Information Retrieval

Modeling Information Retrieval 3: Latent Semantic Indexing and Beyond

Ulf Leser

SHK Stelle zu besetzen

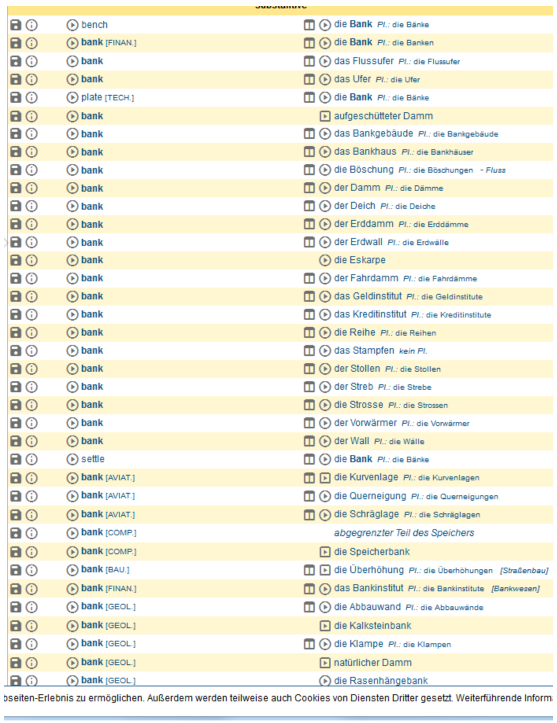
- Ab sofort
- Kenntnisse mind. 4. Semester Bachelor Informatik
- Mitarbeit in Forschung und Lehre
- Internationales Umfeld
- Spannende Themen im Umfeld
  - Verteilte skalierbare Datenanalyse
  - Biomedizinische Data Science, Machinelles Lernen
  - Text Mining
  - Effiziente Indexstrukturen

Content of this Lecture

- IR Models
- Boolean Model
- Vector Space Model
- Relevance Feedback in the VSM
- Probabilistic Model
- Latent Semantic Indexing
- Outlook: Word Semantics and Word Embeddings

Latent Semantic Indexing

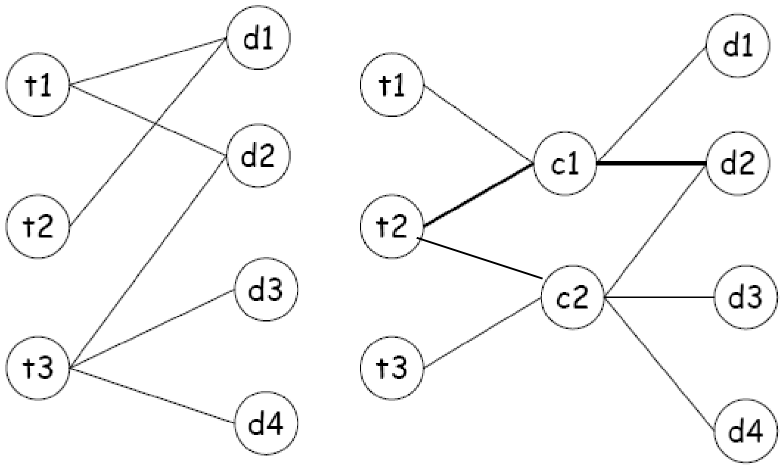
- We so-far ignored semantic relationships between terms
  - Homonyms: bank (money, river, place)
  - Synonyms: House, building, hut, villa, …
  - Hyperonyms: officer – lieutenant
- Latent Semantic Indexing (LSI)
  - Deerwester et al. (1990). "Indexing by latent semantic analysis." JASIS 41(6): 391-407.
    - 2011: ~7500 cit.; 2014: ~9400, 2018: ~13500
  - Map (many) terms into (fewer) semantic concepts
    - Discover the concepts hidden (“latent”) in the docs
  - Compare docs and query in concept space instead of term space
- May find docs that don’t contain a single query term



