# Indices

Tomasz Bartoszewski

# Inverted Index

- Search
- Construction
- Compression

# Inverted Index

- In its simplest form, the inverted index of a document collection is basically a data structure that attaches each distinctive term with a list of all documents that contains the term.

| | | | | |
|---|---|---|---|---|
| $id_1$: | Web | mining | is | useful. |
| | 1 | 2 | 3 | 4 |

| | | | |
|---|---|---|---|
| $id_2$: | Usage | mining | applications. |
| | 1 | 2 | 3 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $id_3$: | Web | structure | mining | studies | the | Web | hyperlink | structure. |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

| | |
|---|---|
| Applications: | $id_2$ |
| Hyperlink: | $id_3$ |
| Mining: | $id_1$, $id_2$, $id_3$ |
| Structure: | $id_3$ |
| Studies: | $id_3$ |
| Usage: | $id_2$ |
| Useful: | $id_1$ |
| Web: | $id_1$, $id_3$ |

(A)

| | |
|---|---|
| Applications: | <$id_2$, 1, [3]> |
| Hyperlink: | <$id_3$, 1, [7]> |
| Mining: | <$id_1$, 1, [2]>, <$id_2$, 1, [2]>, <$id_3$, 1, [3]> |
| Structure: | <$id_3$, 2, [2, 8]> |
| Studies: | <$id_3$, 1, [4]> |
| Usage: | <$id_2$, 1, [1]> |
| Useful: | <$id_1$, 1, [4]> |
| Web: | <$id_1$, 1, [1]>, <$id_3$, 2, [1, 6]> |

(B)

# Search Using an Inverted Index

# Step 1 – vocabulary search

finds each query term in the vocabulary

```
If (Single term in query){
        goto step3;
}
Else{
        goto step2;
}
```

# Step 2 – results merging

- merging of the lists is performed to find their intersection
- use the shortest list as the base
- partial match is possible

# Step 3 – rank score computation

- based on a relevance function (e.g. okapi, cosine)
- score used in the final ranking

# Example

| | |
|---|---|
| Applications: | $id_2$ |
| Hyperlink: | $id_3$ |
| Mining: | $id_1$, $id_2$, $id_3$ |
| Structure: | $id_3$ |
| Studies: | $id_3$ |
| Usage: | $id_2$ |
| Useful: | $id_1$ |
| Web: | $id_1$, $id_3$ |

(A)

| | |
|---|---|
| Applications: | <$id_2$, 1, [3]> |
| Hyperlink: | <$id_3$, 1, [7]> |
| Mining: | <$id_1$, 1, [2]>, <$id_2$, 1, [2]>, <$id_3$, 1, [3]> |
| Structure: | <$id_3$, 2, [2, 8]> |
| Studies: | <$id_3$, 1, [4]> |
| Usage: | <$id_2$, 1, [1]> |
| Useful: | <$id_1$, 1, [4]> |
| Web: | <$id_1$, 1, [1]>, <$id_3$, 2, [1, 6]> |

(B)


| | |
|---|---|
| Mining: | <$id_1$, 1, [2]>, <$id_2$, 1, [2]>, <$id_3$, 1, [3]> |
| Web: | <$id_1$, 1, [1]>, <$id_3$, 2, [1, 6]> |

# Index Construction

# Time complexity

- O(T), where T is the number of all terms (including duplicates) in the document collection (after pre-processing)

# Index Compression

# Why?

- avoid disk I/O
- the size of an inverted index can be reduced dramatically
- the original index can also be reconstructed
- all the information is represented with positive integers -> integer compression

# Use gaps

- 4, 10, 300, and 305 -> 4, 6, 290 and 5
- Smaller numbers
- Large for rare terms – not a big problem

