Information Retrieval

WS 2016 / **2017**

Lecture 10, Tuesday January 10th, 2017
(Latent Semantic Indexing)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

Overview of this lecture

- Organizational
  - Your experiences with ES9 — Clustering
- Contents
  - Latent Semantic Indexing (LSI) — more linear algebra magic
  - Using LSI for retrieval — three variants
  - ES10: modify your VSM code from ES8 to use LSI, evaluate the benchmark again, and use LSI to find related terms

Experiences with ES9 1/5

- Summary / excerpts
  - Interesting + very nice linear algebra
  - Several of you skipped this sheet due to holidays
  - For some it was fast, others took more time than expected

    "We expected this sheet to be very easy and fast …
    however, debugging was very long and tedious"

    "This was the easiest sheet I ever failed to solve"

    "I'll start looking for the next Zen monastery to join"
  - Seeing the centroid words was the best part!
  - "Transpose here and there until the doctest ran through"
  - "Redundancy in TIP file and Exercise Sheet is not good"

Experiences with ES9 2/5

- Results
  - Many clusters on coherent "topics":

    thriller horror fiction crime action dead sci fi …

    kong hong wong chan lau cheung martial …
  - Some clusters of a different nature

    to, her, his, he, they, their, she, that, are, who …

    as, was, in, series, first, of, the, to, s, films, …
  - Some clusters are mixes of "topics" and general words

    short animated 2012 2011 2010 written 2007 drama …

    comedy fatty romance italian written harold lloyd …

Experiences with ES9 3/5

- Reincarnation, excerpts from your thoughts
  - Basic principle of reincarnation: Some soul/software is transferred from old to new body/hardware.
  - If a "soul" exists, it's not: solid, liquid, gas (macroscopic), nor electric, magnetic, light, radiation (microscopic)
  - Some say, they remember foreign countries … But with LSD people see purple elephants → don't trust your brain
  - Why is soul memoryless? (people who were reincarnated do not recall their previous life)
  - Reincarnation seems another excuse to not deal with the difficulties of the current life (e.g. "caste system")

Experiences with ES9 4/5

- Reincarnation, naïve view
  - There is some invisible "spirit" or "ghost" inside of us
  - Which lives on when the body dies
  - And then somehow finds its way into the next body
  - And somehow forgets everything about the previous life
  - Maybe that is so … maybe … but probably not

    However, some of you correctly pointed out that since we know nothing about the stuff that consciousness is made of, we cannot really know anything about this

Experiences with ES9 5/5

- Reincarnation, "interface" view
  - A common phenomenon in our universe: very complex internal state ↔ very compact external manifestation

    Example: personality ↔ a piece of communication

    Example: software executable ↔ the algorithm behind
  - Living beings are very good at (reverse) engineering the complex inner state from the compact representation
  - With death, your complex internal state dies, but some external manifestations easily live on (e.g. a book written)
  - A new "blank" organism with a suitable configuration can reverse-engineer the internal state (not exactly, of course)
  - In a very concrete sense, something has "reincarnated" then

Latent Semantic Indexing 1/10

- Motivation
  - Let's look at our example toy collection from L8 again:

    D_1 and D_2 and D_3 are "about" surfing the web

    D_5 and D_6 are "about" surfing on the beach

    internet and web are **synonyms**, surfing is a **polysem**
    = means different things in different context

	D_1	D_2	D_3	D_4	D_5	D_6
internet	1	1	0	1	0	0
web	1	0	1	1	0	0
surfing	1	1	1	2	1	1
beach	0	0	0	1	1	1

Latent Semantic Indexing 2/10

- Motivation
  - Let's look at the query web surfing again, using dot-product similarity as explained in L8
  - Then sim(D_3, Q) > sim(D_2, Q) = sim(D_5, Q)

    But D_2 seems just as relevant for the query as D_3, only that the word "internet" is used instead of "web"

	REL	REL	REL	REL	NOT	NOT	
	D_1	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**Q**
internet	1	1	0	1	0	0	0
web	1	0	1	1	0	0	1
surfing	1	1	1	2	1	1	1
beach	0	0	0	1	1	1	0
	1	(1)	2	3	1	1	

→ relevant but low score

Latent Semantic Indexing 3/10

- Conceptual solution

	REL	REL	REL	REL	NOT	NOT		
	D_1	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**		**Q**
internet	1	1	**1**	1	0	0		0
web	1	**1**	1	1	0	0		1
surfing	1	1	1	2	1	1		1
beach	0	0	0	1	1	1		0
	2	2	2	3	1	1		

Add the missing synonyms to the documents

Then indeed: $\text{sim}(D_1, Q) = \text{sim}(D_2, Q) = \text{sim}(D_3, Q)$

The goal of LSI is to do something like this **automagically**

Latent Semantic Indexing 4/10

- A simple but powerful observation

	D_1	D_2	D_3	D_4	D_5	D_6
	B_1	B_1	B_1	B_1+B_2	B_2	B_2
internet	1	1	**1**	1	0	0
web	1	**1**	1	1	0	0
surfing	1	1	1	2	1	1
beach	0	0	0	1	1	1

