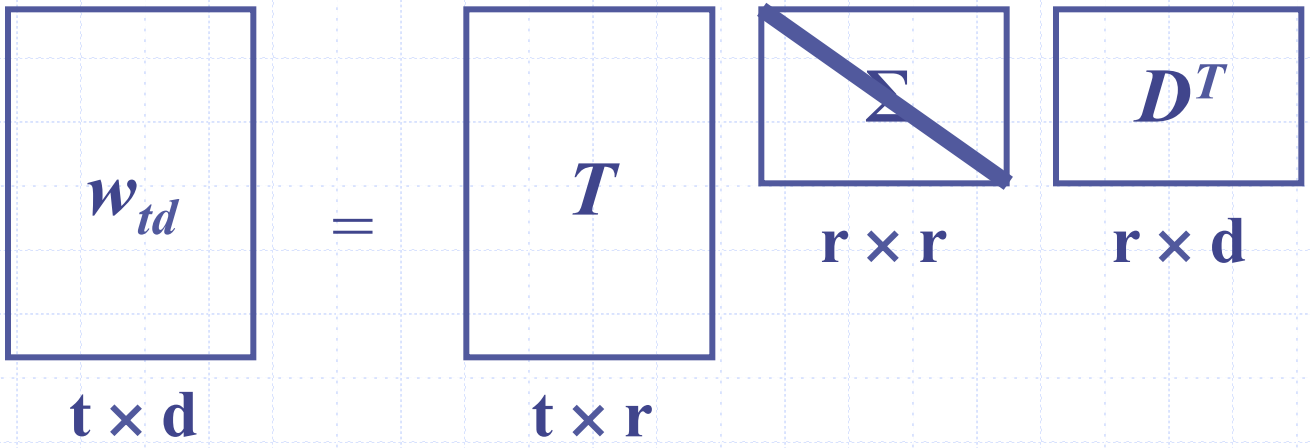
Latent Semantic Indexing

Thanks to Ian Soboroff

Issues: Vector Space Model

- Assumes terms are independent
  - Some terms are likely to appear together
    - synonyms, related words
    - spelling mistakes?
  - Terms can have different meanings depending on context
- Term-document matrix has very high dimensionality
  - are there really that many important features for each document and term?

