bank (money, river)
  - Synonyms: House, building, hut, villa, ...
  - Hyperonyms: officer – lieutenant
  - Co-occurrence of terms in a given theme
- Idea of **Latent Semantic Indexing (LSI)**
  - Map terms into (less) **semantic concepts**
    - Which are hidden (or latent) in the docs
  - Represent and query in **concept space** instead of term space
- Finds 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

Quelle: K. Aberer, IR
Example [Hen07]

- **Query:** “IDF in computer-based information look-up”
- **Which docs are most relevant?**

**What we would expect**
- Doc 1: (retrieval, access, indexing) \~ lookup, (doc, database) \~ information
  - Good fit, though no single term matches
- Doc 2: information = information, computer = computer
  - Two direct matches, but theory?
- Doc 3: information = information, comp. = comp., retrieval \~ lookup
  - Two direct matches, one fuzzy match
  - doc1 > doc3 > doc2

**VSM delivers:** doc3 = doc2 > doc1
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 $d_i$ contains term $k_j$.

  – Alternatively, use TF or TF*IDF values

• Note: The matrix we used in the previous examples was a transposed document-term matrix

• Using $M$, we can compute the VSM-ranks of all docs given $q$ as $M^t \cdot q$ (ignoring normalization)
Singular Value Decomposition (SVD)

- We want to find the **most important components of** $M$
  - Let $r$ be the rank of $M$
- We compute a decomposition of $M$ into the following form:
  $$M = X \cdot S \cdot Y^t$$
  - $S$ is the diagonal $r \times r$ matrix of the **singular values** of $M$, sorted in dec. order
  - $X$ is the matrix of Eigenvectors derived from $M \cdot M^t$
  - $Y$ is the matrix of Eigenvectors derived from $M^t \cdot M$
  - This decomposition is unique and can be computed in $O(r^3)$
Note

- $M \cdot M^t$ is a matrix with $|K|$ columns and $|K|$ rows, the **term correlation** matrix
  - And $X$ are its Eigenvectors
- $M^t \cdot M$ is a matrix with $|D|$ columns and $|D|$ rows, the **document correlation** matrix
  - And $Y$ are its Eigenvectors
- Example

\[
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
A & 1 & 1 & 1 &  &  \\
B & 1 & 1 & 1 & 1 &  \\
C & 1 & 1 &  &  &  \\
D &  & 1 & 1 & & \\
\end{array}
\]

\[
\begin{array}{c|cccc}
A & B & C & D \\
\hline
1 & 1 & 1 &  &  \\
2 & 1 & 1 & 1 &  \\
3 & 1 & 1 & 1 &  \\
4 &  & 1 &  &  \\
5 & 1 & 1 &  &  \\
\end{array}
\]

\[
M \cdot M^t = \begin{array}{c|cccc}
A & B & C & D \\
\hline
A & 3 & 3 & 2 & 0 \\
B & 3 & 4 & 2 & 1 \\
C & 2 & 2 & 2 & 0 \\
D & 0 & 1 & 0 & 2 \\
\end{array}
\]

Term correlation matrix
Approximating M

- The $S_{ii}$ may be used to approximate $M$
- Compute $M_s = X_s \cdot S_s \cdot Y_s^t$
  - First $s$ columns in $X$ -> $X_s$
  - First $s$ columns and last $s$ rows in $S$ -> $S_s$
  - First $s$ rows in $Y$ -> $Y_s$
- $M_s$ has the same size as $M$, but other (approximated) values
s-Approximations

- Since the $S_{ii}$ are sorted in decreasing order,
  - The approximation is the better, the larger $s$ ($s \leq r$)
  - The computation is the faster, the smaller $s$
  - The smaller $s$, the more concentration on the most highly correlated terms and documents = the concepts

- Idea of 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
Query Evaluation

- The similarity of any two docs can be computed as the cosine distance between their columns
  - $M_s^t \cdot M_s$ is the document correlation matrix in concept space
- **Approximated docs** are represented by their rows in $Y_s^t$
- How can we compute the distance between a query and a doc in concept space?
  - Easy
  - Assume $q$ a new row in $M$
  - We have to first apply the same transformations to $q$ as we did to all the docs
  - $q' = q^t \cdot X_s \cdot S_s^{-1}$
  - This vector may now by compared to the doc vectors as usual
Example: Term-Document Matrix

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

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

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.