bench	die Bank *Pl.: die Bänke*
bank [FINAN.]	die Bank *Pl.: die Banken*
bank	das Flussufer *Pl.: die Flussufer*
bank	das Ufer *Pl.: die Ufer*
plate [TECH.]	die Bank *Pl.: die Bänke*
bank	aufgeschütteter Damm
bank	das Bankgebäude *Pl.: die Bankgebäude*
bank	das Bankhaus *Pl.: die Bankhäuser*
bank	die Böschung *Pl.: die Böschungen - Fluss*
bank	der Damm *Pl.: die Dämme*
bank	der Deich *Pl.: die Deiche*
bank	der Erddamm *Pl.: die Erddämme*
bank	der Erdwall *Pl.: die Erdwälle*
bank	die Eskarpe
bank	der Fahrdamm *Pl.: die Fahrdämme*
bank	das Geldinstitut *Pl.: die Geldinstitute*
bank	das Kreditinstitut *Pl.: die Kreditinstitute*
bank	die Reihe *Pl.: die Reihen*
bank	das Stampfen *kein Pl.*
bank	der Stollen *Pl.: die Stollen*
bank	der Streb *Pl.: die Strebe*
bank	die Strosse *Pl.: die Strossen*
bank	der Vorwärmer *Pl.: die Vorwärmer*
bank	der Wall *Pl.: die Wälle*
settle	die Bank *Pl.: die Bänke*
bank [AVIAT.]	die Kurvenlage *Pl.: die Kurvenlagen*
bank [AVIAT.]	die Querneigung *Pl.: die Querneigungen*
bank [AVIAT.]	die Schräglage *Pl.: die Schräglagen*
bank [COMP.]	*abgegrenzter Teil des Speichers*
bank [COMP.]	die Speicherbank
bank [BAU.]	die Überhöhung *Pl.: die Überhöhungen [Straßenbau]*
bank [FINAN.]	das Bankinstitut *Pl.: die Bankinstitute [Bankwesen]*
bank [GEOL.]	die Abbauwand *Pl.: die Abbauwände*
bank [GEOL.]	die Kalksteinbank
bank [GEOL.]	die Klampe *Pl.: die Klampen*
bank [GEOL.]	natürlicher Damm
bank [GEOL.]	die Rasenhängebank

bseiten-Erlebnis zu ermöglichen. Außerdem werden teilweise auch Cookies von Diensten Dritter gesetzt. Weiterführende Inform

Terms and Concepts



Quelle: K. Aberer, IR

- Concepts are more abstract than terms
- Concepts are related to terms and to docs
- LSI models concepts as sets of strongly co-occurring terms
  - Can be computed using matrix manipulations
  - Concepts from LSI cannot be "spelled out", but are matrix columns

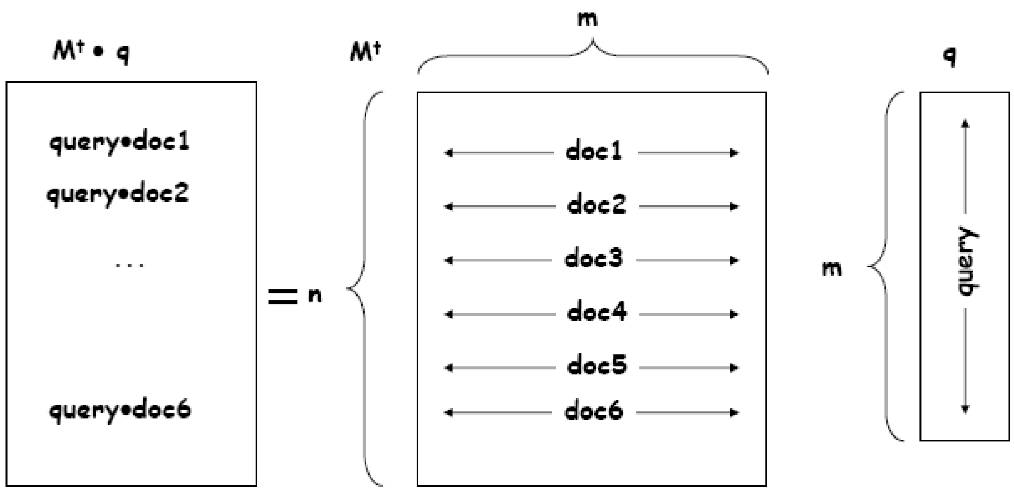
Term-Document Matrix

- 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

Begriff	Dokument 1	Dokument 2	Dokument 3
Access	1	0	0
Document	1	0	0
Retrieval	1	0	1
Information	0	1	1
Theory	0	1	0
Database	1	0	0
Indexing	1	0	0
Computer	0	1	1