B_1	B_2
1	0
1	0
1	1
0	1

The modified matrix has **column rank 2**

That is, we can write each column as a (different) linear combination of the same two base columns (B_1 and B_2)

Note 1: the original matrix had column rank 4

Note 2: one can prove that **column rank = row rank**

Latent Semantic Indexing   5/10

- Matrix factorization

D_1	D_2	D_3	D_4	D_5	D_6
1	1	**1**	1	0	0
1	**1**	1	1	0	0
1	1	1	2	1	1
0	0	0	1	1	1

=

B_1	B_2
1	0
1	0
1	1
0	1

•

D'_1	D'_2	D'_3	D'_4	D'_5	D'_6
1	1	1	1	0	0
0	0	0	1	1	1

Equivalently: the 4 x 6 term-document matrix can be written as a product of a 4 x 2 matrix with a 2 x 6 matrix

The base vectors B_1 and B_2 are the underlying **"concepts"**

The vectors $D'_1, \ldots, D'_6$ are the representation of the documents in the (lower-dimensional) **"concept space"**

Latent Semantic Indexing 6/10

- The goal of LSI
  - Given an $m \times n$ term-document matrix A and $k < \text{rank}(A)$
  - Then find a matrix A' of (column) rank k such that the difference between A' and A is **as small as possible**

  Formally: $A' = \text{argmin}_{A' \; m \times n \text{ with rank } k} \| A - A' \|$

  For the $\| \dots \|$ we take the so-called **Frobenius-norm**

  This is defined as $\| D \| := \text{sqrt}(\Sigma_{ij} D_{ij}^2)$

  The reason for using this norm is purely technical: that way, the math on the next slides works out nicely

Latent Semantic Indexing 7/10

the rows of V are p.w. orthogonal and of unit length

the columns of U are pairwise orthogonal and of unit length

- How to find / compute such an A'
  - We first compute the so-called **singular value decomposition (SVD)** of the given matrix A :

    Theorem: for any m x n matrix A of rank r, there exist U, S, V such that **A = U · S · V** , and where

    U is an m x r matrix with $U^T \cdot U = I_r$ the r x r identity matrix

    S is an r x r matrix with non-zero entries only on its diag.

    V is an r x n matrix with $V \cdot V^T = I_r$

    The decomposition is unique up to simultaneous permutation of the rows/columns of U, S, and V

    Standard form: diagonal entries of S positive and sorted

Latent Semantic Indexing 8/10

- Using the SVD, our task becomes easy
  - Let $A = U \cdot S \cdot V$ be the SVD of A
  - For a given $k < \mathrm{rank}(A)$ let

    U_k = the first k columns of U   now an m x k matrix

    S_k = the upper k x k part of S   now a k x k matrix

    V_k = the first k rows of V   now a k x n matrix

    Note: then $U_k^T \cdot U_k = I_k$ and $V_k \cdot V_k^T = I_k$
  - Let $\mathbf{A_k = U_k \cdot S_k \cdot V_k}$

    Then A_k is a matrix of rank k that minimizes $\| A - A_k \|$

Latent Semantic Indexing 9/10

- How to compute the SVD
  - Can be computed from the **Eigenvector decomposition**

    See next slide for some of the mathematics behind

  - In practice, the more direct Lanczos method is used

    This has complexity O(k · nnz), where k is the rank and nnz is the number of non-zero values in the matrix

    Note that for term-document matrices nnz << n · m

    For ES10, just use **svds** from **scipy.sparse.linalg**

Latent Semantic Indexing 10/10

- Some of the mathematics behind the SVD
  - A real symmetric $n \times n$ matrix B has n pairwise orthogonal unit eigenvectors $u_1, \ldots, u_n$ (with eigenvalues $\lambda_1, \ldots, \lambda_n$)

    That is, $B \cdot u_i = \lambda_i \cdot u_i$ and $u_i \bullet u_j = 0$, for $i \neq j$, and $|u_i| = 1$

    Equivalently, $B = U \cdot \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \cdot U^T$ … and $U \cdot U^T = I$
  - The matrices $A \cdot A^T$ and $A^T \cdot A$ are symmetric, hence there exist orthogonal U and V and diagonals S_1 and S_2 such that

    $A \cdot A^T = U \cdot S_1 \cdot U^T$ and $A^T \cdot A = V \cdot S_2 \cdot V^T$
  - Let us assume that a decomposition $A = U \cdot S \cdot V$ exists:

Using LSI for better Retrieval 1/8

- **Variant 1:** work with A_k instead of A

	D_1	D_2	D_3	D_4	D_5	D_6
internet	1	1	0	1	0	0
web	1	0	1	1	0	0
surfing	1	1	1	2	1	1
beach	0	0	0	1	1	1

Our example A from the beginning

D'_1	D'_2	D'_3	D'_4	D'_5	D'_6
0.9	0.6	0.6	1.0	0.0	0.0
0.9	0.6	0.6	1.0	0.0	0.0
1.1	0.9	0.9	2.1	1.0	1.0
-0.1	0.1	0.1	0.9	1.0	1.0

best rank-2 approximation A_2

Using LSI for better Retrieval 2/8

- **Variant 1:** work with $\mathbf{A_k}$ instead of A
  - Problem: A_k is a dense matrix, that is, most / all of it's $m \cdot n$ entries will be non-zero

