Latent Semantic Analysis

LING 575 F/G, Spring 2019: Text Representation Learning
Assignment 1
April 8, 2019

Group 2
Hayley Luke, Denise Mak, Thomas Phan, Tev'N Powers

Introduction

- Latent Semantic Analysis (LSA):
  - An algorithm for representing words and documents in a dense vector form in order to study their contextual relationships
  - “Latent semantic”: latent / underlying conceptual meaning in an unstructured document
  - **Inputs**: (1) a corpus of documents and (2) a hyperparameter k (typically 100 - 300)
  - **Outputs**: dense k-dimensional vector representations of both the terms and the documents

- References:

  - S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. “Indexing by Latent Semantics Analysis,” JASIS, 41(6), 1990.
    - Originally called Latent Semantic Indexing in information retrieval context
  - D. Jurafsky and J. Martin. “Speech and Language Processing,” 3rd ed. (online), 2018.
  - Wikipedia, “Latent semantic analysis,” en.wikipedia.org/wiki/Latent_semantic_analysis, retrieved April 3, 2019.

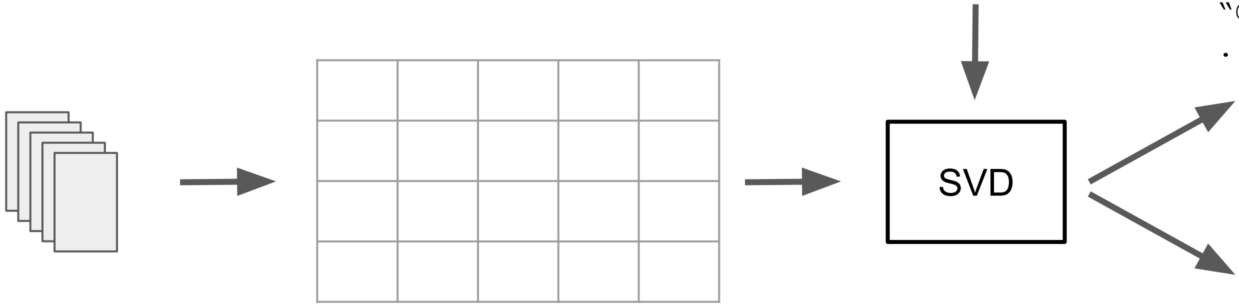
1. Algorithm Details
2. Performance / Working Example
3. Applications

LSA Algorithm Workflow

k-dimensional **dense** vectors for terms

```
"Dog" : [0.123, 0.456, 0.789]
"Cat" : [0.987, 0.654, 0.321]
...
```

Hyperparameter k



Text corpus
with
T terms across
D documents

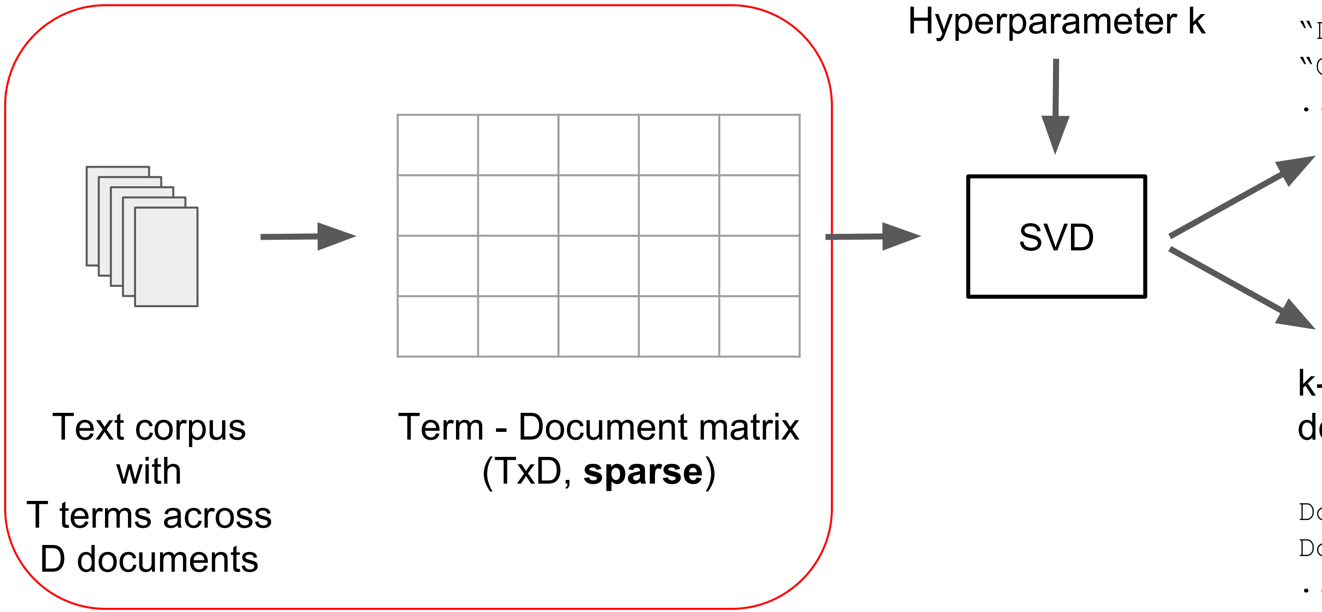
Term - Document matrix
(TxD, **sparse**)

k-dimensional **dense** vectors for documents

```
Doc1 : [0.123, 0.456, 0.789]
Doc2 : [0.987, 0.654, 0.321]
...
```

- Terms typically follow bag-of-words (unigram) model.
- Documents can be sentences, paragraphs, files, webpages, etc.