Term-Document Matrix and VSM

- VSM uses the transposed document-term matrix ($=M^t$)
- Having M, we can in principle compute the vector v of the VSM-scores for q of all docs as $v=M^t \bullet q$
  - Only the dot product, normalization missing



Term and Document Correlation

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

	1	2	3	4	5
A	1	1	1		
B	1	1	1		1
C		1	1		
D				1	1

M (A..: terms; 1…: docs)

•

	A	B	C	D
1	1	1		
2	1	1	1	
3	1	1	1	
4				1
5		1		1

M^t

=

	A	B	C	D
A	3	3	2	0
B	3	4	2	1
C	2	2	2	0
D	0	1	0	2

Term correlation matrix

What to do with a Term-Document Matrix

- In the following, we approximate M by a particular M'
  - M' should be smaller than M
    - Less dimensions; faster computations; higher abstraction
  - M' should abstract from terms to concepts
    - The fewer dimensions capture the most frequent co-occurrences
- Approach: Find an M' such that $M'^t * q' \approx M^t * q$
  - Produce the least error among all M' of the same dimension

	D1	D2	D3	D4
and	1	1		
cat	1	1	1	
eat	1	1	1	
…				1
zoo		1		1

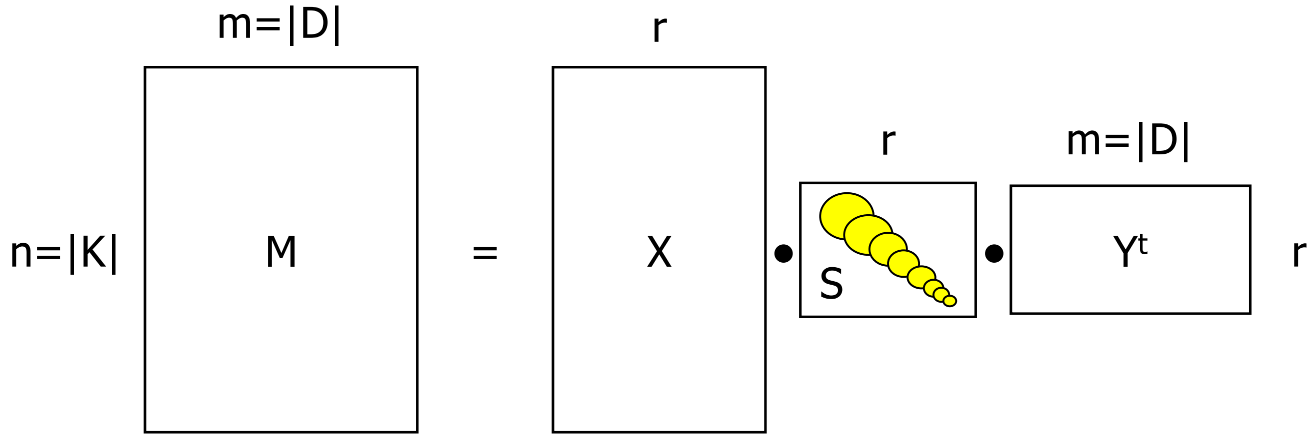
	D1	D2	D3	D4
C1	0,3	0,2	0	0,4
C2	0,7	0	0,1	0,9
C3	0,1	0	0,5	0,3

Some Linear Algebra

- The rank r of a matrix M is the maximal number of linearly independent rows of M
- If $Mx-\lambda x=0$ for a vector $x\neq0$, then λ is called an Eigenvalue of M and x is his associated Eigenvector
  - Eigenvectors/-werte are useful for many things
  - In particular, 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 iff M has "enough" Eigenvectors
    - 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
  - We use another factorization of M