# All in one

| Decimal | Unary | Elias Gamma | Elias Delta | Golomb ($b = 3$) | Golomb ($b = 10$) | Variable byte |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 10 | 1 001 | 0000001 0 |
| 2 | 01 | 0 10 | 0 100 | 1 11 | 1 010 | 0000010 0 |
| 3 | 001 | 0 11 | 0 101 | 01 0 | 1 011 | 0000011 0 |
| 4 | 0001 | 00 100 | 0 1100 | 01 10 | 1 100 | 0000100 0 |
| 5 | 00001 | 00 101 | 0 1101 | 01 11 | 1 101 | 0000101 0 |
| 6 | 000001 | 00 110 | 0 1110 | 001 0 | 1 1100 | 0000110 0 |
| 7 | 0000001 | 00 111 | 0 1111 | 001 10 | 1 1101 | 0000111 0 |
| 8 | 00000001 | 000 1000 | 00 100000 | 001 11 | 1 1110 | 0001000 0 |
| 9 | 000000001 | 000 1001 | 00 100001 | 0001 0 | 1 1111 | 0001001 0 |
| 10 | 0000000001 | 000 1010 | 00 100010 | 0001 10 | 01 000 | 0001010 0 |

# Unary

- For x:

X-1 bits of 0 and one of 1

e.g.

5 -> 00001

7 -> 0000001

# Elias Gamma Coding

- $1 + \lfloor \log_2 x \rfloor$ in unary (i.e., $\lfloor \log_2 x \rfloor$ 0-bits followed by a 1-bit)
- followed by the binary representation of x without its most significant bit.
- efficient for small integers but is not suited to large integers
- $1 + \lfloor \log_2 x \rfloor$ is simply the number of bits of x in binary
- 9 -> 000 1001

# Elias Delta Coding

- For small int longer than gamma codes (better for larger)
- gamma code representation of $1 + \lfloor \log_2 x \rfloor$
- followed by the binary representation of x less the most significant bit
- Dla 9:

$1 + \lfloor \log_2 9 \rfloor = 4$ -> 00100

9 -> 00100 001

# Golomb Coding

- values relative to a constant b
- several variations of the original Golomb
- E.g.

$q = \lfloor x/b \rfloor$

Remainder $\mathrm{r} = x - qb$ (b possible reminders e.g. b=3: 0,1,2)

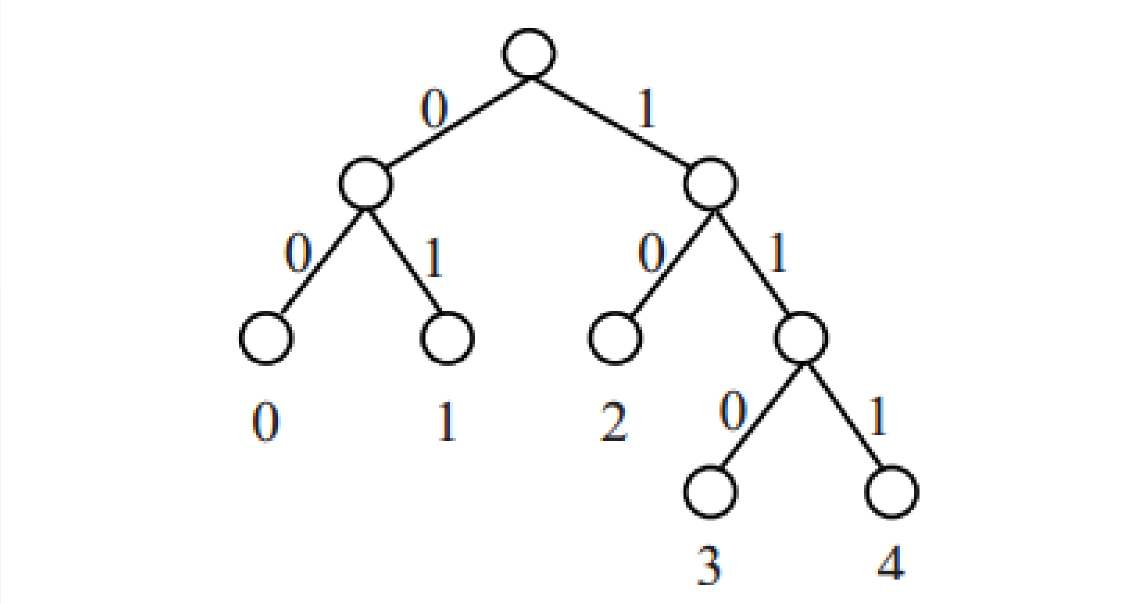
binary representation of a remainder requires $\lfloor \log_2 b \rfloor$ or $\lceil \log_2 b \rceil$

write the first few remainders using $\lfloor \log_2 b \rfloor$ rest $\lceil \log_2 b \rceil$