LSA Algorithm Workflow



k-dimensional **dense** vectors for terms

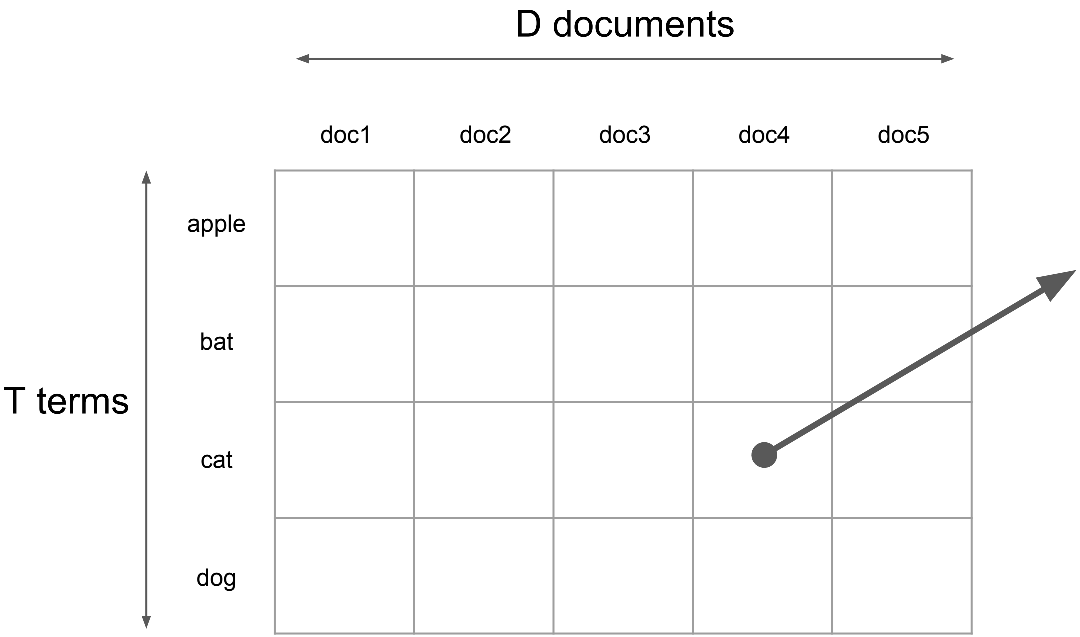
```
"Dog" : [0.123, 0.456, 0.789]
"Cat" : [0.987, 0.654, 0.321]
...
```

k-dimensional **dense** vectors for documents

```
Doc1 : [0.123, 0.456, 0.789]
Doc2 : [0.987, 0.654, 0.321]
...
```

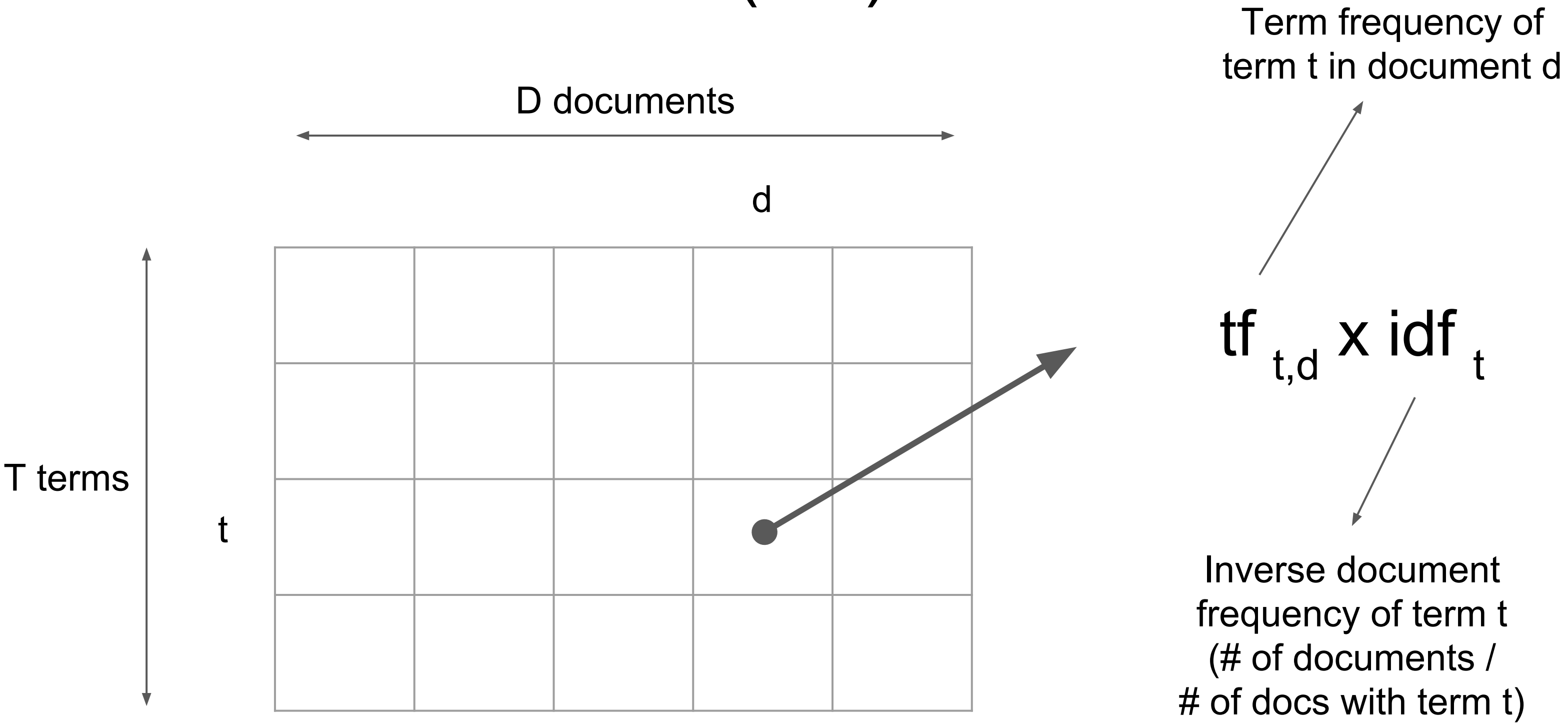
We'll discuss this next

Term-Document Matrix (1/4)

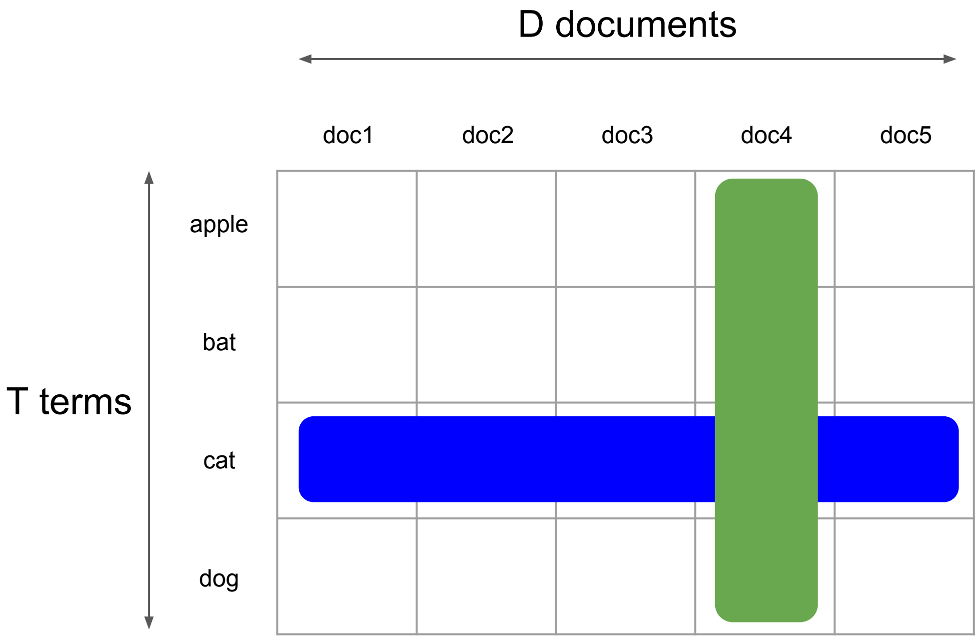


- What values are in the term-document matrix?
- Can be:
  - Term frequency count
  - Binary co-occurrence
  - TF-IDF (next slide)