Singular Value Decomposition (SVD)

- SVD decomposes any matrix M into $M = X \bullet S \bullet Y^t$
  - S is the diagonal matrix of the singular values of M in descending order and has size rxr (with r=rank(M))
  - X is the matrix of Eigenvectors of $M \bullet M^t$
  - Y is the matrix of Eigenvectors of $M^t \bullet M$
  - This decomposition is unique and can be computed in $O(r^3)$
    - Use approximation in practice



Example

- Assume for now M is quadratic and has full rank
  - Full rank: r=|K|=|D|

$$
\begin{array}{|c|c|c|}\hline M_{11} & M_{12} & M_{13} \\ \hline M_{21} & \dots & \dots \\ \hline M_{31} & \dots & M_{33} \\ \hline \end{array}
=
\begin{array}{|c|c|c|}\hline x_{11} & \dots & \dots \\ \hline x_{21} & \dots & \dots \\ \hline \dots & \dots & x_{33} \\ \hline \end{array}
\bullet
\begin{array}{|c|c|c|}\hline s_{11} & 0 & 0 \\ \hline 0 & s_{22} & 0 \\ \hline 0 & 0 & s_{33} \\ \hline \end{array}
\bullet
\begin{array}{|c|c|c|}\hline y_{11} & \dots & \dots \\ \hline y_{21} & \dots & \dots \\ \hline \dots & \dots & y_{33} \\ \hline \end{array}
$$

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

Approximating M

- LSI idea: What if we stop the sums earlier?
  - Recall: s_{ii} are sorted by descending value
  - Aggregating only over the first s_{ii}–values captures "most" of M

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

- $M_{11} = x_{11}*s_{11}*y_{11} + x_{12}*s_{22}*y_{21} + x_{13}*s_{33}*y_{31}$

$\boxed{\text{largest } s_{ij}}$ $\boxed{\text{2nd largest } s_{ij}}$ $\boxed{\text{3rd largest } s_{ij}}$

- What if $M_{11}' = x_{11}*s_{11}*y_{11} + x_{12}*s_{22}*y_{21}$

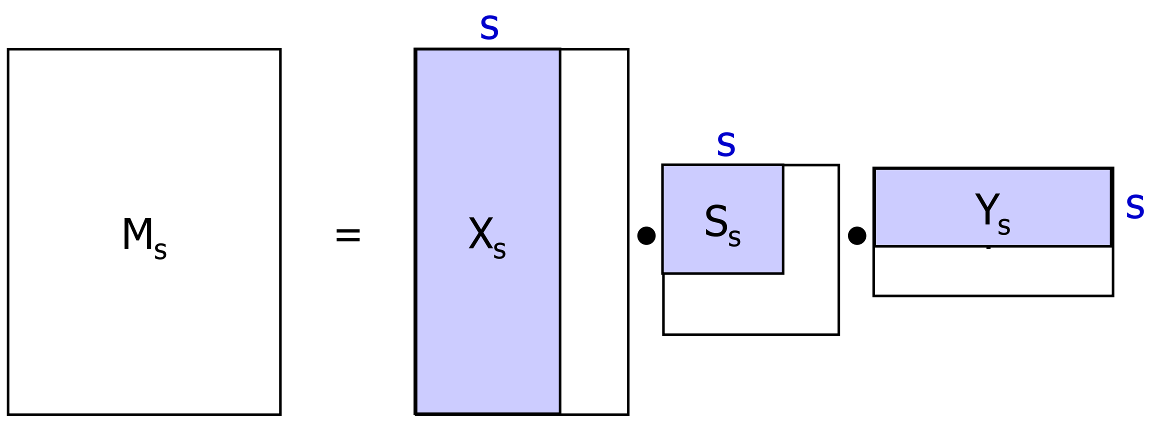
General Case

- In general, M is not quadratic and r < min(|K|,|D|)
  - All sums range from 1 to r