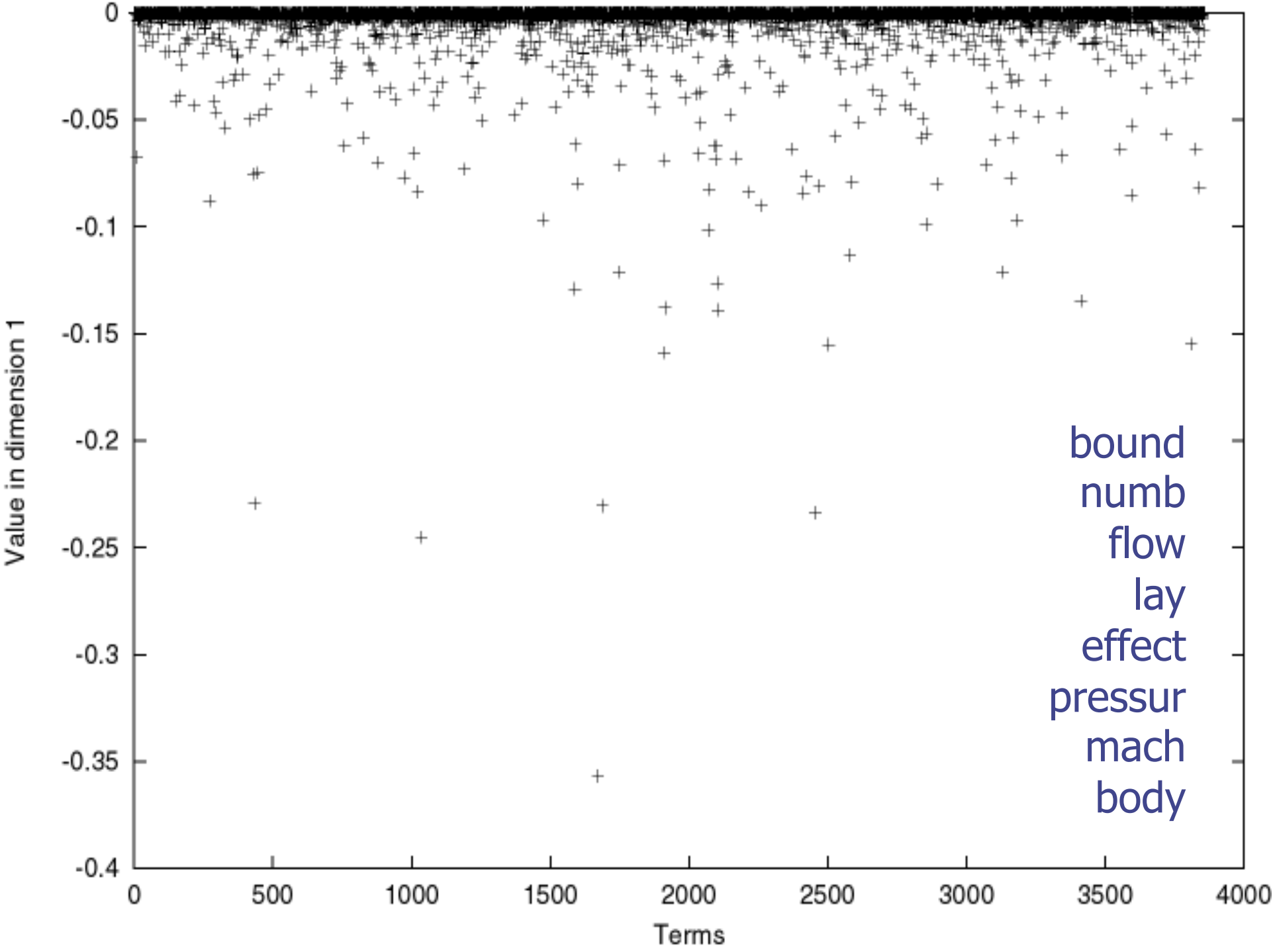
Latent Semantic Indexing

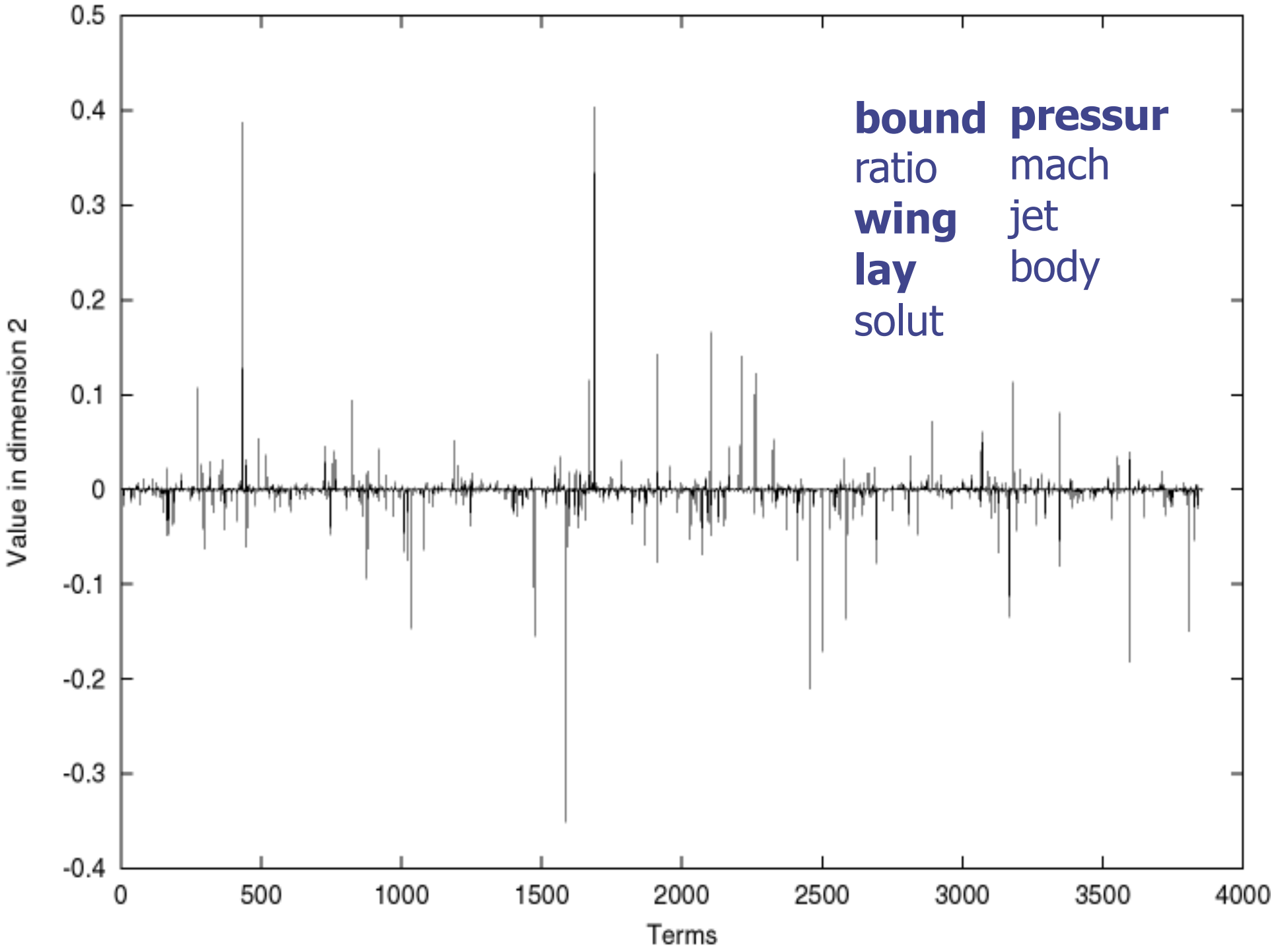


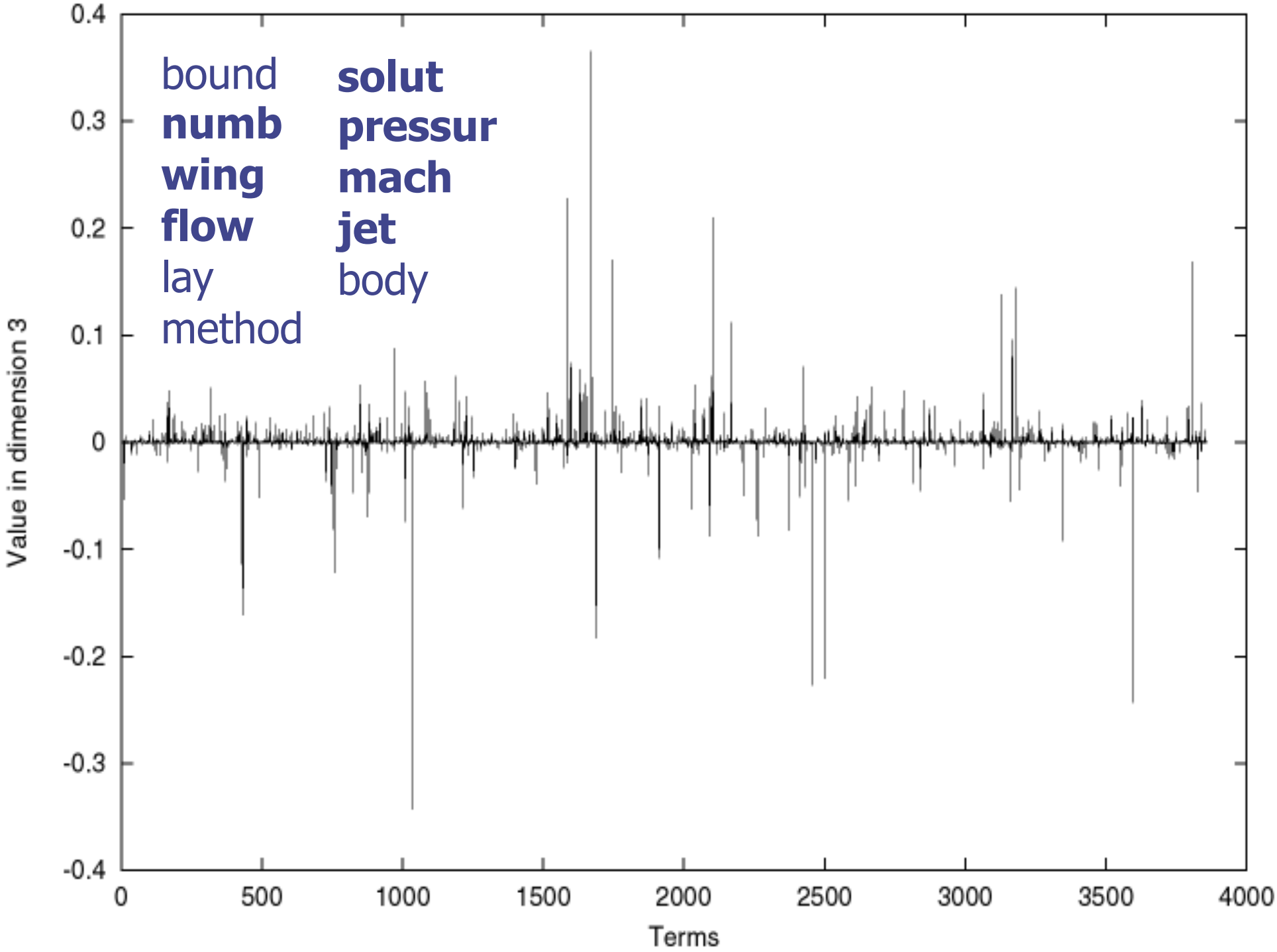
- Compute *singular value decomposition* of a term-document matrix
  - D, a representation of M in *r* dimensions
  - T, a matrix for transforming new documents
  - diagonal matrix Σ gives relative importance of dimensions

