

Variable Latent Semantic Indexing

Prabhakar Raghavan

Yahoo! Research
Sunnyvale, CA

November 2005

Joint work with A. Dasgupta, R. Kumar, A. Tomkins.

Yahoo! Research.

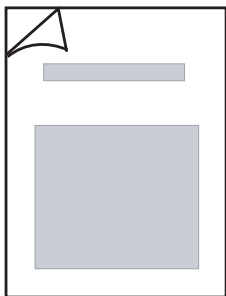
Outline

- 1 Introduction
- 2 Background
- 3 Variable Latent Semantic Indexing
- 4 Experiments

Outline

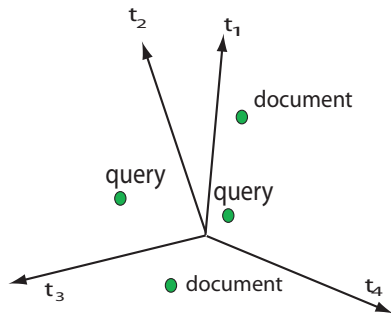
- 1 Introduction
- 2 Background
- 3 Variable Latent Semantic Indexing
- 4 Experiments

Searching Text Corpora



<u>Word</u>	<u>Count</u>
Apple	10
...	
Drivers	12
Oranges	0
...	
Tiger	20
Widget	5

Term-Document Matrices



- Each term is a dimension.
- Each document is a vector over terms.
- Query is a vector over terms.
- Weighting schemes
 - Boolean, Okapi, TF-IDF etc.
- Document "similarity"
 \approx closeness in term-space.

Document Similarity

query	1	1	1	0	0
-------	---	---	---	---	------	---

document ₁	1	0	1	0	
document ₂	0	0	1	1	0
document _n	1			0		1

A

- Term-document matrix A , query vector q .
- Document relevance to query given by (weighted) number of terms in common.
- Relevance scores given by $q^T A$.

Outline

- 1 Introduction
- 2 Background**
- 3 Variable Latent Semantic Indexing
- 4 Experiments

LSI at a high level

- Term-document matrix A .
- Want a representation \tilde{A} such that
 - \tilde{A} preserves semantic associations.
 - uses less resources.
- Goal : measuring query-document similarity using \tilde{A} is efficient and gives better results.

Basic intuition of SVD/LSI used in clustering, collaborative filtering.

Singular Value Decomposition

- Singular Value Decomposition of a 3×3 matrix

$$A = U \times \Sigma \times V^T$$
$$\begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}$$

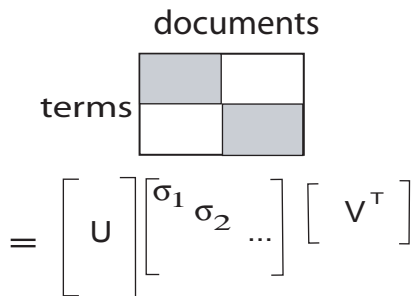
Latent Semantic Indexing



$$= \begin{bmatrix} U \end{bmatrix} \begin{bmatrix} \sigma_1 & \sigma_2 & \dots \end{bmatrix} \begin{bmatrix} V^T \end{bmatrix}$$

- Suppose only k topics in data.
- Keep top k singular values and vectors.
- Denoted as $A^{(k)}$.

Latent Semantic Indexing

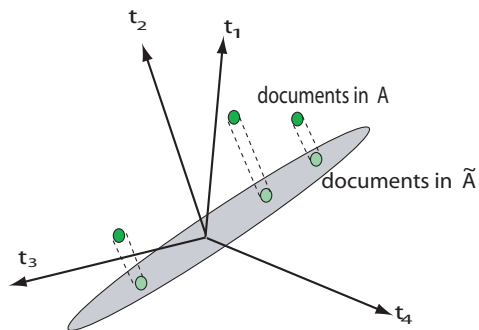


- Suppose only k topics in data.
- Keep top k singular values and vectors.
- Denoted as $A^{(k)}$.
- $A^{(k)}$ is “closest” rank- k matrix to A .
 - “closest” \equiv Frobenius, L_2 norms.