$$\begin{bmatrix} \sum X_{1i}S_{ii}Y_{i1} & \cdots & \sum X_{1i}S_{ii}Y_{im} \\ & & \\ & & \\ & & \\ & & \\ \sum X_{ni}S_{ii}Y_{im} & & \sum X_{ni}S_{ii}Y_{im} \end{bmatrix} = \left.\begin{bmatrix} & & \\ & & \\ & & \\ & & \\ & & \\ & & \end{bmatrix}\right\} n=|K| \bullet \overbrace{\begin{bmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & 0 \\ 0 & 0 & s_{33} \end{bmatrix}}^{r} \bullet \overbrace{\begin{bmatrix} & & & & \\ & & & & \\ & & & & \end{bmatrix}}^{m=|D|}$$

Approximating M

- LSI: Use S to approximate M
- Fix some $s<r$; Compute $M_s = X_s \bullet S_s \bullet Y_s^t$
  - 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
  - In fact, we don't need to compute M_s, but only need X_s, S_s and Y_s



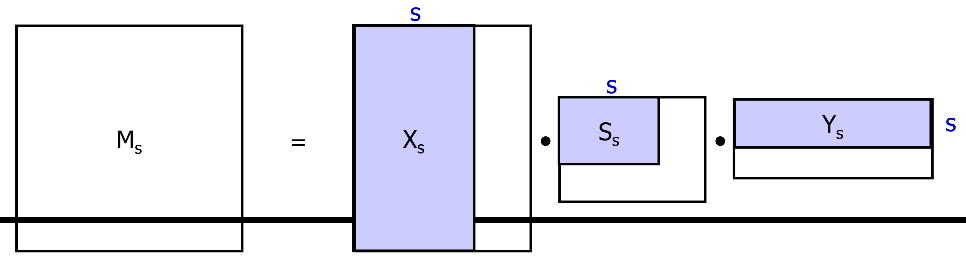
s-Approximations

- One can prove: M_s is the matrix with minimal $||M-M_s||_2$
  - M' is the optimal approximation of M when reducing r to s
- 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 (spurious co-occurrences) and to provide "semantic reduction"
  - s must be large enough to represent the diversity in the documents
  - Typical value: 200-500
    - While r is typically >100.000

LSI for Information Retrieval

- We map document vectors from a n-dimensional space into a s-dimensional space
- Approximated docs (still) are represented by columns in Y_s^t
- SVD as much as possible preserves distances between docs (depending on number of shared co-occurring terms)
- To this end, SVD (in a way) maps combinations of co-occurring terms onto the same new dimensions
- These terms-combinations can be understood as concepts
  - But they cannot easily be "named" – they are a bit of everything
- Universal idea: LSI has ample applications outside IR
  - Approximate a high-dimensional space through analysis of interdependencies between components

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?
  - Transform q into 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: $q' = q^t \bullet X_s \bullet S_s^{-1}$
  - This vector is compared to the transformed doc vectors as usual

Example: Term-Document Matrix

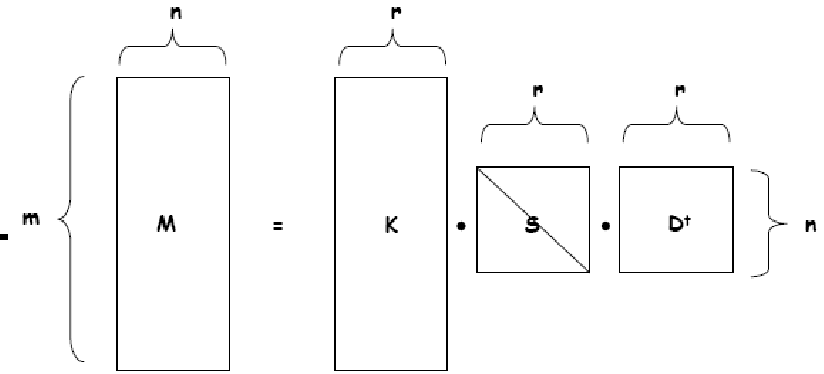
- Taken from Mi Islita: “Tutorials on SVD & LSI”
  - http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-1-understanding.html
    - Who took if from the Grossman and Frieder book