LSI Term matrix T

- T matrix
  - gives a vector for each term combo in LSI space
  - for a new document c, c'*T gives a new row in D
  - That is, "fold in" the new document into the LSI space
- LSI is a rotation of the term-space
  - original matrix: terms are d-dimensional
  - new space has lower dimensionality
  - dimensions are groups of terms that tend to co-occur in the same documents
    - synonyms, contextually-related words, variant endings

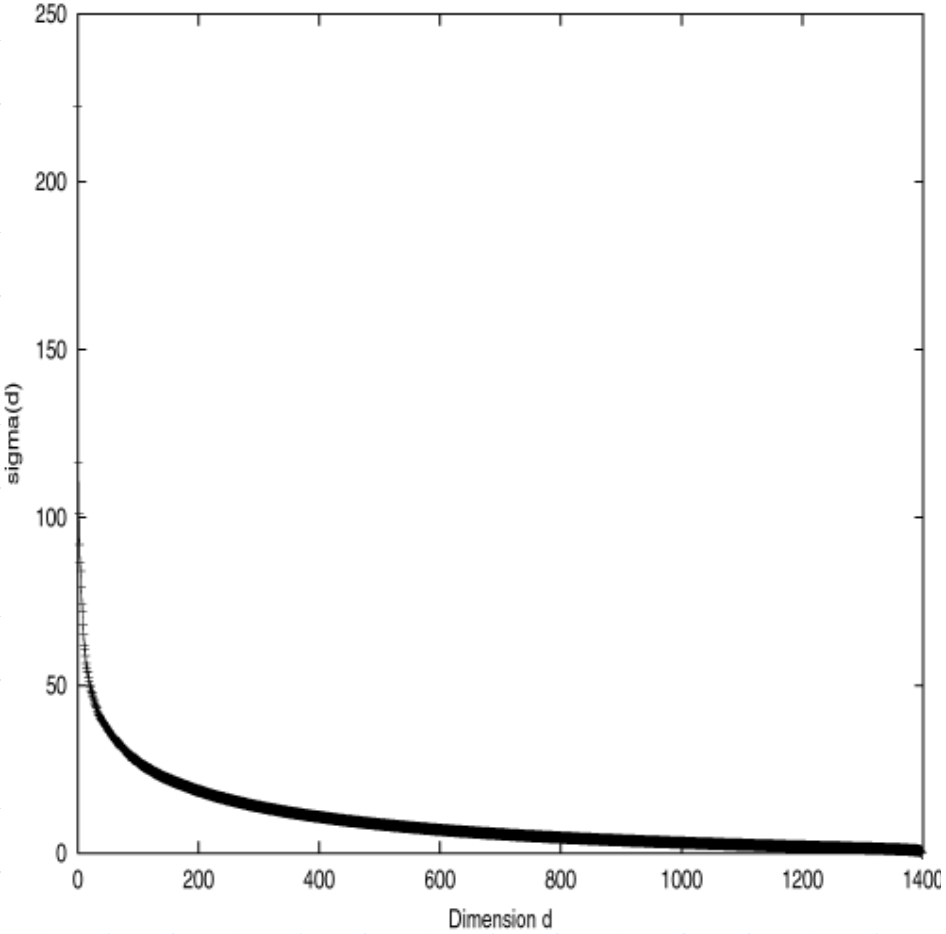

0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
Value in dimension 1
0
500
1000
1500
2000
2500
3000
3500
4000
Terms
bound
numb
flow
lay
effect
pressur
mach
body


0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
Value in dimension 2
0
500
1000
1500
2000
2500
3000
3500
4000
Terms
bound pressur
ratio mach
wing jet
lay body
solut

