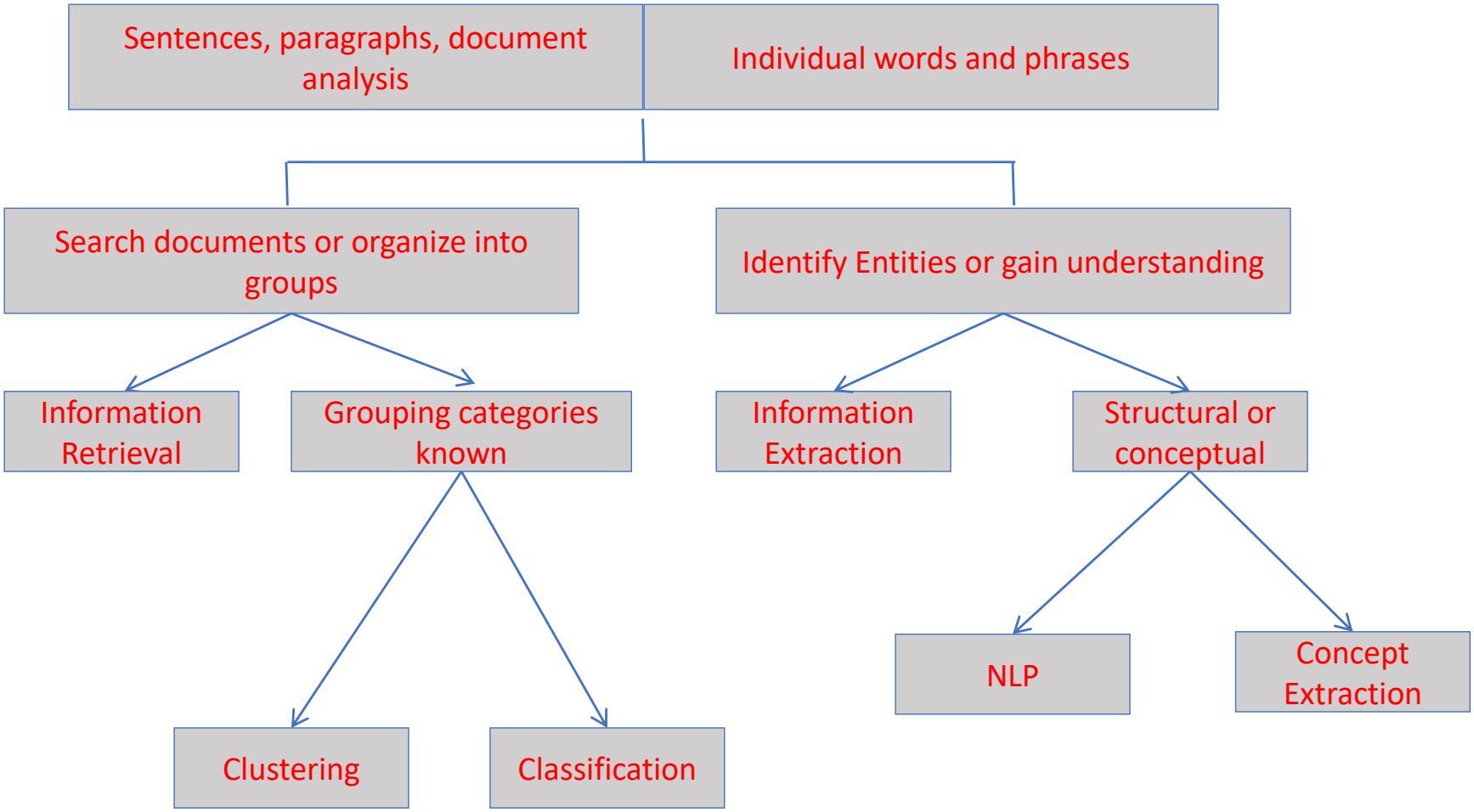
Text Mining

What is Text Mining?

Text mining is the process of compiling, organizing, and analyzing large document collections to support the delivery of targeted types of information to analysts and decision makers and to discover relationships between related facts that span wide domains of inquiry.

