Latent semantic indexing

- Relationship between concepts and words is many-to-many.
- Solve problems of synonymy and ambiguity by representing documents as vectors of **ideas** or **concepts**, not terms.
- For retrieval, analyze queries the same way, and compute cosine similarity of vectors of ideas.

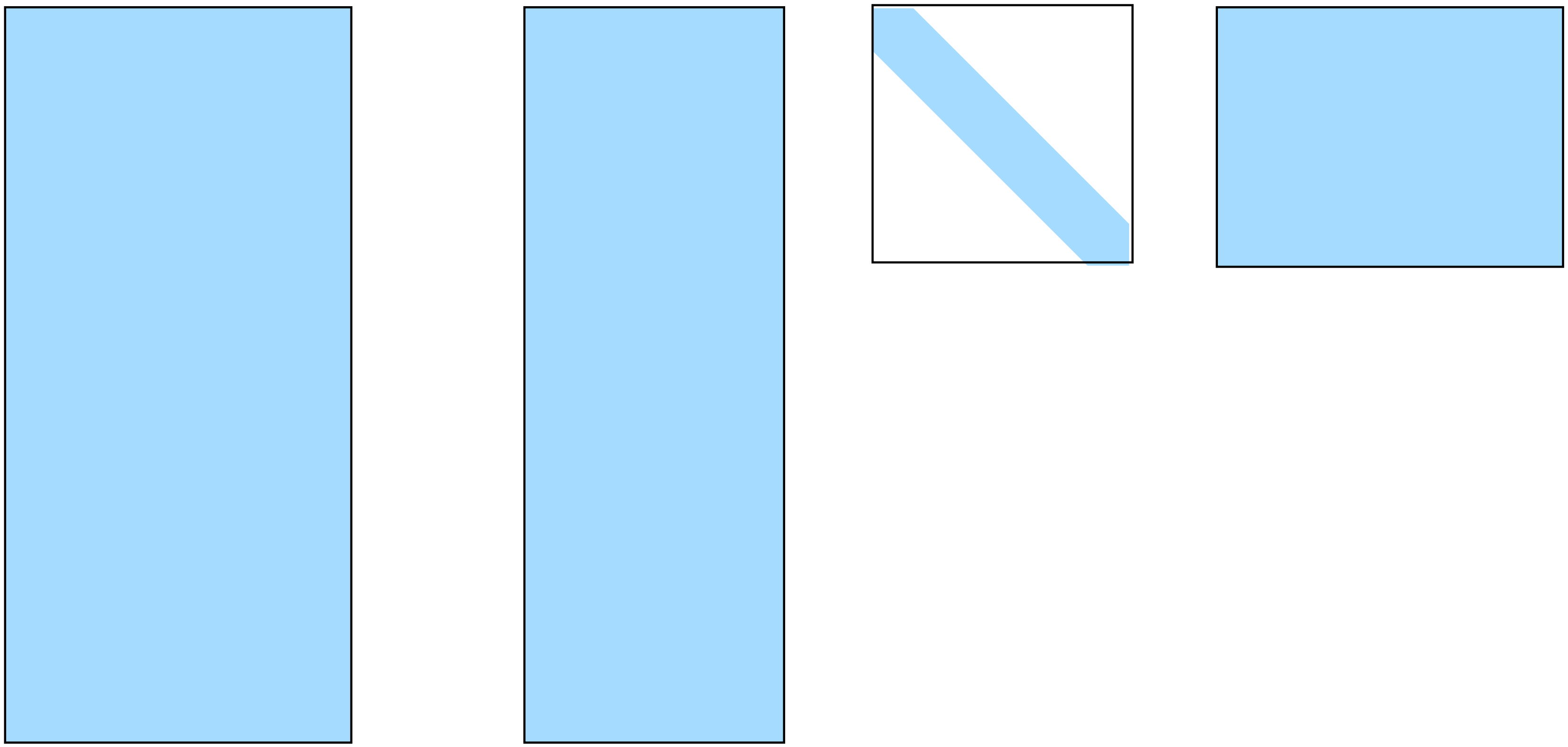
Latent semantic analysis

- **Latent semantic analysis (LSA).**
  - Find the **latent semantic space** that underlies the documents.
  - Find the basic (coarse-grained) ideas, regardless of the words used to say them.
  - A kind of **co-occurrence analysis**; co-occurring words as “bridges” between non–co-occurring words.
- Latent semantic space has many fewer dimensions than term space has.
  - Space depends on documents from which it is derived.
  - Components have no names; can’t be interpreted.