# Example

- b=3 and x=9
- $q = \lfloor 9/3 \rfloor = 3$
- $i = \lfloor \log_2 3 \rfloor = 1 => d = 1 \ \ (d = 2^{i+1} - b)$
- $r = 9 - 3 * 3 = 0$
- Result 00010

# The coding tree for b=5



# Selection of b

- $b \approx 0.69 * \frac{N}{n_t}$
- N – total number of documents
- $n_t$– number of documents that contain term t

# Variable-Byte Coding

- seven bits in each byte are used to code an integer
- last bit 0 – end, 1 – continue
- E.g. 135 -> 00000011 00001110

# Summary

- Golomb coding better than Elias
- Gamma coding does not work well
- Variable-byte integers are often faster than Variable-bit (higher storage costs)
- compression technique can allow retrieval to be up to twice as fast than without compression
- space requirement averages 20% – 25% of the cost of storing uncompressed integers

# Latent Semantic Indexing

# Reason

- many concepts or objects can be described in multiple ways
- find using synonyms of the words in the user query
- deal with this problem through the identification of statistical associations of terms

# Singular value decomposition (SVD)

- estimate latent structure, and to remove the “noise”
- hidden “concept” space, which associates syntactically different but semantically similar terms and documents

# LSI

- LSI starts with an m*n termdocument matrix A
- row = term; column = document
- value e.g. term frequency

# Singular Value Decomposition

- factor matrix A into three matrices:

$$A = UEV^T$$

m is the number of row in A

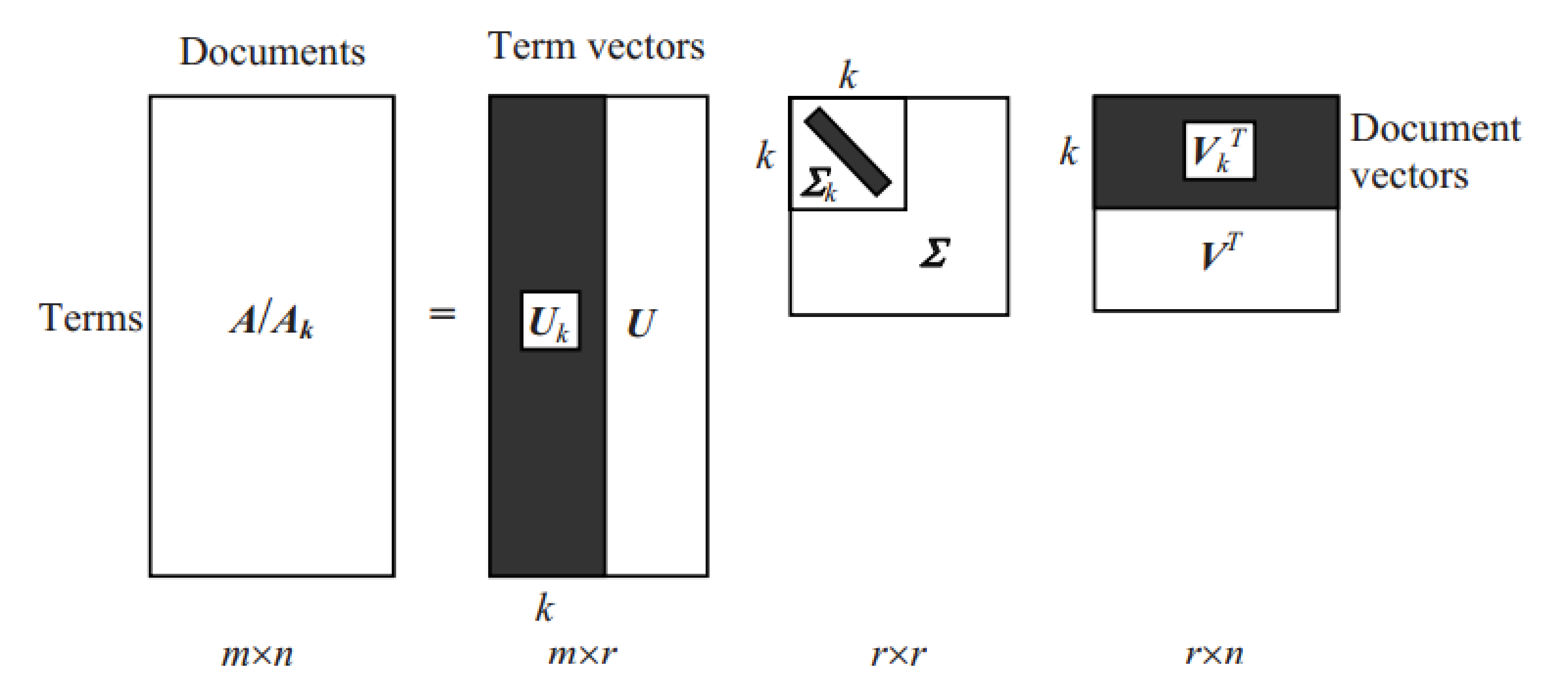
n is the number of columns in A

r is the rank of A, $\mathrm{r} \leq \min(m, n)$

# Singular Value Decomposition

- U is a $m * r$ matrix and its columns, called left singular vectors, are eigenvectors associated with the r non-zero eigenvalues of $AA^T$
- V is an $\mathrm{n} * r$ matrix and its columns, called right singular vectors, are eigenvectors associated with the r non-zero eigenvalues of $A^T A$
- E is a $\mathrm{r} * r$ diagonal matrix, E = diag($\sigma_1, \sigma_2, \dots, \sigma_r$), $\sigma_1 > 0$. $\sigma_1, \sigma_2, \dots,$ $\sigma_r$, called singular values, are the non-negative square roots of r non-zero eigenvalues of $AA^T$ they are arranged in decreasing order, i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$
- reduce the size of the matrices

$$A_k = U_k E_k V_k^T$$


Documents
Term vectors
Terms
A/A_k
=
U_k
U
k
k
k
Σ_k
Σ
k
V_k^T
V^T
Document vectors
m×n
m×r
r×r
r×n

# Query and Retrieval

- q - user query (treated as a new document)
- document in the k-concept space, denoted by $q_k$
- $q_k = q^T U_k E_k^{-1}$

# Example

$c_1$: Human machine interface for Lab ABC computer applications
$c_2$: A survey of user opinion of computer system response time
$c_3$: The EPS user interface management system
$c_4$: System and human system engineering testing of EPS
$c_5$: Relation of user-perceived response time to error measurement
$m_1$: The generation of random, binary, unordered trees
$m_2$: The intersection graph of paths in trees
$m_3$: Graph minors IV: Widths of trees and well-quasi-ordering
$m_4$: Graph minors: A survey

# Example

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

| | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $m_1$ | $m_2$ | $m_3$ | $m_4$ |
|---|---|---|---|---|---|---|---|---|---|
| *human* | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| *interface* | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| *computer* | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *user* | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| *system* | 0 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 |
| *response* | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| *time* | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| *EPS* | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| *survey* | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| *trees* | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| *graph* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| *minors* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# Example

$$\boldsymbol{U} = \begin{pmatrix}
0.22 & -0.11 & 0.29 & -0.41 & -0.11 & -0.34 & 0.52 & -0.06 & -0.41 \\
0.20 & -0.07 & 0.14 & -0.55 & 0.28 & 0.50 & -0.07 & -0.01 & -0.11 \\
0.24 & 0.04 & -0.16 & -0.59 & -0.11 & -0.25 & -0.30 & 0.06 & 0.49 \\
0.40 & 0.06 & -0.34 & 0.10 & 0.33 & 0.38 & 0.00 & 0.00 & 0.01 \\
0.64 & -0.17 & 0.36 & 0.33 & -0.16 & -0.21 & -0.17 & 0.03 & 0.27 \\
0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\
0.27 & 0.11 & -0.43 & 0.07 & 0.08 & -0.17 & 0.28 & -0.02 & -0.05 \\
0.30 & -0.14 & 0.33 & 0.19 & 0.11 & 0.27 & 0.03 & -0.02 & -0.17 \\
0.21 & 0.27 & -0.18 & -0.03 & -0.54 & 0.08 & -0.47 & -0.04 & -0.58 \\
0.01 & 0.49 & 0.23 & 0.03 & 0.59 & -0.39 & -0.29 & 0.25 & -0.23 \\
0.04 & 0.62 & 0.22 & 0.00 & -0.07 & 0.11 & 0.16 & -0.68 & 0.23 \\
0.03 & 0.45 & 0.14 & -0.01 & -0.30 & 0.28 & 0.34 & 0.68 & 0.18
\end{pmatrix}$$