Term-Document Matrix (2/4)

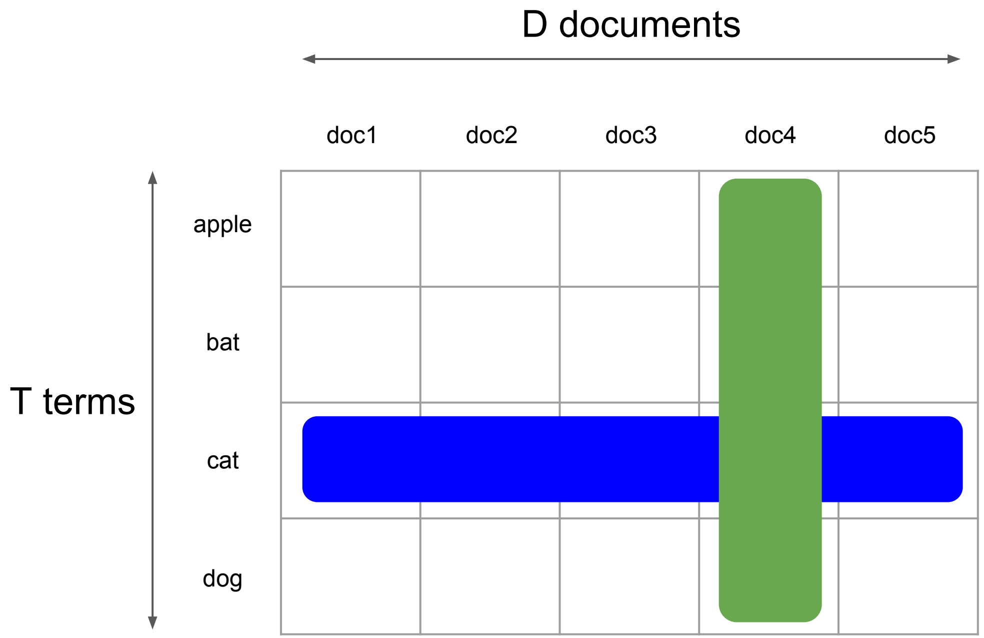


Term-Document Matrix (3/4)



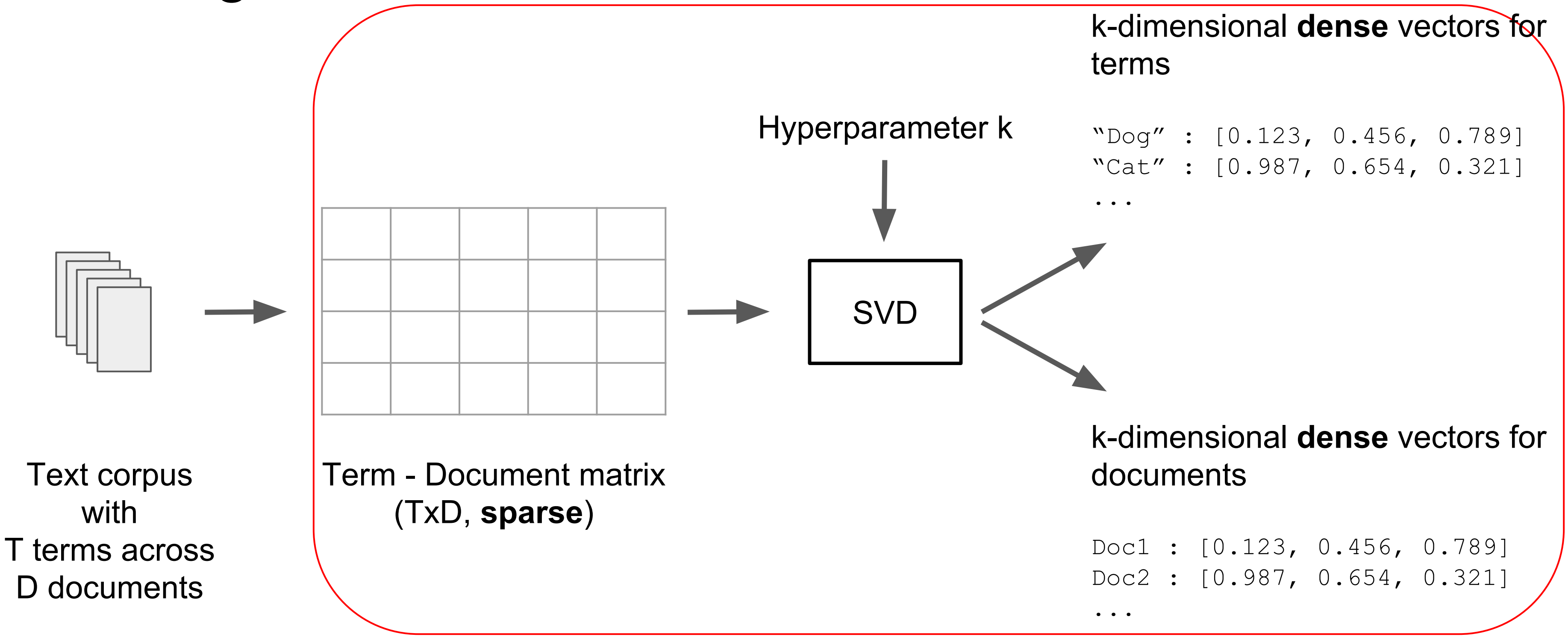
- At this point, we have easy-to-understand vector representations for both the terms and the documents
- Each term is a D-dimensional row vector
- Each document is a T-dimensional column vector

Term-Document Matrix (4/4)



- Problem: both the term and document vectors can be very **large** and **sparse**
  - T can be O(10,000)
  - D can be O(1M)
- Can we make the vectors **dense** O(k=100)?
- We can generate dense vectors using SVD
- But they will be in a latent space