Singular value decomposition (1)

- **Dimensionality reduction** by **singular value decomposition (SVD)**.
- Analogous to least-squares fit: closest fit of a lower-dimensional matrix to a higher-dimensional matrix.
- **Theorem:** Let $A_{t\times d}$ be a real-valued matrix, and let $n = rank(A) \leq \min(t,d)$. There exist $T_{t\times n}$, diagonal $S_{n\times n}$, and $D_{d\times n}$ such that
  - $A = TSD^T$,
  - $s_{ii} \geq s_{jj}$ for all $1 \leq i < j \leq n$,
  - the columns of both T and D are orthonormal.
- Columns of T and D are the **singular vectors** of A; they represent terms and documents respectively); elements of S are the **singular values** of A.

Singular value decomposition (2)



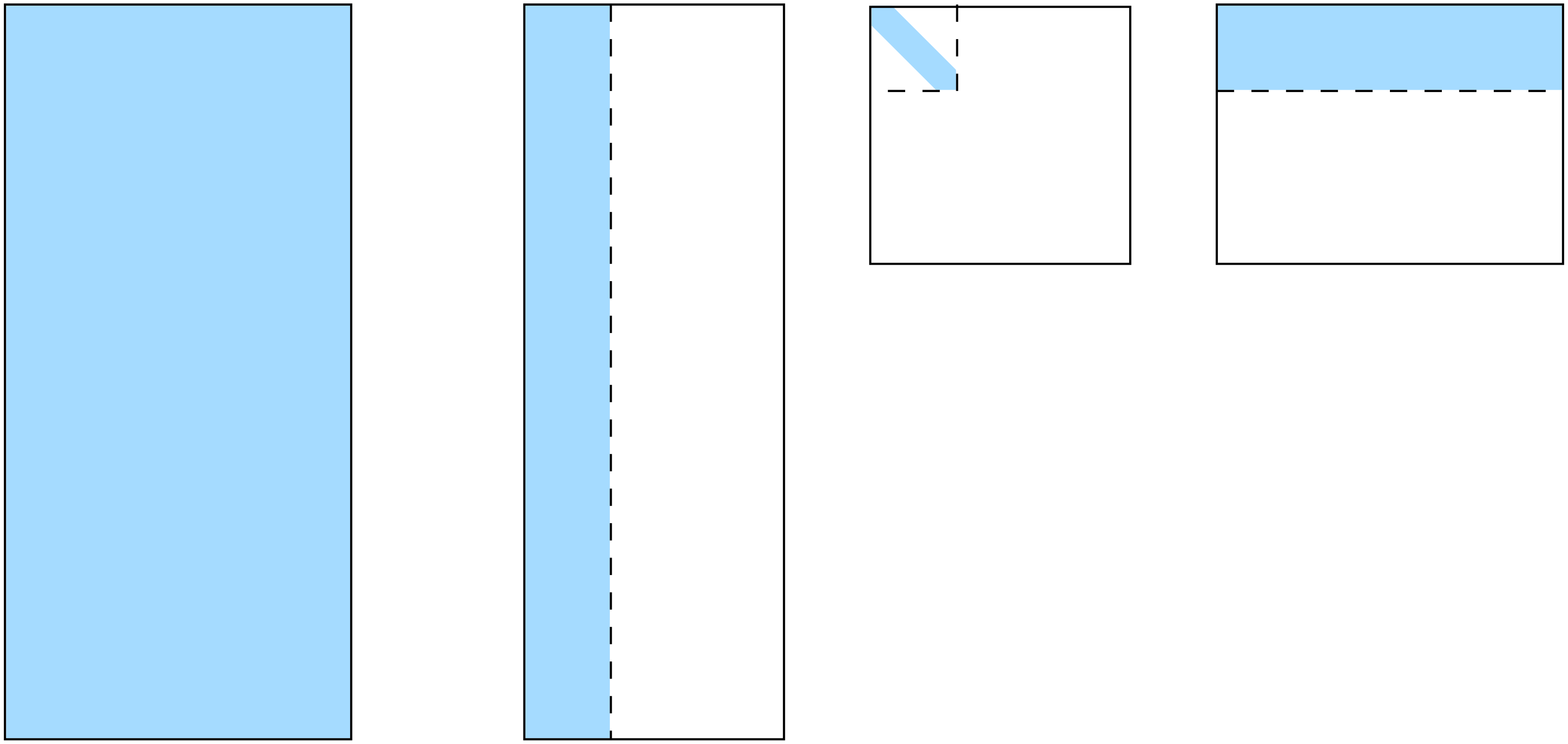
$$A_{t \times d} \quad = \quad T_{t \times n} \quad S_{n \times n} \quad {D_{d \times n}}^{T}$$