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“

$$
\begin{array}{l}
\text{Terms} \\
\text{a} \\ \text{arrived} \\ \text{damaged} \\ \text{delivery} \\ \text{fire} \\ \text{gold} \\ \text{in} \\ \text{of} \\ \text{shipment} \\ \text{silver} \\ \text{truck}
\end{array}
\quad
M = \begin{array}{ccc} d1 & d2 & d3 \\ \end{array}
\begin{bmatrix}
1 & 1 & 1 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 2 & 0 \\
0 & 1 & 1
\end{bmatrix}
\quad
\mathbf{q} = \begin{array}{c} q \\ \end{array}
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1
\end{bmatrix}
$$

Singular Value Decomposition



$$M = X \bullet S \bullet Y^t$$

$$X = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \quad S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

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

A Two-Approximation (s=2)

$$X_2 = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \qquad S_2 = \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix}$$

$$Y_2 = \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix} \qquad Y_2^t = \begin{array}{c} \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix} \\ \uparrow \qquad\quad \uparrow \qquad\quad \uparrow \\ d_1 \qquad\quad d_2 \qquad\quad d_3 \end{array}$$

Transforming the Query

$$q' = q^t \bullet X_2 \bullet S_2^{-1}$$

$$q` = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix}$$

$$= \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

Computing the Cosine of the Angle

$$\text{sim}(q, d) = \frac{q \bullet d}{|q|\,|d|}$$

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

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

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

Visualization of Results in 2D

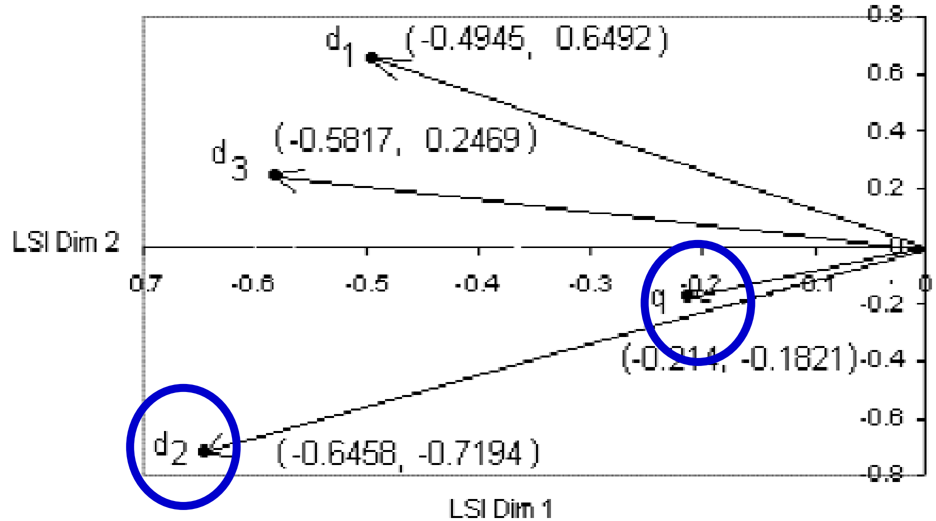
M =

Terms	d1	d2	d3
a	1	1	1
arrived	0	1	1
damaged	1	0	0
delivery	0	1	0
fire	1	0	0
gold	1	0	1
in	1	1	1
of	1	1	1
shipment	1	0	1
silver	0	2	0
truck	0	1	1

q =

Terms	q
a	0
arrived	0
damaged	0
delivery	0
fire	0
gold	1
in	0
of	0
shipment	0
silver	1
truck	1

Very large distance in original space



Pros and Cons

- Pro
  - Practical implementations exist, but not if corpus is very large
    - [MPS08] says: “no more than 1M docs”
  - 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 |K|, but more than |q|
    - Mapping the query turns a few keywords into an s-dimensional vector
    - We cannot simply index the “concepts” of M_s using inverted files etc.
    - Thus, LSI needs other techniques than inverted files
      - Means: lots of memory
    - Query speed not reduced compared to VSM (despite less dimensions)