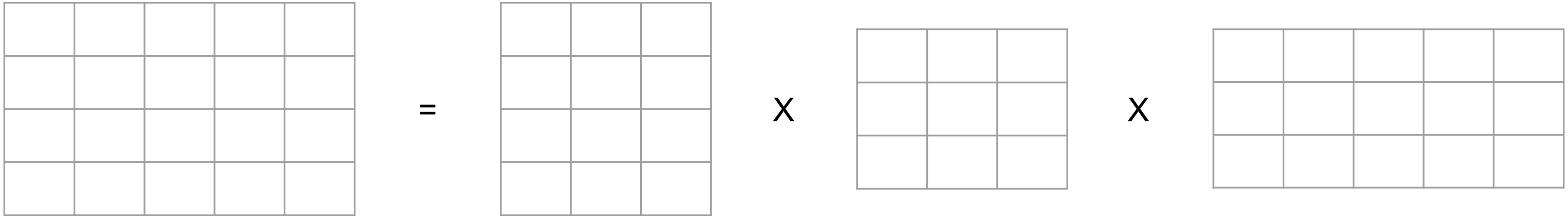
LSA Algorithm Workflow



We'll discuss this next

Singular Value Decomposition (1/2)

- SVD factorizes a rectangular matrix A into three matrices U, Σ, and V^T
- Optimal U, Σ, and V^T are found through linear algebra steps (rotating data onto orthonormal basis vectors while maximizing projected variance)



Original matrix A
with R rows and
C columns

Matrix U
with R rows and
L columns

Matrix Σ
with L rows and
L columns

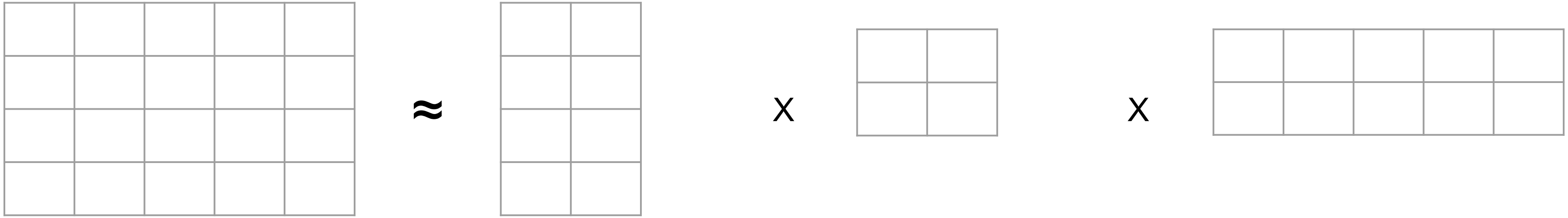
Matrix V^T
with L rows and
C columns

L = rank(A)
L <= min(R, C)

- Columns of U and rows of V^T are orthogonal
- Diagonal of Σ contains *singular values*
- Singular values are sorted by decreasing variance

Singular Value Decomposition (2/2)

- We can apply **truncated SVD** by reducing L to a smaller value k that we can choose, where typically k << L
- Matrix A is **approximated** with lower-dimensional U, Σ, and V^T



Original matrix A
with R rows and
C columns

Matrix U
with R rows and
k columns

k=2 here

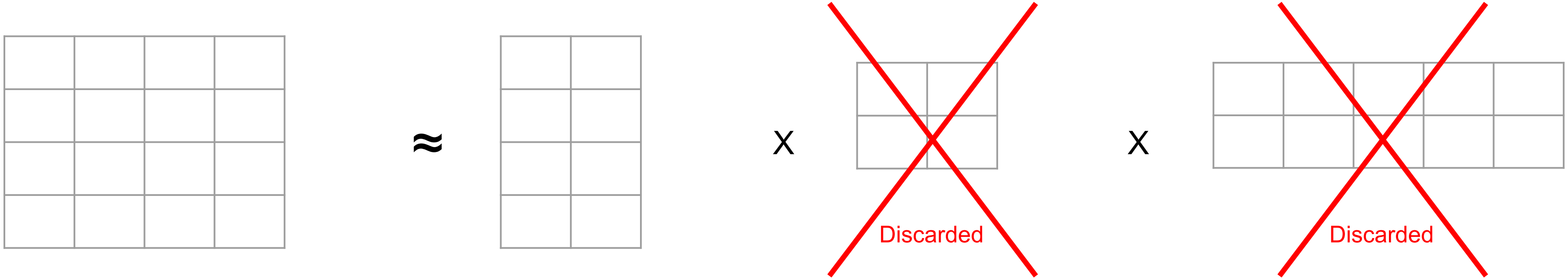
Matrix Σ
with k rows and
k columns

Matrix V^T
with k rows and
C columns

- After truncating, use only first k singular values
- Singular values in Σ are sorted by decreasing variance
- First k → top k explaining most variance

Brief aside: Principal Component Analysis and SVD

- PCA is another well-known dimensionality algorithm that uses SVD
- PCA reduces data from F features to k features
- Columns of U are eigenvectors called the Principal Components (new axes)



Original covariance square matrix A with F rows and F columns (F=4 features)

Matrix U with F rows and k columns

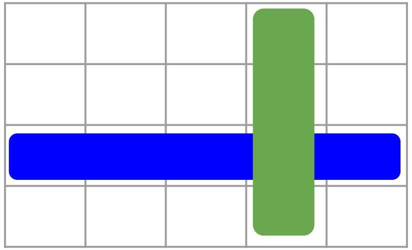
k=2 principal components

Matrix Σ with k rows and k columns

Matrix V^T with k rows and F columns

Applying SVD to the term-document matrix (1/4)

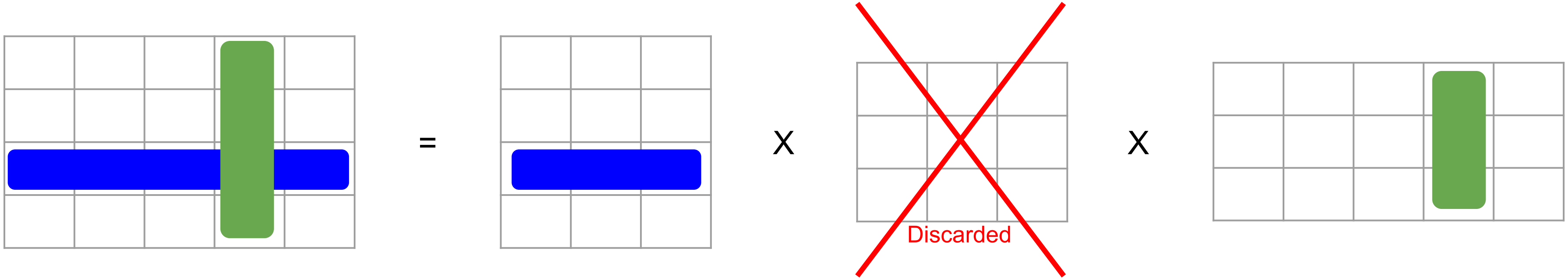
- Recall: we had sparse vector representations for terms and documents
- Here, terms are D-dimensional and documents are T-dimensional