Query: „gold silver truck“
Singular Value Decomposition

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

\[ X = \begin{bmatrix}
-0.4201 & 0.0748 & -0.0460 \\
-0.2995 & -0.2001 & 0.4078 \\
-0.1206 & 0.2749 & -0.4538 \\
-0.1576 & -0.3046 & -0.2006 \\
-0.1206 & 0.2749 & -0.4538 \\
-0.2626 & 0.3794 & 0.1547 \\
-0.4201 & 0.0748 & -0.0460 \\
-0.4201 & 0.0748 & -0.0460 \\
-0.2626 & 0.3794 & 0.1547 \\
-0.3151 & -0.6093 & -0.4013 \\
-0.2995 & -0.2001 & 0.4078
\end{bmatrix} \quad \begin{bmatrix}
4.0939 & 0.0000 & 0.0000 \\
0.0000 & 2.3616 & 0.0000 \\
0.0000 & 0.0000 & 1.2737
\end{bmatrix} \]

\[ S = \begin{bmatrix}
4.0939 & 0.0000 & 0.0000 \\
0.0000 & 2.3616 & 0.0000 \\
0.0000 & 0.0000 & 1.2737
\end{bmatrix} \]

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

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

\[
X_2 = \begin{bmatrix}
-0.4201 & 0.0748 \\
-0.2995 & -0.2001 \\
-0.1206 & 0.2749 \\
-0.1576 & -0.3046 \\
-0.1206 & 0.2749 \\
-0.2626 & 0.3794 \\
-0.4201 & 0.0748 \\
-0.4201 & 0.0748 \\
-0.2626 & 0.3794 \\
-0.3151 & -0.6093 \\
-0.2995 & -0.2001
\end{bmatrix}
\]

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

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

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

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

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

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

\[
q' = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
-0.4201 & 0.0748 \\
-0.2996 & -0.2001 \\
-0.1206 & 0.2749 \\
-0.1576 & -0.3046 \\
-0.1206 & 0.2749 \\
-0.2526 & 0.3794 \\
-0.4201 & 0.0748 \\
-0.4201 & 0.0748 \\
-0.2526 & 0.3794 \\
-0.3181 & -0.6093 \\
-0.2996 & -0.2001
\end{bmatrix}
\begin{bmatrix}
1 \\
4.0989 & 0.0000 \\
0.0000 & 2.3616
\end{bmatrix}
\]

\[ = \begin{bmatrix}
-0.2140 \\
-0.1821
\end{bmatrix} \]
Computing the Ranks

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

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

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

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

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

• Strong argument: Made it into practice, used by many search engines
• Pros
  – Speed-up through less computation in query evaluation
  – Generally leads to an increase in precision (rather than recall)
• Cons
  – Computing SVD is expensive
    • Fast approximations of SVD exist
    • Do not update with every new document
    • Use stemming, stop-word removal etc. to already shrink the original term-document matrix
  – Comparing the ranks is expensive
    • VSM etc. use inverted files (later) on the terms of the document
    • But we cannot simply index the “concepts” of $M_s$
    • 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 formula for OR and NOT
- Using the appropriate definition of distance, the extended Boolean model may mimic both the Boolean and the VSM
Fuzzy-Set IR

- In a fuzzy set $S$, each element $o$ is in the set $S$ with a certainty between 0 and 1
  - Membership function $m_S(o)$
- Operations on sets (one possible definition)
  - $m_{\neg S}(o) = 1 - m_S(o)$
  - $m_{S \cap T}(o) = \min(m_S(o), m_T(o))$
  - $m_{S \cup T}(o) = \max(m_S(o), m_T(o))$
- We define one **fuzzy set per term** in $D$
  - Thus, each doc is more or less a member of the set of the term
- Queries are like Boolean expressions
- Evaluation computes the membership function of the query expression (= combined term membership functions)
- Provides ranking; fuzzy-set people love it
Generalized Vector Space Model

• One critique to the VSM: Terms are not independent
• In reality, the term “one-vectors” should not be orthogonal
• Generalized Vector Space Model
  – Build a much larger vector space with $2^{\left|\mathcal{K}\right|}$ dimensions
    • Each dimension (“minterm”) stands for all docs containing a particular subset 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
Inference Networks

- A Bayesian Network is a **DAG of random variables**, where edges represent conditional probabilities.
- Used to capture which variables depend on which variables:
  - Instead of assuming mutual independence.
- Now: Documents, queries and terms are modeled as random variables.
- Support for a query are **aggregated over the network**.