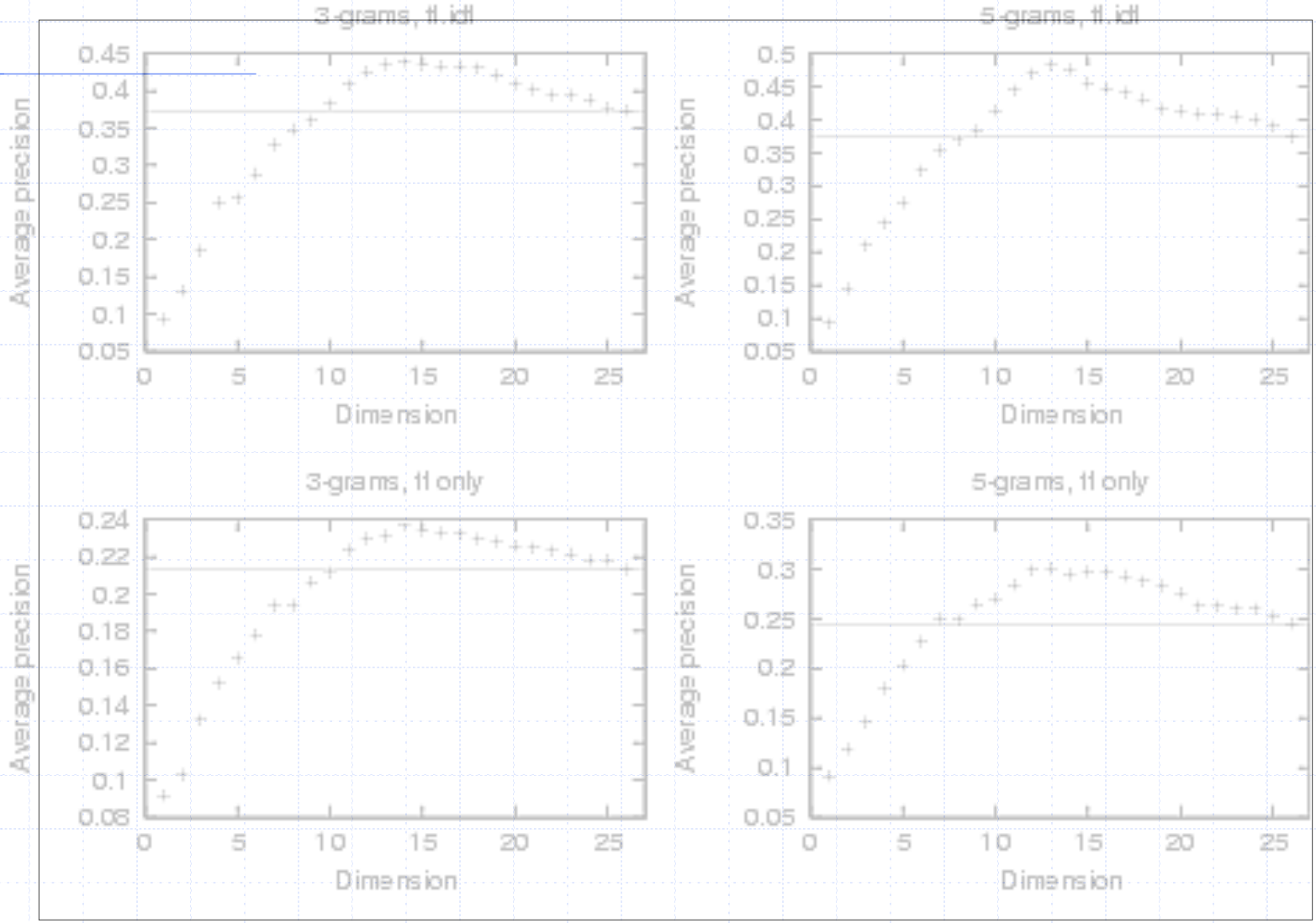

bound
numb
wing
flow
lay
method
solut
pressur
mach
jet
body
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
Value in dimension 3
0
500
1000
1500
2000
2500
3000
3500
4000
Terms

Singular Values

- Σ gives an ordering to the dimensions
  - values tend to drop off very quickly
  - singular values at the tail represent "noise"
  - cutting off low-value dimensions reduces noise and can improve performance



Truncating Dimensions in LSI



Document matrix D

- D matrix
  - coordinates of documents in LSI space
  - same dimensionality as T vectors
  - can compute the similarity between a term and a document

http://lsi.research.telcordia.com/

Improved Retrieval with LSI

- New documents and queries are "folded in"
  - multiply vector by $T\Sigma^{-1}$
- Compute similarity for ranking as in VSM
  - compare queries and documents by dot-product
- Improvements come from
  - reduction of noise
  - no need to stem terms (variants will co-occur)
  - no need for stop list
    - stop words are used uniformly throughout collection, so they tend to appear in the first dimension
  - No speed or space gains, though…

LSI in TREC-3

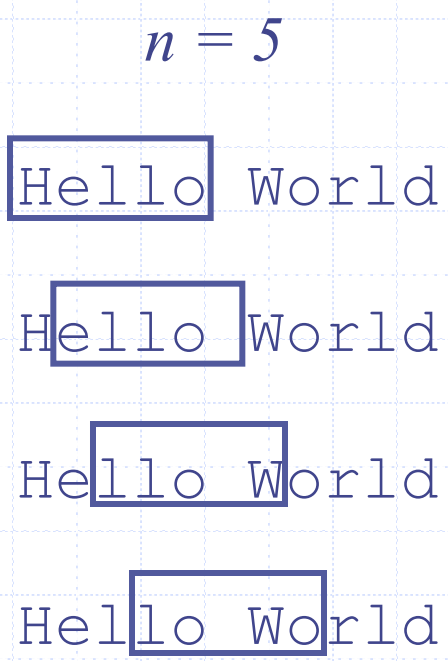
- LSI space computed from a sample of the document collection
- Documents and queries folded into LSI space for comparison
- Improvement in AP with LSI: 5%
  - Improvements up to 20% seen in smaller collections