Term-document matrix
T x D

Applying SVD to the term-document matrix (2/4)

- We can apply SVD to get an equivalent representation of the term-document matrix using three matrices
- But the new representation is now in hard-to-understand latent space



Term-document matrix
T x D

Matrix U
T x L

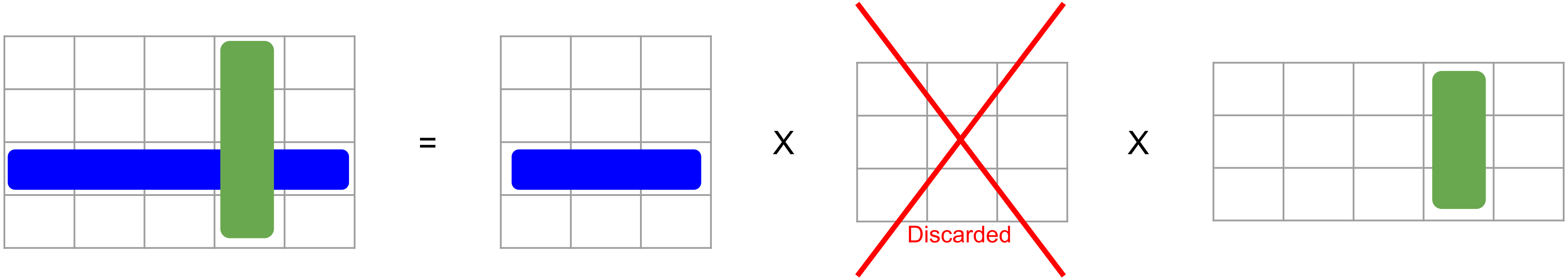
L <= min(T, D)
L=3 here

Matrix Σ
L x L

Matrix V^T
L x D

Applying SVD to the term-document matrix (3/4)

- Before SVD: terms are rows and documents are columns in Term-Doc Matrix
- After SVD: terms are rows in U, and documents are columns in V^T



Term-document matrix
T x D

Matrix U
T x L

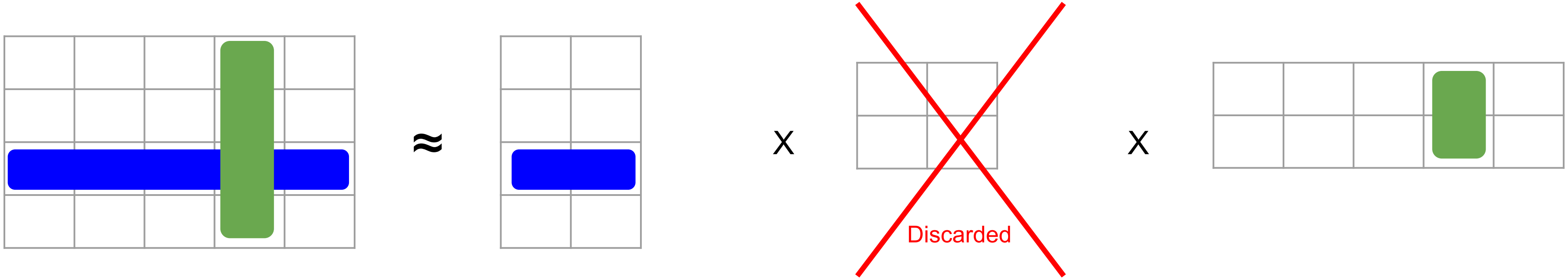
L <= min(T, D)
L=3 here

Matrix Σ
L x L

Matrix V^T
L x D

Applying SVD to the term-document matrix (4/4)

- We can further **truncate** the vectors to a smaller size (here, k=2)
- Before SVD: terms were D-dimensional and documents were T-dimensional
- After SVD and truncation: terms and documents are both k-dimensional



Term-document matrix
T x D

Matrix U
T x k

k=2 here

Matrix Σ
k x k

Matrix V^T
k x D

1. Algorithm Details
2. Performance / Working Example
3. Applications

Performance / Working Example

- Example of generating dense vectors for terms and documents
- Based on example from:

  https://www.datascienceassn.org/sites/default/files/users/user1/lsa_presentation_final.pdf