LSI in Answering Queries

- Propose $A^{(k)}$ as the representation \tilde{A} .
 - Space reduced by factor $\frac{k}{\text{avg}(\#terms)}$.
 - “Dimensionality” of corpus \equiv number of topics in corpus.
 - Results in “cleaner” representation of structure by ignoring “irrelevant” axes.
 - Believed to identify synonymous terms
 - e.g. car and automobile.
 - Disambiguate based on context.

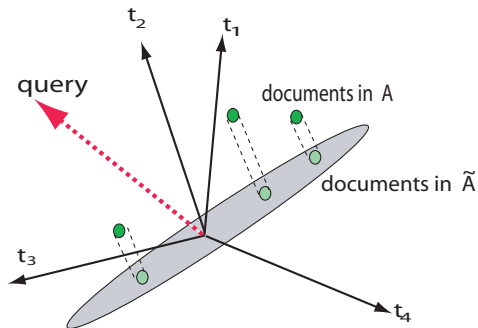
Latent Semantic Indexing



- Finds best rank- k subspace “fitting” the documents.

$$\tilde{A} = A^{(k)}.$$

Motivating Variable LSI



- Finds best rank- k subspace “fitting” the documents.

$$\tilde{A} = A^{(k)}.$$

- But, we were dealing with answering queries ?

Outline

- 1 Introduction
- 2 Background
- 3 Variable Latent Semantic Indexing**
- 4 Experiments

Query Distribution

- Query vectors have a skewed distribution over terms.
- In a corpus, might get queries only for a small subset of terms.
 - *e.g.*: Queries about sport and politics ...
- Co-occurrence between query terms ?
 - *e.g.* “data” + “mining”.
- Ad-hoc solution: delete irrelevant terms.
- Any principled approach ?

Variable Latent Semantic Indexing

- Probability distribution Q over set of terms.
- Query vector q chosen according to Q .
- Want \tilde{A} such that for most such vectors q ,

$$q^T A \approx q^T \tilde{A}.$$

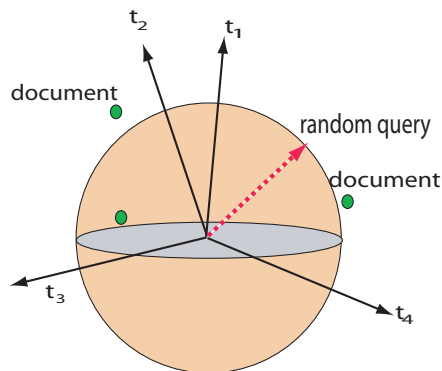
Variable Latent Semantic Indexing

- Probability distribution Q over set of terms.
- Query vector q chosen according to Q .
- Want \tilde{A} such that for most such vectors q ,

$$q^T A \approx q^T \tilde{A}.$$

- First cut : minimize expectation of $\|q^T(A - \tilde{A})\|$.

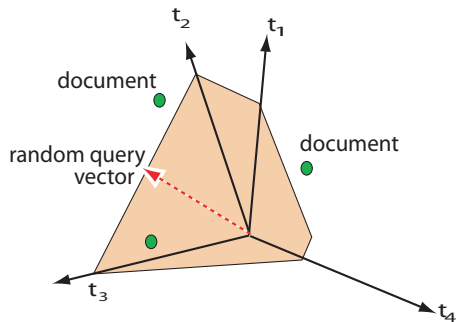
Isotropic Query Distribution



- Query distribution has uniformly random direction.
- Expected $\|q^T(A - \tilde{A})\|$ is minimized at

$$\tilde{A} = A^{(k)}.$$

Skewed Query Distribution



- Need to skew rank-k approximation to match query distribution.

Variable Latent Semantic Indexing

- Recall

A : term-document matrix, Q : query distribution.

- Co-occurrence matrix:

$$C = \mathbf{E} \mathbf{x}_{q \sim Q} [qq^T]$$

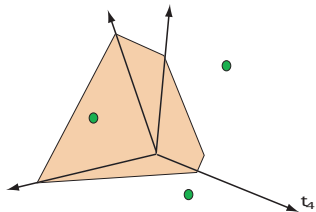
- $X = C^{1/2}A$.

- Find rank- k approximation $X^{(k)}$ of X .

- Return $\tilde{A} = C^{-1/2}X^{(k)}$.