Content of this Lecture

- IR Models
- Boolean Model
- Vector Space Model
- Relevance Feedback in the VSM
- Probabilistic Model
- Latent Semantic Indexing
- Outlook: Word Semantics and Word Embeddings

Word Semantics

- VSM considers two tokens as different when they have different spelling (“surface form”)
  - No gray: Equal or not, dimensions in VSM are orthogonal
  - King, princess, earl, milk, butter, cow, white, crown, emperor, …
- Makes models very dependent on a specific vocabulary and ignores richness of human languages – bad generalization
- Humans do compare words in a multi-facetted way
  - King is similar to princess to earl to queen, but not to cow
    - But all are mammals
  - Kings use crowns much more often than cows
- How can we capture word semantics to derive meaningful similarity scores instead of 1/0?

Knowledge-based: WordNet, Wikipedia, …

- Let‘s dream: A comprehensive resource of all words and their relationships
  - Specialization, synonymy, partonomy, relatedness, is_required_for, develops_into, is_possible_with, …
  - Example: WordNet
    - Roughly 150K concepts, 200K senses, 117K synsets
    - Specialization, partonomy, antonomy,
  - Can be turned into a semantic similarity measure, e.g., length of shortest path between two concepts
- Problem: Incomplete, costly, outdated, imprecise
  - Especially in specific domains like Biomedicine
- Much research to automatically expand WordNet, but no real breakthrough

Distributional Semantics

- Central idea: Represent a word by its context
- „You shall know a word by the company it keeps" [Firth, 1957]
  - The distribution of words co-occurring (context) with a given word X is characteristic for X
  - To learn about X, look at its context
  - If X and Y are semantically similar, also their contexts are similar
  - If X and Y are a bit different, also their contexts will be a bit different
- Finding: True in all domains and corpora of sufficient size
- For similarity: Compare contexts, not strings
- How can we do this efficiently and effectively?

Example

- Hunde bellen am Tag oft laut
- Katzen jagen nachts
- Luchse sind nachtaktiv und bewegen sich lautlos
- Wölfe jagen tagsüber
- Wölfe können bellen, aber meistens jaulen sie
- Wölfe jagen im Rudel
- Hunde bewachen oft Gruppen
- Katzen sind Einzelgänger
- Luchse jagen alleine

	Bellen	Tag	Laut	Schleichen	Nachts	Lautlos	Jagen	Jaulen	Rudel	Bewachen	Einzel
Hunde	1	1	1						1	1	
Katzen				1	1		1				1
Luchse					1	1	1				1
Wölfe	1	1					2	1	1		

Example

- Collocations “powerful”
  - enormously, especially, exceptionally, extraordinarily, extremely, immensely, incredibly, particularly, really, remarkably, surprisingly, tremendously, unusually, very | increasingly | fairly, pretty, quite, reasonably, relatively | enough, sufficientlyry
- Collocations “strong”
  - extremely, immensely, really, very | pretty, quite | enough
- Collocations “mighty”
  - River, warrior, man, blow, effort, force, hero, power, arm, hand, …

Source: https://www.freecollocation.com/

Naive Approach

- Given a large corpus D and a vocabulary K
- Define a context window (typically sentence)
- Represent every $k \in K$ as a $|K|$-dimensional vector v_k
  - Find set W of all context windows in D containing k
  - For every $k' \neq k$, count frequency of k' in W: $v_k[k'] = freq(k', W)$
  - May be normalized, e.g. tf*idf
- Similarity of words: Cosine similarity between their vectors
- Problem: Our model for each $d \in D$ grew from $|K|$ to $|K|^2$
  - Infeasible
  - We need an efficient and conservative dimensionality reduction
    - Efficient: Fast to compute; conservative: Distances are preserved
  - LSI too expensive

Word Embeddings (or Language Models)