Text Mining Tasks



Topics in Text Mining

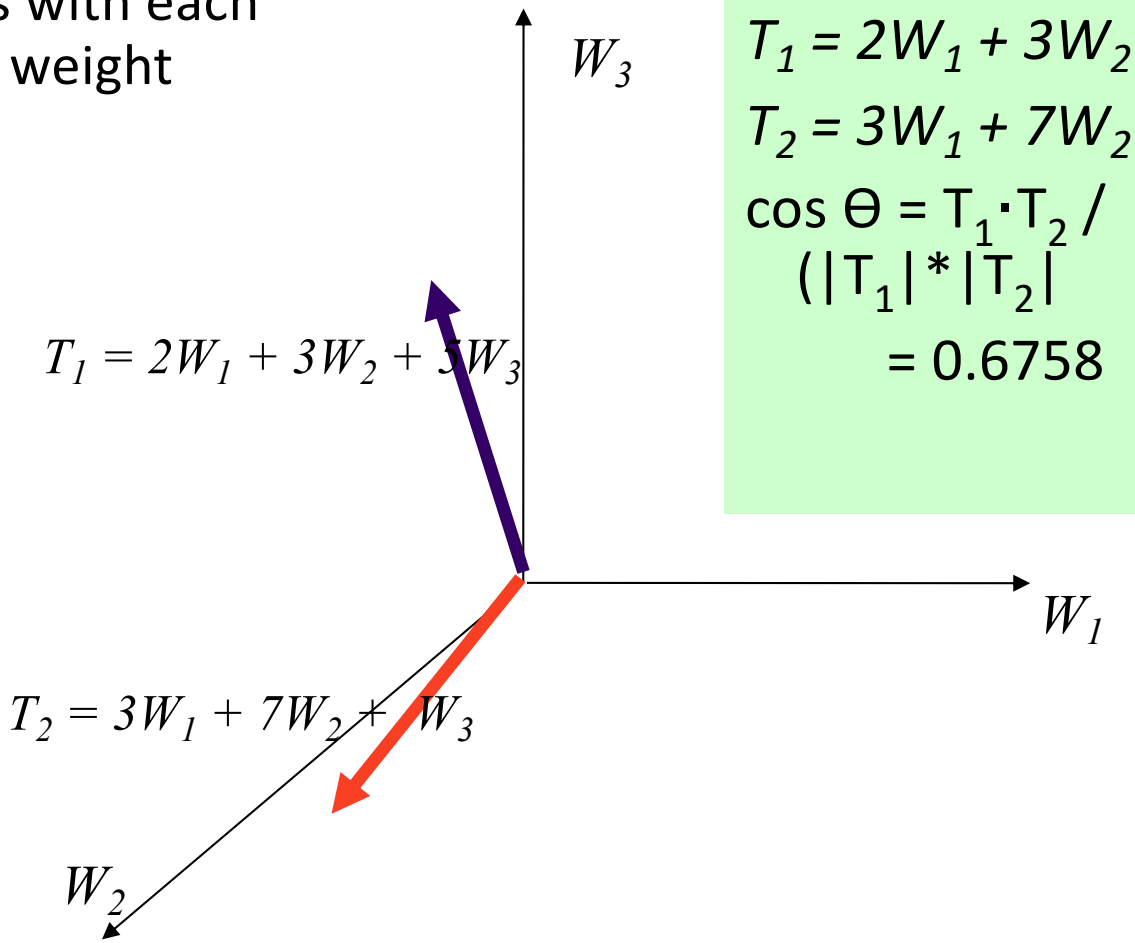
- Text and word similarities
- Text Clustering and Topic Modeling
- Information extraction/Entity recognition
- Sentiment analysis
- Information filtering

Not a comprehensive list

Text Similarity

Vector Space Model for Document Similarity

- Represents each document as a bag of words with each word having a weight



Example:

$T_1 = 2W_1 + 3W_2 + 5W_3$

$T_2 = 3W_1 + 7W_2 + W_3$

$\cos \Theta = T_1 \cdot T_2 / (|T_1| * |T_2|$

$= 0.6758$

Hurricane Gilbert swept toward the Dominican Republic Sunday , and the Civil Defense alerted its heavily populated south coast to prepare for high **winds**, heavy **rains** and high seas.

The **storm** was approaching from the southeast with sustained **winds** of 75 mph gusting to 92 mph .