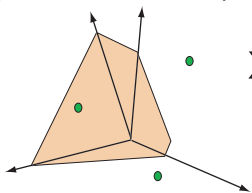
Proof Intuition

original term-document space



Proof Intuition

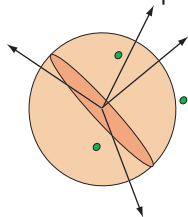
original term-document space



$$X = C^{1/2}A$$

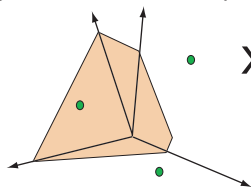


transformed term space



Proof Intuition

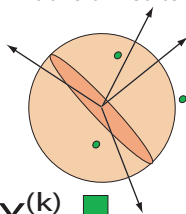
original term-document space



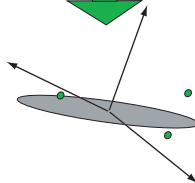
$$X = C^{1/2} A$$



transformed term space



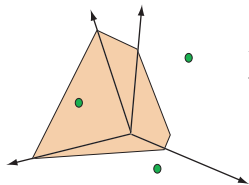
$$X^{(k)}$$



low-rank space in
transformed term space

Proof Intuition

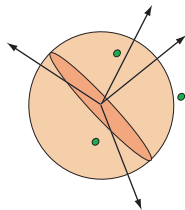
original term-document space



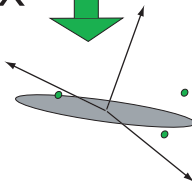
$$X = C^{1/2} A$$



transformed term space

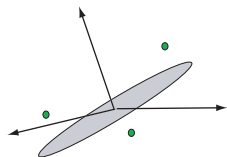


$$X^{(k)}$$



low-rank space in transformed term space

$$C^{-1/2} X^{(k)}$$



return to original space

Outline

- 1 Introduction
- 2 Background
- 3 Variable Latent Semantic Indexing
- 4 Experiments**

Experimental Setup

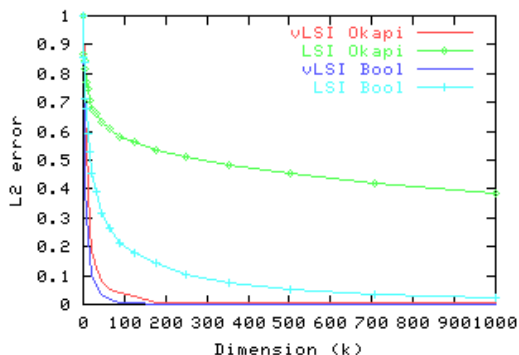
- Reuters data (1987).
 - 21k documents.
 - Five categories.
 - 112k terms.
 - 134 terms per document.
- preprocessing
 - porter-stemmed, case-folded and stop-worded.
 - term-document matrices with boolean, okapi weighting.
- used svdpackc.

Experimental Setup: Query Distribution

- Single-word
 - terms distributed according to frequency in corpus.
 - power law, ordered by distribution in corpus.
 - power law on random ordering.
- Two topics:
 - money, commodities
- Double-word: power law on ranked bigrams.

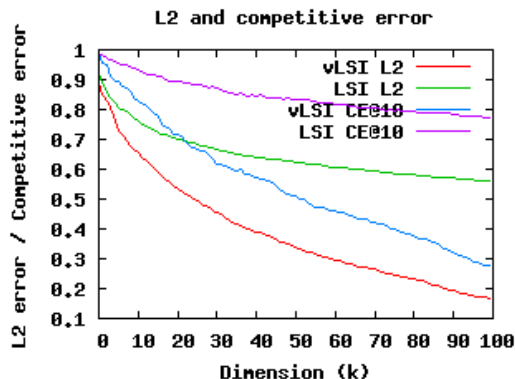
VLSI Results: L_2 error

L_2 error for query distribution D2, both matr



- Typically, saves an order of magnitude in dimensions for L_2 error.

Results: Competitive Error



- comp. error = 1 – comp. precision.
- Substantial improvements for competitive error too.

Summary

- LSI does effective dimension reduction, but can be fine-tuned using VLSI to query distributions.
- Space requirements same as that of LSI.
- Have to estimate co-occurrence matrix.

Future Work

- Personalized versions ?
- Analyze retrieval using stochastic data models ?
- Computational issues
 - Using sampling for efficiency ?
 - Updating using query streams ?
- Application to other domains?

Thanks !