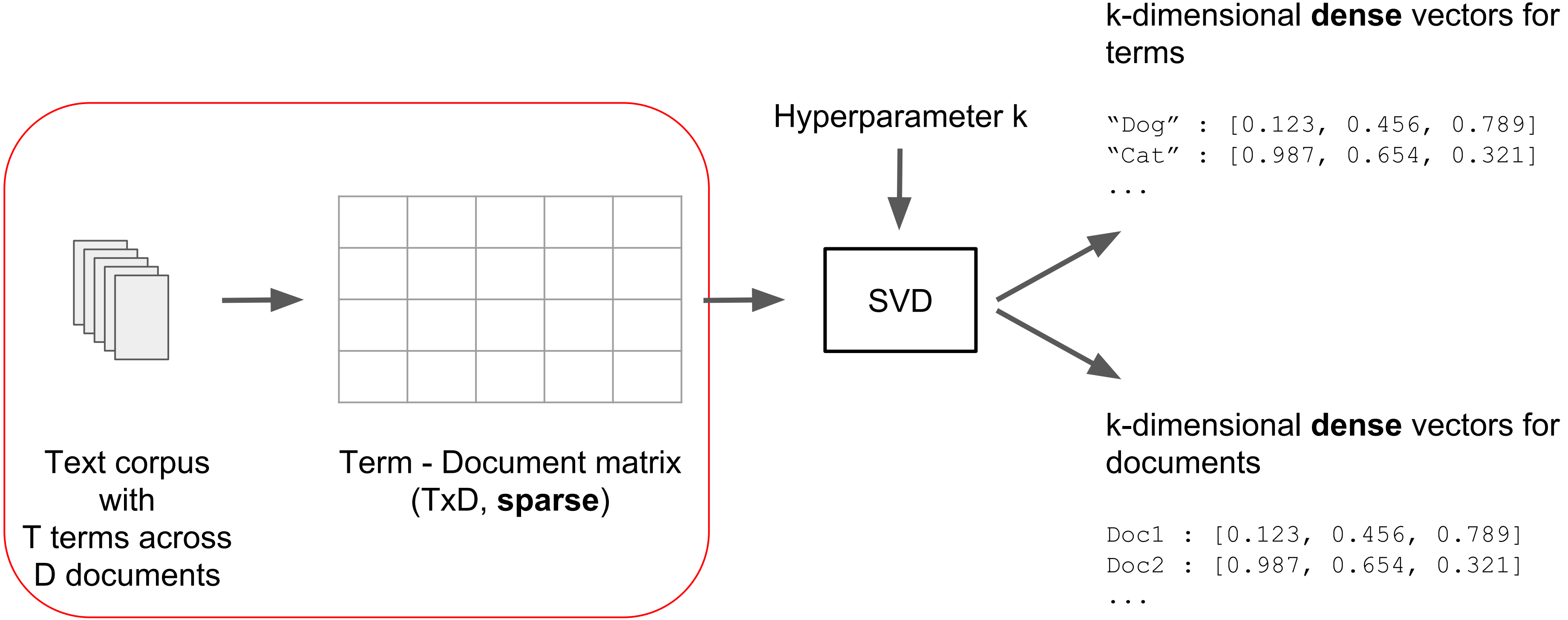
- Used **Python scikit-learn**: CountVectorizer, TfidfVectorizer, TruncatedSVD
- Sample documents:

```
documents = [
    "In a football game Giants defeated Cardinals by 20 points",
    "Cardinals beat Eagles by 20 points in a football game",
    "Eagles scored 20 points but lost the game",
    "Giants lost by 20 points to the Cardinals",
    "Powell is the chairman of the Fed",
    "Powell raised rates by 20 basis points",
    "Trump criticized Powell and the Fed",
    "Trump wants Powell to reduce rates by 20 basis points"
]
```

Sports

Economics

Performance - Building T-D matrix



We’ll discuss this next

Performance - Building T-D matrix

- scikit-learn vectorizer: CountVectorizer, TfidfVectorizer
- Generates term-document matrix with counts or TF-IDF
- Confusingly, the output is actually document-term !!!

```
Signature: tfidf_vectorizer.fit_transform(raw_documents, y=None)
Docstring:
Learn vocabulary and idf, return term-document matrix.

This is equivalent to fit followed by transform, but more efficiently implemented.

Returns
-------
X : sparse matrix, [n_samples, n_features]
    Tf-idf-weighted document-term matrix.
```

!!!

Performance - Building T-D matrix

This matrix is sparse:
128 of 168 entries are 0

- CountVectorizer output

Terms (8-dimensional)

Documents (21-dimensional)

	20	basis	beat	cardinals	chairman	criticized	defeated	eagles	fed	football	game	giants	lost	points	powell	raised	rates	reduce	scored	trump	wants
In a football game Giants defeated Cardinals by 20 points	1	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0
Cardinals beat Eagles by 20 points in a football game	1	0	1	1	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0
Eagles scored 20 points but lost the game	1	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	1	0	0
Giants lost by 20 points to the Cardinals	1	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
Powell is the chairman of the Fed	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
Powell raised rates by 20 basis points	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
Trump criticized Powell and the Fed	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0
Trump wants Powell to reduce rates by 20 basis points	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	1

Performance - Building T-D matrix

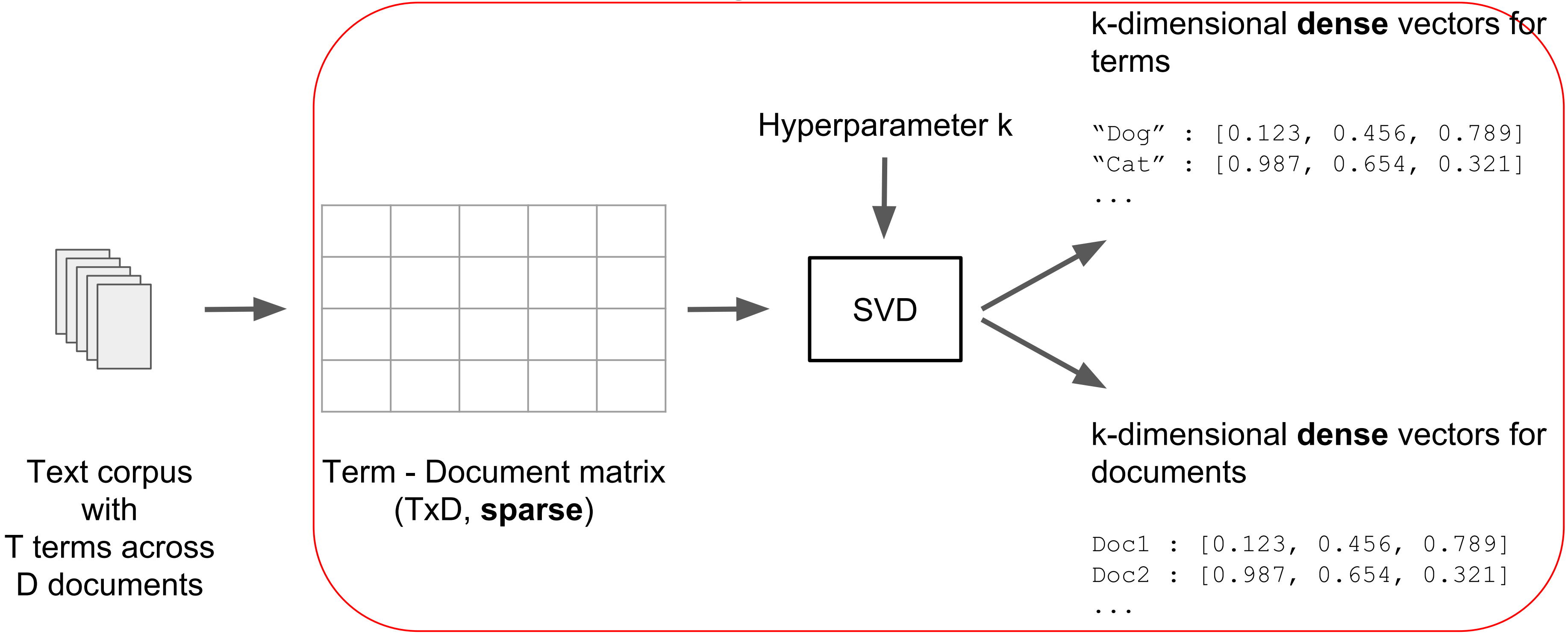
- TfidfVectorizer output

Terms (8-dimensional)