“There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday .

Cabral said residents of the province of Barahona should closely follow **Gilbert** 's movement .

An estimated 100,000 people live in the province, including 70,000 in the city of Barahona , about 125 miles west of Santo Domingo .

Tropical **Storm Gilbert** formed in the eastern Caribbean and strengthened into a **hurricane** Saturday night

Document1

- Gilbert: 3
- Hurricane: 2
- Rains: 1
- Storm: 2
- Winds: 2

The National **Hurricane** Center in Miami reported its position at 2a.m. Sunday at latitude 16.1 north , longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.

The National Weather Service in San Juan , Puerto Rico , said **Gilbert** was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the **storm**.

The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6p.m. Sunday.

Strong **winds** associated with the **Gilbert** brought coastal flooding , strong southeast **winds** and up to 12 feet to Puerto Rico 's south coast.

Document2

- Gilbert: 2
- Hurricane: 1
- Rains: 0
- Storm: 1
- Winds: 2

Words in red are assumed keywords

Cosine Similarity: 0.9439

Issues with the Vector Space Model

- Ignores semantic similarities
  - *I own a dog* vs. *I have a pet*
  - Solution: Supplement with Word Similarity
- Polysemy: Words often have a multitude of meanings and different types of usage, for example, *surfing*
- Synonymy: Different words mean the same; for example, “automobile” when querying on “car”

SVD & LSI

Singular Value Decomposition (SVD)

- A technique for handling matrices (sets of equations) that do not have an inverse. This includes square matrices whose determinant is zero and all rectangular matrices.
- Handy mathematical technique that has application to many problems
- According to SVD, an arbitrary matrix **F** of size m x n can be expressed as

$$\mathbf{U}^t\mathbf{F}\mathbf{V} = \mathbf{\Lambda}^{1/2},$$

where **U** and **V** are orthogonal matrices of size m x m and n x n, respectively. The matrix $\mathbf{\Lambda}^{1/2}$ is a mxn diagonal matrix as shown below

$$\Lambda^{1/2} = \begin{bmatrix} \lambda_1^{1/2} & & & \vdots & \\ & \ddots & & \vdots & 0 \\ & & \lambda_r^{1/2} & \vdots & \\ \cdots & \cdots & \cdots & \vdots & \cdots \\ & 0 & & \vdots & 0 \end{bmatrix}$$

This was covered in dimensionality reduction also.

Singular Value Decomposition (SVD)

- We can also write $\mathbf{F} = \mathbf{U}\,\mathbf{\Lambda}^{1/2}\,\mathbf{V}^t$, since **U** and **V** are orthogonal matrices.
- The columns of the matrix **U** are composed of the eigenvectors of the symmetric matrix $\mathbf{FF}^t$, and the columns of the matrix **V** are the eigenvectors of the symmetric matrix $\mathbf{F}^t\mathbf{F}$.
- In terms of eigenvectors, it is possible to express matrix **F** as follows:

$$\mathbf{F} = \sum_{j=1}^{r} \lambda_j^{1/2} \mathbf{u}_j \mathbf{v}_j^t$$

- The outer products of the eigenvectors above form a set of unit rank matrices each of which is scaled by a corresponding singular value of **F**.

SVD

- The diagonal elements of the matrix $\mathbf{\Lambda}^{1/2}$ are called the singular values of **F**
- If **F** is singular, some of the diagonal elements will be 0
- In general *rank*(**F**) = number of nonzero *diagonal elements of* $\mathbf{\Lambda}^{1/2}$

Example

Let us find SVD for matrix $\mathbf{F} = \begin{bmatrix} 6 & 6 \\ 0 & 1 \\ 4 & 0 \\ 0 & 6 \end{bmatrix}$

Step 1: Compute **V**.

$\mathbf{F}^t\mathbf{F} = \begin{bmatrix} 52 & 36 \\ 36 & 73 \end{bmatrix}$

The eigenvalues of the above matrix are 100 and 25, respectively.
The corresponding eigenvectors are $[0.6\ 0.8]^t$ and $[0.8\ -0.6]^t$.

Step 2: Compute **U**.