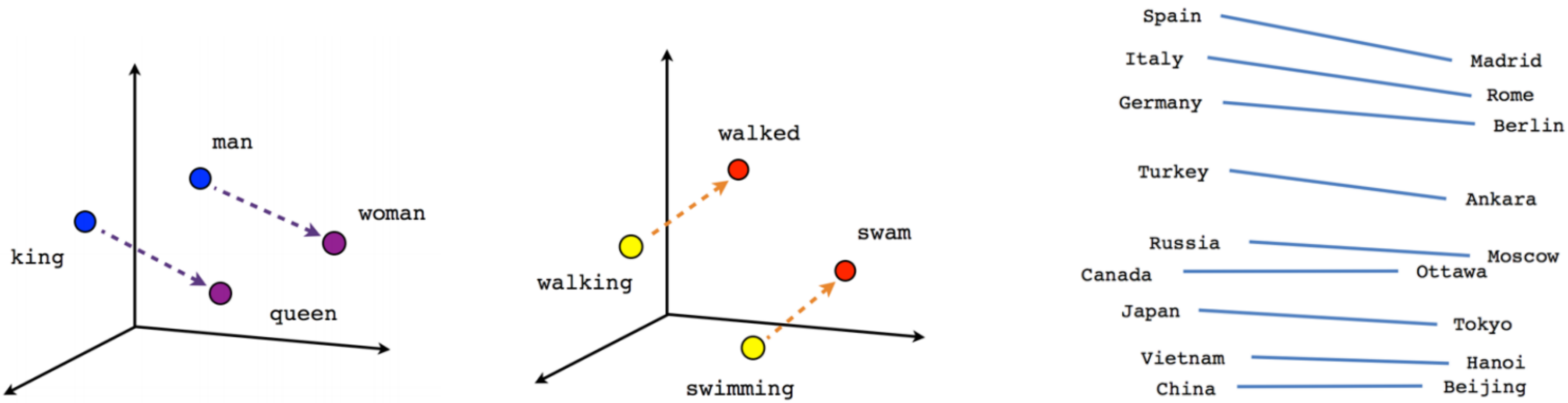
- How can we find “small” vectors for words such that semantic similarity is correlated to vector similarity?
- Word embeddings – embed word in a low-dim space
  - Low dimensional – typically 100-500 (a hyper parameter)
  - Very popular technique since app. 2015
- Today: Word embeddings are learned automatically
  - Can be precomputed and used without re-training in apps
  - Use statistical Machine Learning, not exact algebra
- Flourishing idea: Word2Vec, Glove, Elmo, Bert, Flair, …

Word2Vec [Mikolov et al. 2013]

- Idea of Word2Vec
  - Use a very large corpus
  - Define a context around words (sentence, window)
  - Cast the problem as classification
  - Continuous bag-of-words model (CBOW)
    - Turn every word w in every context into a classification problem
    - Learn a vector for each word such that the vectors of words in a context minus w can predict w
    - Note the "context" – we are close to distributional semantics
- Unsupervised learning – may use extremely large corpora
- Specific techniques to scale-up training (e.g. GPUs)

K2 is the second ? mountain in the world.

Does it Work?



Male-Female | Verb tense | Country-Capital

king – man ~ queen – woman

walking – walked ~ swimming – swam

Russia – Moscow ~ Vietnam – Hanoi

man - computer programmer ~ woman – homemaker
father - doctor ~ mother - nurse

Usage in Information Retrieval?

- Problem: We want to compare a query to a doc, not a word to a word
- Simple
  - Represent a doc by the average of all its word vectors
  - Same for query
  - Compute cosine of vectors
- More advanced
  - Compute sentence embeddings as average over words in sentence
  - Cluster sentence embeddings to find document segments
  - Match doc segments to query vector
- Fancy: Compute document embeddings
- Many more ideas

Self Assessment

- Explain the general approach of the probabilistic relevance model in IR
- How does one typically bootstrap this model?
- Which relevance model we discussed does consider the non-existent of terms in docs not existing in the query?
- Discuss the performance (speed) of the LSI approach to IR
- What is the difference between concept space and term space in LSI?
- Explain the Extended Boolean Model. Which of the shortcomings of the Boolean Model does it address?