where $n = rank(A) \leq \min(t, d)$.

Singular value decomposition (3)

- For $k < n$, define $\hat{A}_{t\times d} = T_{t\times k} S_{k\times k} (D_{d\times k})^T$.
  - Although $\hat{A}$ and A are both $t \times d$ matrices, $\hat{A}$ is really "smaller": has rank k, can be represented as a smaller matrix.
- **Theorem:** $\hat{A}$ is the closest fit to A of a matrix of rank k; i.e., minimizes $\| A - \hat{A} \|_2$.

Singular value decomposition (4)



$$\hat{A}_{t\times d} \quad = \quad T_{t\times k} \quad S_{k\times k} \quad D_{d\times k}{}^{T}$$

Usually choose $k \ll n$.

Using singular vectors

- SVD algorithms.
- The k columns of T and D that remain in $T_{t\times k}$ and $D_{d\times k}$ are the “most important” ones.
- For document $\vec{d}$ in original normalized A, $A^T\vec{d}$ is vector of document similarities with $\vec{d}$; A^TA is (symmetrical) matrix of document-to-document similarities.
- Analogously in reduced space,

  $\hat{A}^T\hat{A} = (S_{k\times k}D_{d\times k}{}^T)^T(S_{k\times k}D_{d\times k}{}^T)$.
- Term similarity: AA^T approximated by

  $\hat{A}\hat{A}^T = (T_{t\times k}S_{k\times k})(T_{t\times k}S_{k\times k})^T$.

Example (1)

Six documents, five terms.

$$A = \left(\begin{array}{l|cccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\ \hline \text{cosmonaut} & 1 & 0 & 1 & 0 & 0 & 0 \\ \text{astronaut} & 0 & 1 & 0 & 0 & 0 & 0 \\ \text{moon} & 1 & 1 & 0 & 0 & 0 & 0 \\ \text{car} & 1 & 0 & 0 & 1 & 1 & 0 \\ \text{truck} & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right)$$

Example (2)

$$A = \underbrace{\left(\begin{array}{l|rrrrr} & \text{Dim 1} & \text{Dim 2} & \text{Dim 3} & \text{Dim 4} & \text{Dim 5} \\ \hline \text{cosmonaut} & -0.44 & -0.30 & 0.57 & 0.58 & 0.25 \\ \text{astronaut} & -0.13 & -0.33 & -0.59 & 0.00 & 0.73 \\ \text{moon} & -0.48 & -0.51 & -0.37 & 0.00 & -0.61 \\ \text{car} & -0.70 & 0.35 & 0.15 & -0.58 & 0.16 \\ \text{truck} & -0.26 & 0.65 & -0.41 & 0.58 & -0.09 \end{array}\right)}_{T_{5\times 5}} \underbrace{\left(\begin{array}{cccc} 2.16 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.59 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.28 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{array}\right.}_{S_{5\times 5}}$$

$$\underbrace{\left(\begin{array}{l|rrrrrr} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\ \hline \text{Dim 1} & -0.75 & -0.28 & -0.20 & -0.45 & -0.33 & -0.12 \\ \text{Dim 2} & -0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.41 \\ \text{Dim 3} & 0.28 & -0.75 & 0.45 & -0.20 & 0.12 & -0.33 \\ \text{Dim 4} & 0.00 & 0.00 & 0.58 & 0.00 & -0.58 & 0.58 \\ \text{Dim 5} & -0.53 & 0.29 & 0.63 & 0.19 & 0.41 & -0.22 \end{array}\right)}_{D_{6\times 5}{}^T}$$