$\mathbf{F}\mathbf{F}^t = \begin{bmatrix} 72 & 6 & 24 & 36 \\ 6 & 1 & 0 & 6 \\ 24 & 0 & 16 & 0 \\ 36 & 6 & 0 & 36 \end{bmatrix}$

This matrix has only two nonzero eigenvalues (same as above) and the corresponding eigenvectors are $[0.84\ 0.08\ 0.24\ 0.48]^t$ and $[0.24\ -0.12\ 0.64\ -0.72]^t$.

Example (Cntnd)

Step 3: Express **F** in SVD form:

$$\mathbf{F} = (100)^{1/2}[0.84\ 0.08\ 0.24\ 0.48]^{t}[0.6\ 0.8] +$$
$$(25)^{1/2}[0.24\ -0.12\ 0.64\ -0.72]^{t}[0.8\ -0.6]$$

```
from scipy import linalg
import numpy as np
np.set_printoptions(precision=2)
```

```
F = np.array([[6,6],[0,1],[4,0],[0,6]])
print(F)
```

```
[[6 6]
 [0 1]
 [4 0]
 [0 6]]
```

SVD using numpy

```
U, s, V = linalg.svd(F)
```

```
print(U)
print(s)
print(V)
```

```
[[-0.84  0.24 -0.44 -0.2 ]
 [-0.08 -0.12  0.48 -0.87]
 [-0.24  0.64  0.66  0.3 ]
 [-0.48 -0.72  0.36  0.35]]
[10.  5.]
[[-0.6 -0.8]
 [ 0.8 -0.6]]
```

```
# Lets get back F
sigma = np.zeros((4, 2))# 4x2 is the size of F
for i in range(min(4, 2)):
    sigma[i, i] = s[i]
Fa = np.dot(U,np.dot(sigma,V))
print(Fa)
```

```
[[ 6.00e+00  6.00e+00]
 [ 0.00e+00  1.00e+00]
 [ 4.00e+00 -1.11e-15]
 [-4.44e-16  6.00e+00]]
```

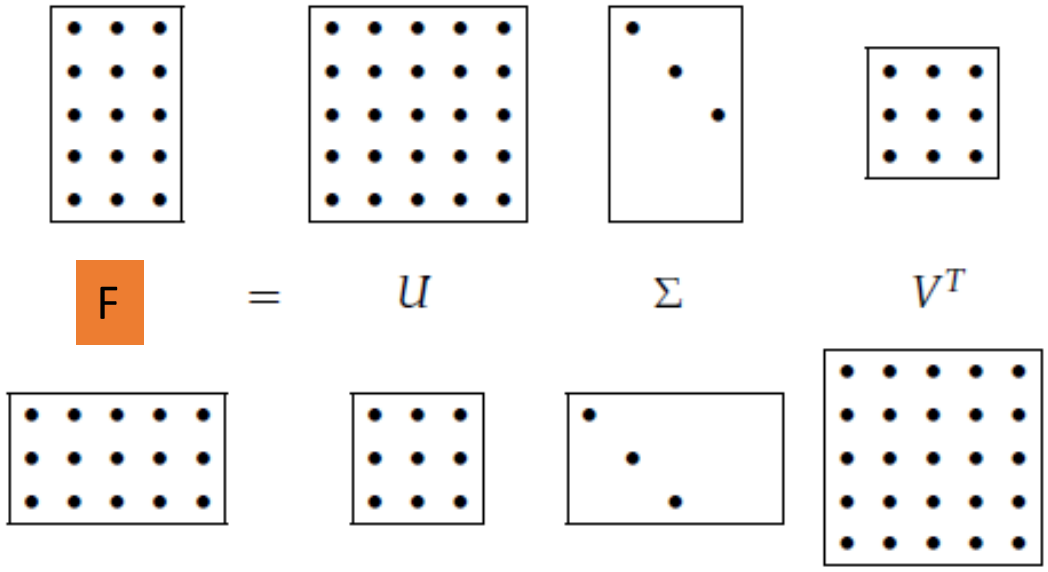
```
print(np.round(Fa))
```

```
[[ 6.  6.]
 [ 0.  1.]
 [ 4. -0.]
 [-0.  6.]]
```

```
# Lets construct an approximation to F by keeping only the first singular value
sigma = np.zeros((4, 2))# 4x2 is the size of F
for i in range(min(4, 1)):
    sigma[i, i] = s[i]
Fa = np.dot(U,np.dot(sigma,V))
print(Fa)
```

```
[[5.04 6.72]
 [0.48 0.64]
 [1.44 1.92]
 [2.88 3.84]]
```

SVD Illustration



The top half illustrates the case when matrix F has more rows than columns. The lower half shows the opposite case.