Documents (21-dimensional)

	20	basis	beat	cardinals	chairman	criticized	defeated	eagles	fed	football	game	giants	lost	points	powell	raised	rates	reduce	scored	trump	wants
In a football game Giants defeated Cardinals by 20 points	0.251	0.000	0.000	0.364	0.000	0.000	0.503	0.000	0.000	0.422	0.364	0.422	0.000	0.251	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Cardinals beat Eagles by 20 points in a football game	0.251	0.000	0.503	0.364	0.000	0.000	0.000	0.422	0.000	0.422	0.364	0.000	0.000	0.251	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Eagles scored 20 points but lost the game	0.270	0.000	0.000	0.000	0.000	0.000	0.000	0.453	0.000	0.000	0.391	0.000	0.453	0.270	0.000	0.000	0.000	0.000	0.540	0.000	0.000
Giants lost by 20 points to the Cardinals	0.321	0.000	0.000	0.464	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.538	0.538	0.321	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Powell is the chairman of the Fed	0.000	0.000	0.000	0.000	0.689	0.000	0.000	0.000	0.578	0.000	0.000	0.000	0.000	0.000	0.437	0.000	0.000	0.000	0.000	0.000	0.000
Powell raised rates by 20 basis points	0.275	0.461	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.275	0.349	0.550	0.461	0.000	0.000	0.000	0.000
Trump criticized Powell and the Fed	0.000	0.000	0.000	0.000	0.000	0.597	0.000	0.000	0.500	0.000	0.000	0.000	0.000	0.000	0.378	0.000	0.000	0.000	0.000	0.500	0.000
Trump wants Powell to reduce rates by 20 basis points	0.223	0.374	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.223	0.283	0.000	0.374	0.447	0.000	0.374	0.447

Performance - Generating dense vectors



We'll discuss this next

Performance - Terms as dense 2-D vectors

- Applying SVD with truncation to k=2 dimensions
- Term vector reduced from sparse 8-dimensional to dense 2-dimensional

```
lsa = TruncatedSVD(n_components=2, algorithm='arpack')
dtm_lsa = lsa.fit_transform(doc_term_matrix_tfidf)
```

Before SVD:
Terms are
sparse
8-dim vectors

giants
0.422
0.000
0.000
0.538
0.000
0.000
0.000
0.000

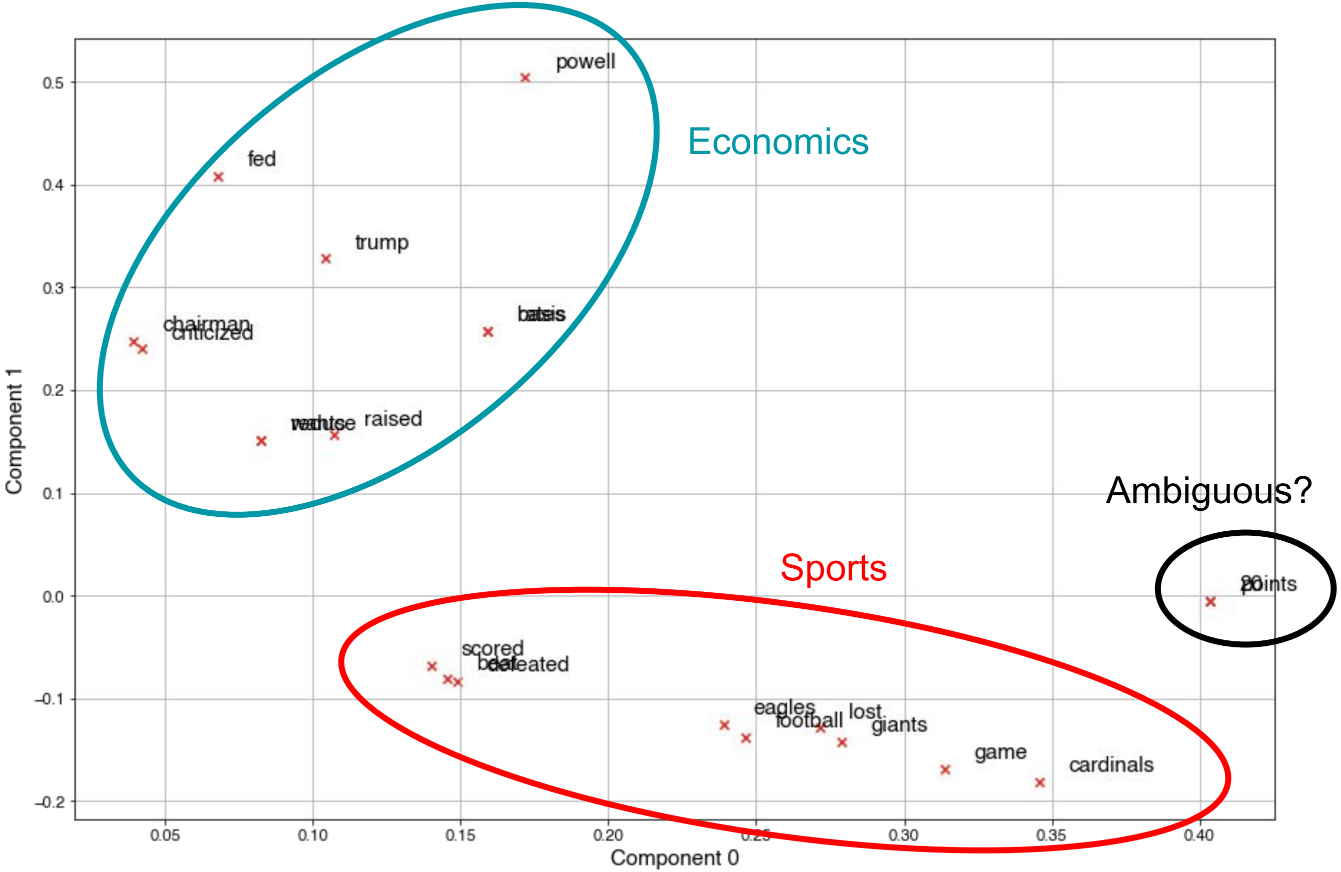
After SVD:
Terms are
dense
2-dim vectors

	0	1
20	0.404	-0.005
basis	0.159	0.258
beat	0.145	-0.081
cardinals	0.346	-0.181
chairman	0.039	0.247
criticized	0.042	0.240
defeated	0.149	-0.083
eagles	0.239	-0.125
fed	0.068	0.408
football	0.247	-0.138