Example (3)

Choose $k = 2$.

$$
\underbrace{\begin{pmatrix}
2.16 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.59 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.28 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.39
\end{pmatrix}}_{S_{2\times2}}
\underbrace{\left(\begin{array}{l|rrrrrr}
 & d_1 & d_2 & d_3 & d_4 & d_5 & d \\
\hline
\text{Dim 1} & -0.75 & -0.28 & -0.20 & -0.45 & -0.33 & -0.1 \\
\text{Dim 2} & -0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.4 \\
\hline
\text{Dim 3} & 0.28 & -0.75 & 0.45 & -0.20 & 0.12 & -0.3 \\
\text{Dim 4} & 0.00 & 0.00 & 0.58 & 0.00 & -0.58 & 0.5 \\
\text{Dim 5} & -0.53 & 0.29 & 0.63 & 0.19 & 0.41 & -0.2
\end{array}\right.}_{{D_{6\times2}}^T}
$$

$$
= \underbrace{\left(\begin{array}{l|rrrrrr}
 & d_1 & d_2 & d_3 & d_4 & d_5 & d \\
\hline
\text{Dim 1} & -1.62 & -0.60 & -0.04 & -0.97 & -0.71 & -0.2 \\
\text{Dim 2} & -0.46 & -0.84 & -0.30 & 1.00 & 0.35 & 0.6
\end{array}\right.}_{B_{2\times6}}
$$

Example (4)

Hence inter-document similarity is given by $\hat{A}^T\hat{A} = B^T B =$

$$\left(\begin{array}{c|rrrrrr}
 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
\hline
d_1 & 1.00 & & & & & \\
d_2 & 0.78 & 1.00 & & & & \\
d_3 & 0.40 & 0.88 & 1.00 & & & \\
d_4 & 0.47 & -0.18 & -0.62 & 1.00 & & \\
d_5 & 0.74 & 0.16 & -0.32 & 0.94 & 1.00 & \\
d_6 & 0.10 & -0.54 & -0.87 & 0.93 & 0.74 & 1.00
\end{array}\right)$$