SVD and Matrix Similarity

- One common definition for the norm of a matrix is the Frobenius norm:

$$\|F\|_{\text{FNorm}} = \sum_i \sum_j f_{ij}^2$$

- Frobenius norm can be computed from SVD

$$\|F\|_{\text{FNorm}} = \sum_i \lambda_i^2$$

- So changes to a matrix can be evaluated by looking at changes to singular values

Approximation using SVD

- An approximation of matrix **F** can be obtained by expressing it as the sum of *k* m x n rank-one matrices:

$$\hat{\mathbf{F}} = \sum_{j=1}^{k} \lambda_j^{1/2} \mathbf{u}_j \mathbf{v}_j^t, k \le r$$

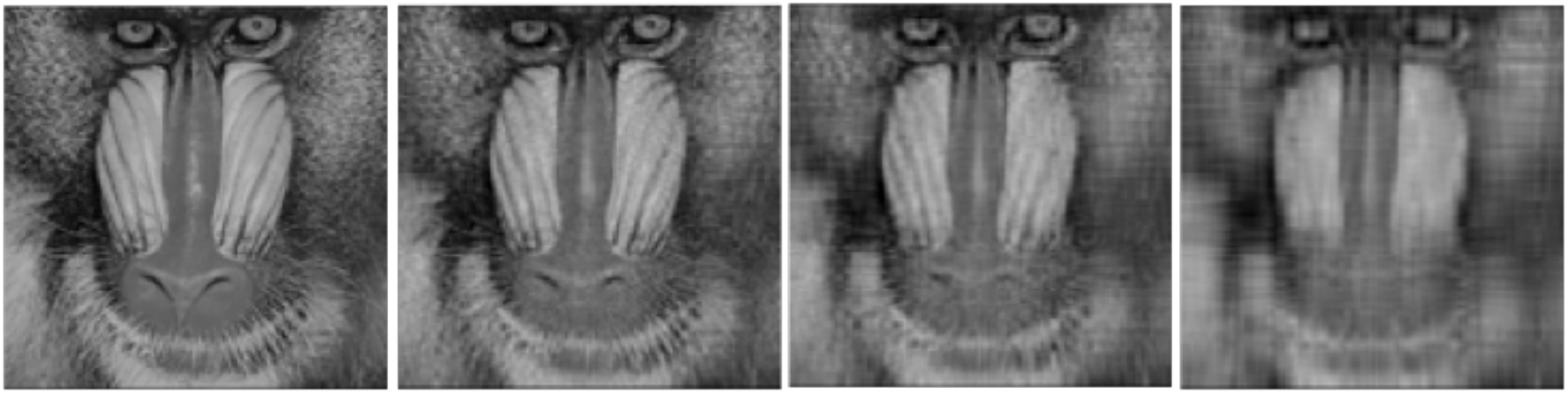
- Expressing the error in approximation as:

$$\varepsilon^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} | f(i,j) - \hat{f}(i,j) |^2$$

- It is possible to show:

$$\varepsilon^2 = \sum_{p=k+1}^{r} \lambda_p$$

Sum of singular values not used in approximation



Original and three SVD representations using 32, 16, and 8 basis vectors

SVD for Dimensionality Reduction using TruncatedSVD from sklearn

```
from scipy import linalg
import numpy as np
from sklearn.decomposition import TruncatedSVD
import matplotlib.pyplot as plt
np.set_printoptions(precision=2)
```

```
F = np.array([[6,6],[0,1],[4,0],[0,6]])
F = F.astype('Float64')
lsa = TruncatedSVD(1, algorithm = 'arpack')
Fa = lsa.fit_transform(F)
print(Fa)
```

```
[[8.4]
 [0.8]
 [2.4]
 [4.8]]
```