Other LSI Applications

- Text classification
  - by topic
    - dimension reduction -> good for clustering
  - by language
    - languages have their own stop words
  - by writing style
- Information Filtering
- Cross-language retrieval

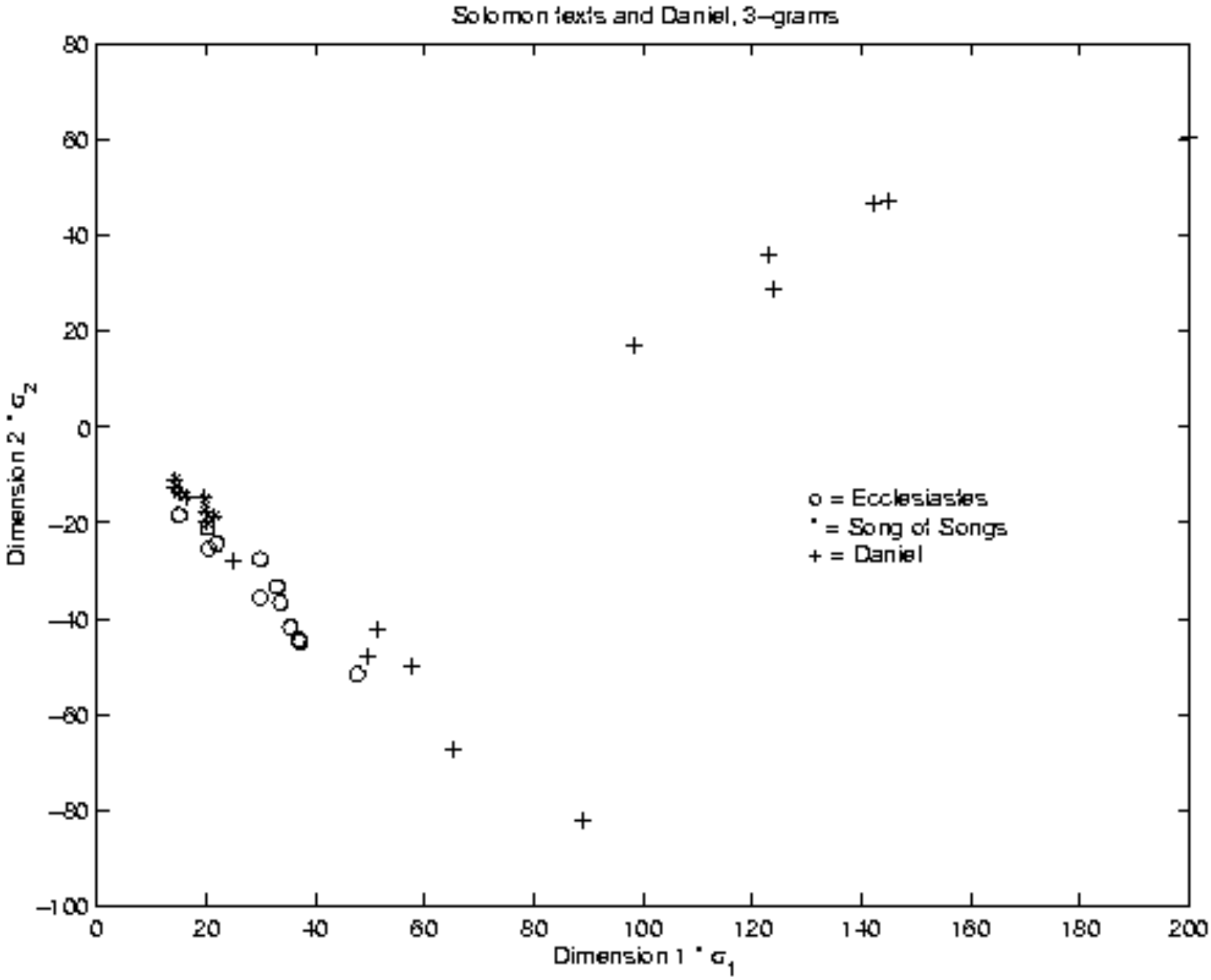
N-gram indexing recap

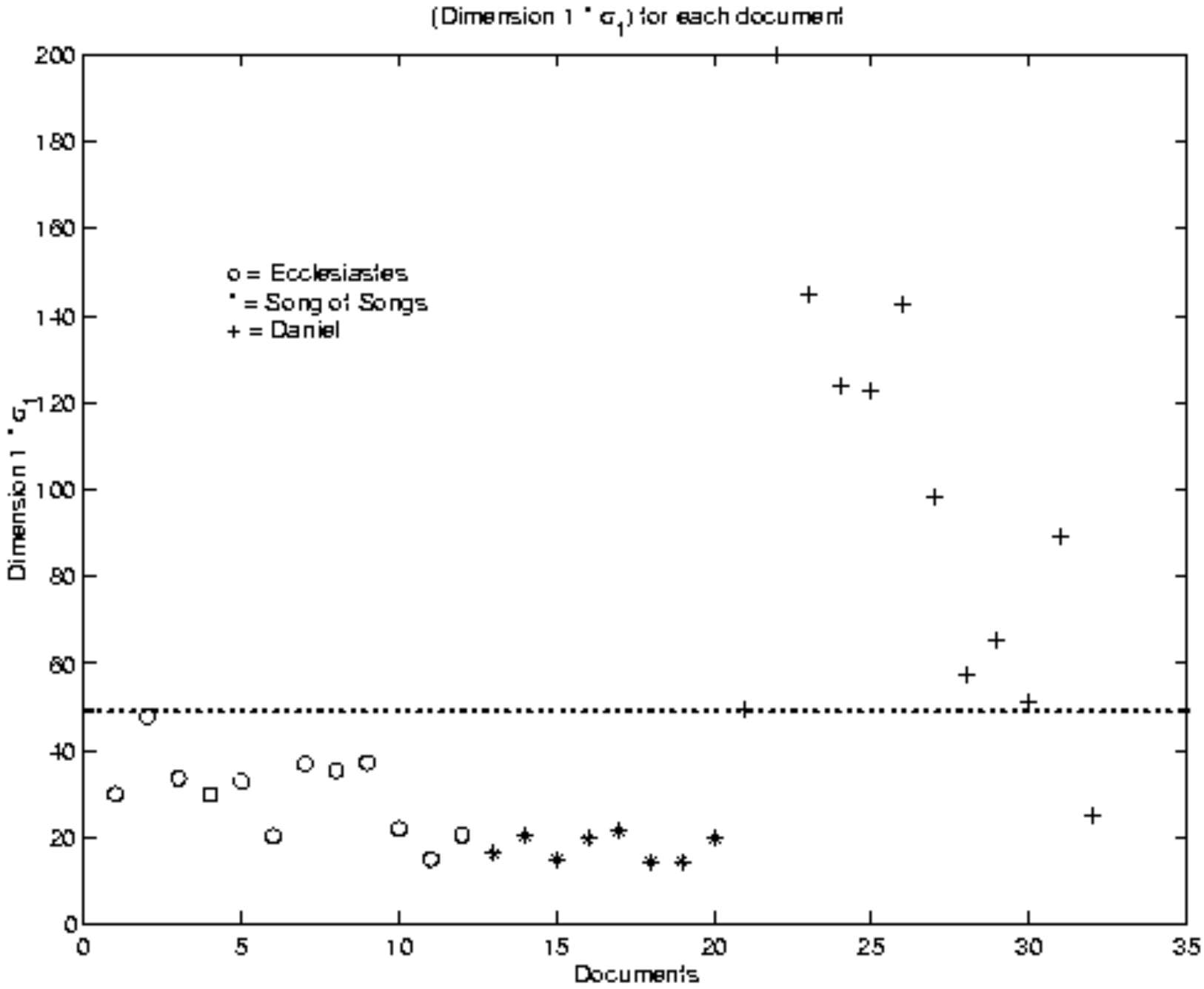
- Index all *n* character sequences
  - language-independent
  - resistant to noisy text
  - no stemming
  - easy to do
- Document ⇒ array of n-gram frequencies