	0	1
giants	0.279	-0.142
lost	0.271	-0.129
points	0.404	-0.005
powell	0.172	0.504
raised	0.107	0.156
rates	0.159	0.258
reduce	0.082	0.151
scored	0.140	-0.068
trump	0.104	0.328
wants	0.082	0.151

Performance - Terms as dense 2-D vectors

- "In a football game Giants defeated Cardinals by 20 points"
- "Cardinals beat Eagles by 20 points in a football game"
- "Eagles scored 20 points but lost the game"
- "Giants lost by 20 points to the Cardinals"
- "Powell is the chairman of the Fed"
- "Powell raised rates by 20 basis points"
- "Trump criticized Powell and the Fed"
- "Trump wants Powell to reduce rates by 20 basis points"



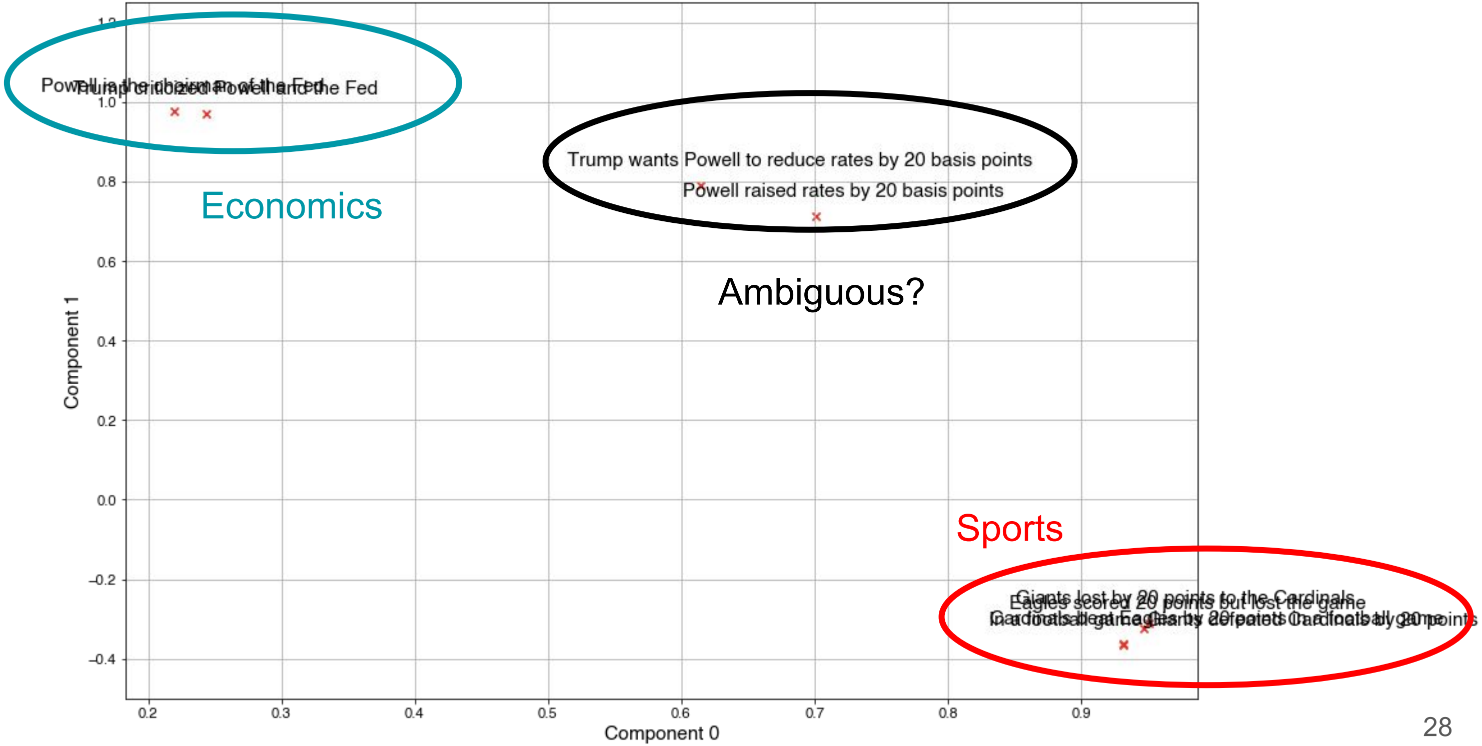
Performance - Documents as dense 2-D vectors

- Document vector reduced from sparse 21-dimensional to dense 2-dimensional

	0	1
In a football game Giants defeated Cardinals by 20 points	0.931	-0.365
Cardinals beat Eagles by 20 points in a football game	0.932	-0.364
Eagles scored 20 points but lost the game	0.947	-0.322
Giants lost by 20 points to the Cardinals	0.951	-0.310
Powell is the chairman of the Fed	0.220	0.976
Powell raised rates by 20 basis points	0.701	0.713
Trump criticized Powell and the Fed	0.244	0.970
Trump wants Powell to reduce rates by 20 basis points	0.615	0.789

Performance - Documents as dense 2-D vectors

- "In a football game Giants defeated Cardinals by 20 points"
- "Cardinals beat Eagles by 20 points in a football game"
- "Eagles scored 20 points but lost the game"
- "Giants lost by 20 points to the Cardinals"
- "Powell is the chairman of the Fed"
- "Powell raised rates by 20 basis points"
- "Trump criticized Powell and the Fed"
- "Trump wants Powell to reduce rates by 20 basis points"



1. Algorithm Details
2. Performance / Working Example
3. Applications

Applications

- Word relationship/synonymy
  - Different words describing the same concept
  - LSA can be used to aid searches where the query matches the concept but not the specific words
    - e.g. searching for “doctor” does not return documents with “physician”
  - Confounding factor - polysemy
    - Same word has multiple meanings
    - LSA has no way of differentiating the meanings on its own, so results can be skewed
- Memory studies
  - Free recall
    - Use LSA to measure semantic similarity between words
    - Correlate with probability of word to be recalled from a random set of nouns
    - Faster to recall semantically similar words vs non-similar

Applications

- Document classification
- Searching/information retrieval
  - Combine document classification with word synonymy
- Dream content analysis
  - Find semantic associations in “dream reports”
  - e.g. comparing words related to “run” in reports from dreams vs real life
    - Dreams: chase, running, scream, chasing, escape, runs, chases, grab, hide, chased, yells, safety
    - Non-dream: running, runs, ran, go, operate, organise, compete, starts, jump, loaded, weekend, vms, startx, marathons, mkdir