This is F in one dimension

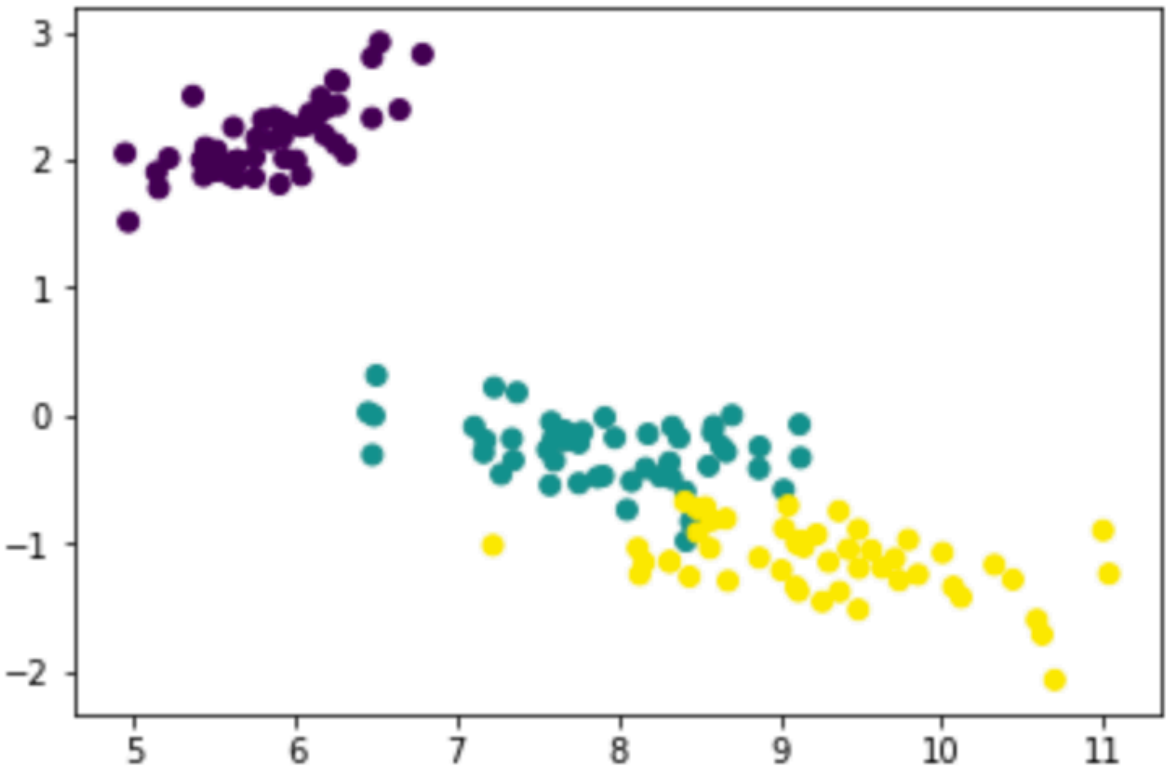
```
U,s,V = linalg.svd(F)
print(U)
print(s)
```

```
[[-0.84   0.24 -0.44 -0.2 ]
 [-0.08 -0.12   0.48 -0.87]
 [-0.24   0.64   0.66   0.3 ]
 [-0.48 -0.72   0.36   0.35]]
[10.   5.]
```

One-dimensional representation of F is simply the first column of U multiplied by the first singular value of F

```
from sklearn.datasets import load_iris
iris = load_iris()
iris_data = iris.data
iris_target = iris.target
lsa = TruncatedSVD(2, algorithm = 'arpack')
iris_transformed = lsa.fit_transform(iris_data)
```

```
plt.scatter(iris_transformed[:,0],iris_transformed[:,1], c = iris_target)
plt.show()
```