# Example

$$\boldsymbol{\Sigma}=\begin{pmatrix} 3.34 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.54 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.35 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.64 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.31 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.56 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.36 \end{pmatrix}$$

# Example

$$V = \begin{pmatrix} 0.20 & -0.06 & 0.11 & -0.95 & 0.05 & -0.08 & 0.18 & -0.01 & -0.06 \\ 0.61 & 0.17 & -0.50 & -0.03 & -0.21 & -0.26 & -0.43 & 0.05 & 0.24 \\ 0.46 & -0.13 & 0.21 & 0.04 & 0.38 & 0.72 & -0.24 & 0.01 & 0.02 \\ 0.54 & -0.23 & 0.57 & 0.27 & -0.21 & -0.37 & 0.26 & -0.02 & -0.08 \\ 0.28 & 0.11 & -0.51 & 0.15 & 0.33 & 0.03 & 0.67 & -0.06 & -0.26 \\ 0.00 & 0.19 & 0.10 & 0.02 & 0.39 & -0.30 & -0.34 & 0.45 & -0.62 \\ 0.01 & 0.44 & 0.19 & 0.02 & 0.35 & -0.21 & -0.15 & -0.76 & 0.02 \\ 0.02 & 0.62 & 0.25 & 0.01 & 0.15 & 0.00 & 0.25 & 0.45 & 0.52 \\ 0.08 & 0.53 & 0.08 & -0.03 & -0.60 & 0.36 & 0.04 & -0.07 & -0.45 \end{pmatrix}$$

# Example

$$\boldsymbol{A}_k = \underset{\boldsymbol{U}_k}{\begin{pmatrix} 0.22 & -0.11 \\ 0.20 & -0.07 \\ 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & -0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{pmatrix}} \underset{\boldsymbol{\Sigma}_k}{\begin{bmatrix} 3.34 & 0 \\ 0 & 2.54 \end{bmatrix}} \underset{\boldsymbol{V}_k^{T}}{\begin{bmatrix} 0.20 & 0.61 & 0.46 & 0.54 & 0.28 & 0.00 & 0.02 & 0.02 & 0.08 \\ -0.06 & 0.17 & -0.13 & -0.23 & 0.11 & 0.19 & 0.44 & 0.62 & 0.53 \end{bmatrix}}$$

# Example

q - "user interface"

$$\mathbf{q}_k = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T \begin{pmatrix} 0.22 & -0.11 \\ 0.20 & -0.07 \\ 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & -0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{pmatrix} \begin{bmatrix} 3.34 & 0 \\ 0 & 2.54 \end{bmatrix}^{-1} = (0.179 \quad -0.004)$$

# Example

| | |
|---|---|
| $c_1$: | 0.964 |
| $c_2$: | 0.957 |
| $c_3$: | 0.968 |
| $c_4$: | 0.928 |
| $c_5$: | 0.922 |
| $m_1$: | –0.022 |
| $m_2$: | 0.023 |
| $m_3$: | 0.010 |
| $m_4$: | 0.127 |

| | |
|---|---|
| $c_1$: | Human machine interface for Lab ABC computer applications |
| $c_2$: | A survey of user opinion of computer system response time |
| $c_3$: | The EPS user interface management system |
| $c_4$: | System and human system engineering testing of EPS |
| $c_5$: | Relation of user-perceived response time to error measurement |
| $m_1$: | The generation of random, binary, unordered trees |
| $m_2$: | The intersection graph of paths in trees |
| $m_3$: | Graph minors IV: Widths of trees and well-quasi-ordering |
| $m_4$: | Graph minors: A survey |

# Summary

- The original paper of LSI suggests 50–350 dimensions.
- k needs to be determined based on the specific document collection
- association rules may be able to approximate the results of LSI