    Typically, both m and n will be very large, and then already storing this matrix is infeasible

    E.g. if $m = 1000$ and $n = 10M$ → $m \cdot n =$ **10 G**

Using LSI for better Retrieval 3/8

- **Variant 2:** work with $\mathbf{V_k}$ instead of with A
  - Recall: V_k gives the representation of the documents in the k-dimensional concept space

	D_1	D_2	D_3	D_4	D_5	D_6
internet	1	1	0	1	0	0
web	1	0	1	1	0	0
surfing	1	1	1	2	1	1
beach	0	0	0	1	1	1

Our example A from the beginning

D'_1	D'_2	D'_3	D'_4	D'_5	D'_6
0.4	0.3	0.3	0.7	0.3	0.3
0.5	0.2	0.2	0.0	-0.6	-0.6

V_2 from the SVD of A

Using LSI for better Retrieval 4/8

- **Variant 2:** work with $\mathbf{V_k}$ instead of with A
  - Observation: V_k is a dense matrix, that is, most or all of its $k \cdot n$ entries are non-zero

    Note: the original matrix A has $m' \cdot n$ non-zero entries, where m' is the average number of words in a document

    So storing V_k instead of A is ok if $k \approx m'$ or smaller

    Note: no need for a sparse representation (= an inverted index) when storing / using V_k

    This is the variant you should use for ES10.4

Using LSI for better Retrieval 5/8

- **Variant 2:** work with $\mathbf{V_k}$ instead of with A
  - Problem 2: we need to map the query to concept space

    The dot-product similarity of query q with all documents is

    $q^T \cdot A_k = q^T \cdot (U_k \cdot S_k \cdot V_k) = (q^T \cdot U_k \cdot S_k) \cdot V_k$

    Then $q_k^T := q^T \cdot U_k \cdot S_k$ is query mapped to concept space
  - The dot product $q_k^T \cdot V_k$ requires time $\sim n \cdot k$ … since both q_k and V_k are dense

    In comparison: computing the similarities of q with the original documents requires time $O(n \cdot \#q)$ and less

    where $\#q$ = number of query words in q

Using LSI for better Retrieval 6/8

- **Variant 3:** expand the original documents
  - In Variant 2, we have transformed both the query and the documents to concept space
  - LSI can also be viewed as doing **document expansion** in the original space + no change in the query

  Namely, let $T_k = U_k \cdot U_k^T$ this is an m x m matrix

  # terms

  $m \times m \quad m \times k \quad k \times m$

  Then one can easily prove that $A_k = T_k \cdot A$

  $A = U \cdot S \cdot V$

  $r = \text{rank}(A)$

  $T_k \cdot A = U_k \cdot \underbrace{U_k^T \cdot U}_{[I_k \; 0] \atop k \times r} \cdot S \cdot V = U_k \cdot S_k \cdot V_k = A_k$ ☒

  For ES10, simply compute T_k from U_k as shown, then compute the 100 term pairs with the largest entries in T_k

Using LSI for better Retrieval 7/8

- Variant 3: expand the original documents
  - Here is some intuition for T_k, assuming 0 or 1 entries

    In practice, we can get 0-1 entries by setting all entries in T above a certain threshold to 1, and all others to 0

What this effectively does: If „internet" is present in D_i → add „web" in D_i!

	internet	web	surfing	beach
internet	1	1	0	0
web	1	1	0	0
surfing	0	0	1	0
beach	0	0	0	1

•

D_i
1
0
1
0

=

D'_i
1
1
1
0

Using LSI for better Retrieval 8/8

- Linear combination with original scores
  - Experience: LSI adds some useful information to the term-document matrix, but also a lot of **noise**
  - In practice, one therefore uses a linear combination of the original scores and the LSI scores:

    Variant 1: $\text{scores} = \lambda \cdot q^T \cdot A + (1 - \lambda) \cdot q^T \cdot A_k$

    Variant 2: $\text{scores} = \lambda \cdot q^T \cdot A + (1 - \lambda) \cdot q_k^T \cdot V_k$

    Variant 3: $\text{scores} = \lambda \cdot q^T \cdot A + (1 - \lambda) \cdot q^T \cdot T_k \cdot A$

    For ES10, take Variant 2 and experiment with a good λ

References

- Further reading
  - Textbook Chapter 18: Matrix decompositions & LSI
    http://nlp.stanford.edu/IR-book/pdf/18lsi.pdf
  - Deerwester, Dumais, Landauer, Furnas, Harshman
    Indexing by Latent Semantic Analysis, JASIS 41(6), 1990
- Web resources
  - http://en.wikipedia.org/wiki/Latent_semantic_indexing
  - http://en.wikipedia.org/wiki/Singular_value_decomposition
  - http://www.numpy.org/
  - http://www.scipy.org/