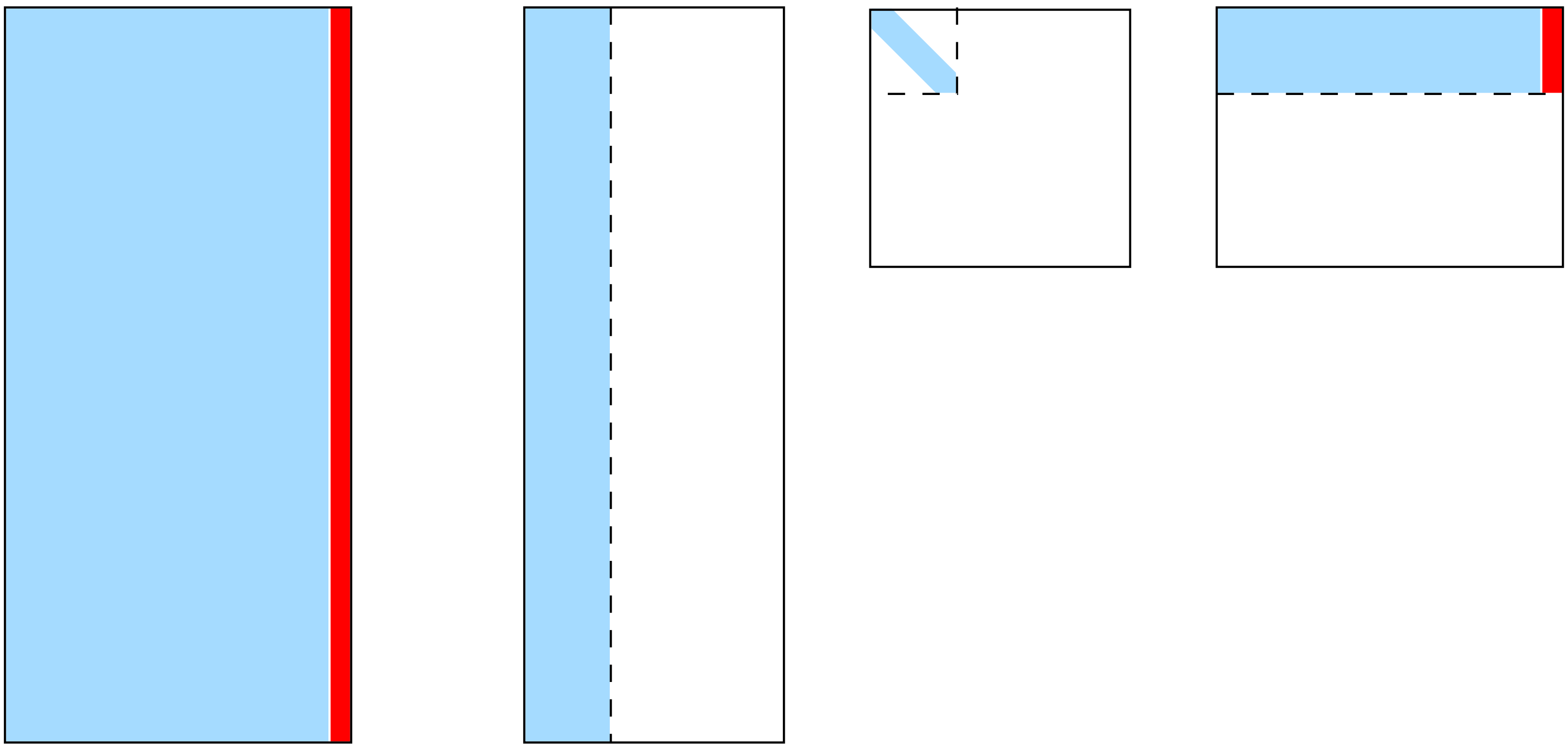
Queries and new documents

- Two problems:
  - Need to represent queries in same space.
  - Want to add new documents without recomputing SVD.
- **"Folding in":** Let $\vec{q}$ be the term vector for a query or new document. Then

  $$\hat{\vec{q}}_{k\times 1} = (T_{t\times k})^T \vec{q}_{t\times 1}$$

  is the vector representing $\vec{q}$ in the reduced space.
- If $\vec{q}$ is a query, $\hat{\vec{q}}$ can be compared to other documents in D by cosine similarity.
- If $\vec{q}$ is a new document, $\hat{\vec{q}}$ can be "appended" to D; d is increased by 1.
- As new documents are added, SVD will become much poorer fit. Eventually need to recompute SVD.

Adding a new document



$$\hat{A}_{t\times(d+1)} = T_{t\times k} \quad S_{k\times k} \quad D_{(d+1)\times k}^{T}$$

Choosing a value for *k*

- LSI is useful only if $k \ll n$.
- If k is too large, it doesn't capture the underlying latent semantic space; if k is too small, too much is lost.
- No principled way of determining the best k; need to experiment.

How well does this work?

- Effectiveness of LSI compared to regular term-matching depends on nature of documents.
  - Typical improvement: 0 to 30% better precision.
  - Advantage greater for texts in which synonymy and ambiguity are more prevalent.
  - Best when recall is high.
- Costs of LSI might outweigh improvement.
  - SVD is computationally expensive; limited use for really large document collections (as in TREC).
  - Inverted index not possible.

Other applications of LSI and LSA in NLP

- Cross-language information retrieval.
  - Concatenate multilingual abstracts to act as “bridge” between languages.
- People-retrieval by information retrieval.
- Text segmentation.
- Essay scoring.