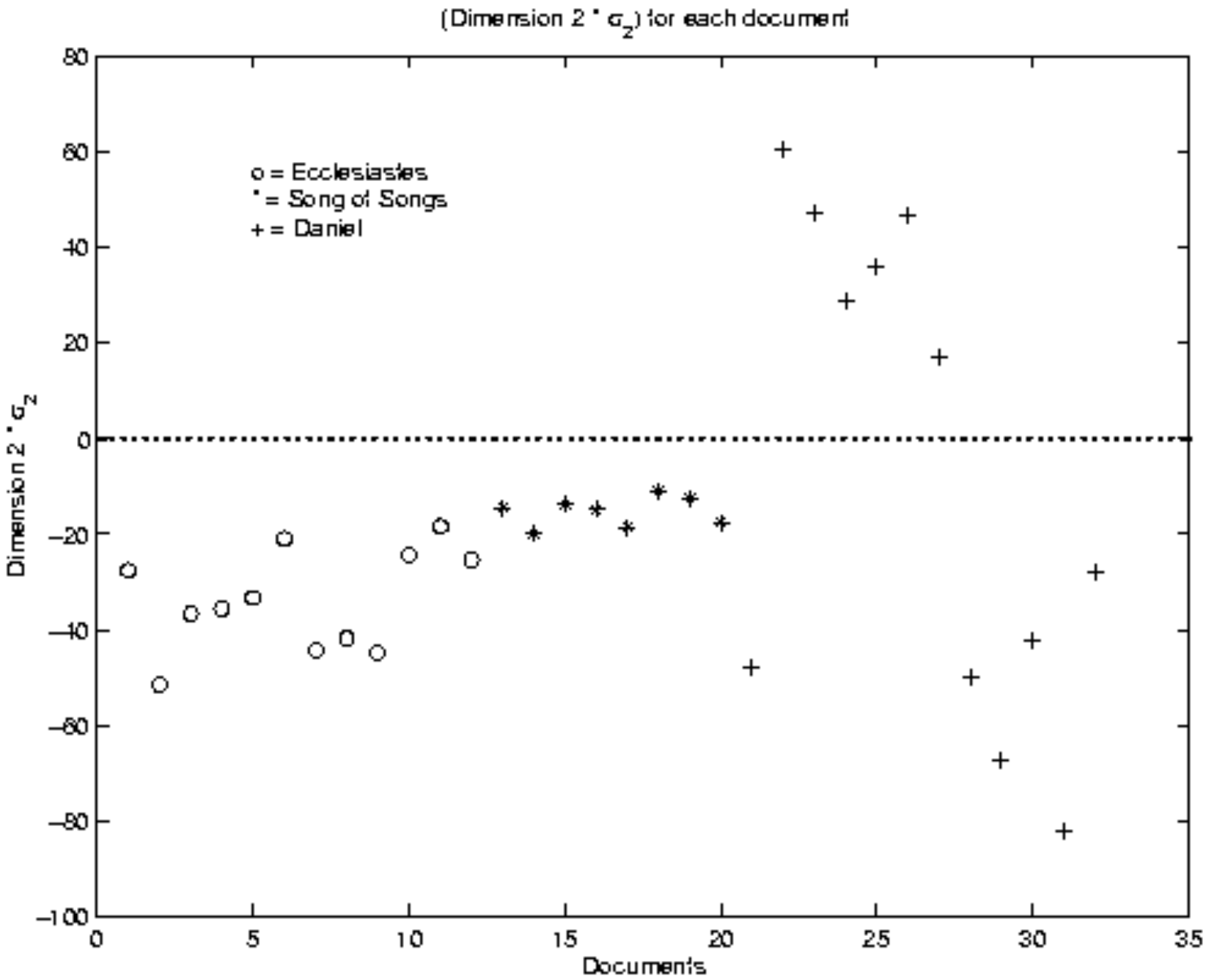


Why N-grams?

- N-grams capture pairs of words
  - Brings out phraseology and word choice
- LSI using n-grams might cluster documents by writing style and/or author
  - a lot of what makes style is word choices and stop word usage
- Small experiment
  - Three biblical Hebrew texts: Ecclesiastes, Song of Songs, Book of Daniel
  - used 3-grams in original Hebrew


Solomon texts and Daniel, 3–grams
Dimension 2 * σ2
Dimension 1 * σ1
o = Ecclesiastes
* = Song of Songs
+ = Daniel
80
60
40
20
0
−20
−40
−60
−80
−100
0
20
40
60
80
100
120
140
160
180
200


(Dimension 1 * σ1) for each document
o = Ecclesiastes
* = Song of Songs
+ = Daniel
Dimension 1 * σ1
Documents
0
20
40
60
80
100
120
140
160
180
200
0
5
10
15
20
25
30
35


(Dimension 2 * σ2) for each document
o = Ecclesiastes
* = Song of Songs
+ = Daniel
Dimension 2 * σ2
Documents
80
60
40
20
0
−20
−40
−60
−80
−100
0
5
10
15
20
25
30
35

Conclusion

- LSI can be a useful technique for reducing the dimensionality of an IR problem
  - reduction can improve effectiveness
  - reduction can find surprising relationships!
- SVD can be expensive to compute on large matrices
- Available tools for working with LSI
  - MATLAB or Octave (small data sets only)
  - SMART (an IR system) with SVDPACK