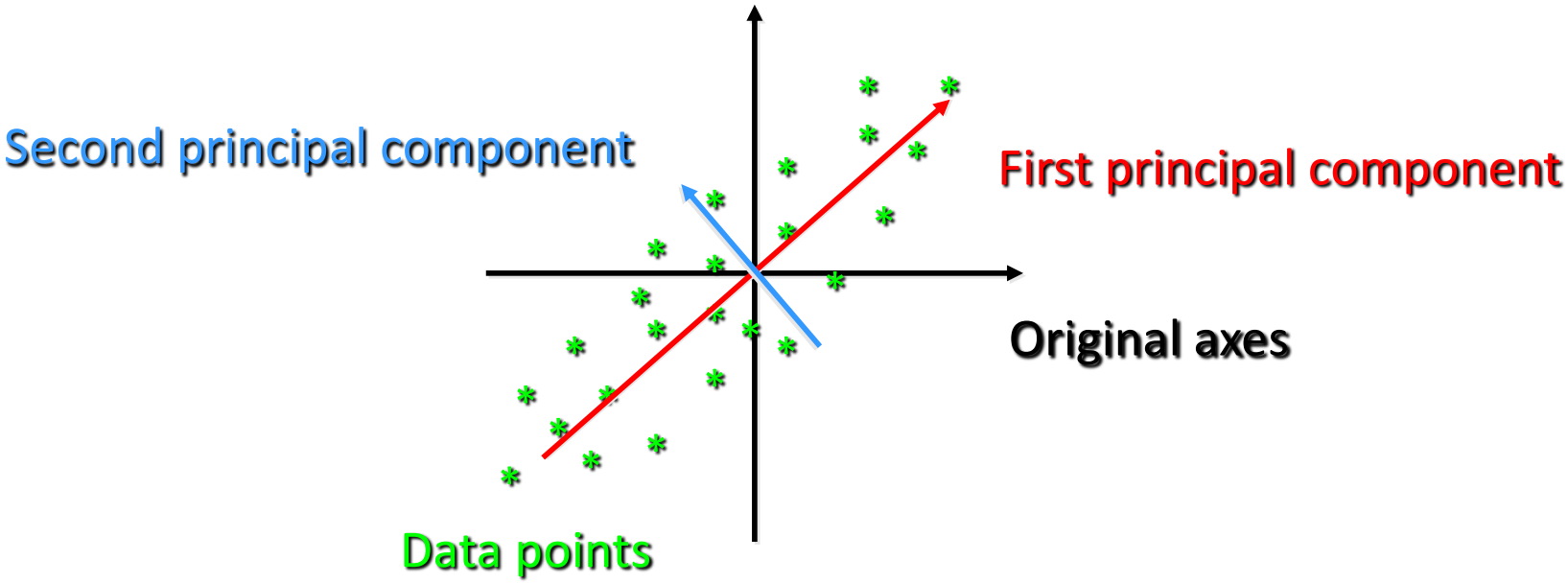


Dimensionality reduction of Iris data

This is similar to applying PCA for dimensionality reduction.

SVD and PCA

- Principal Components Analysis (PCA): approximating a high-dimensional data set with a lower-dimensional subspace

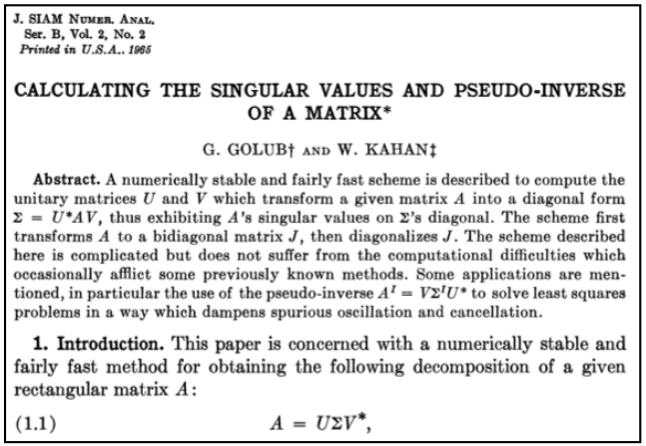


SVD and PCA

Both SVD and PCA are used for approximation (dimensionality reduction) by using the first *k* highest eigenvalues to obtain an approximation in the mean square sense. The difference is that the SVD approach is applicable to a single set of observations while the PCA method applies to sets of observations. Also the PCA can only be applied to a square matrix whereas SVD can be applied to any matrix.

Singular Value Decomposition

- The SVD was discovered independently by Beltrami(1873) and Jordan(1874) and again by Sylvester(1889).
- The SVD did not become widely known in applied mathematics until the late 1960s, when Golub and others showed that it could be computed effectively.



J. SIAM NUMER. ANAL.
Ser. B, Vol. 2, No. 2
Printed in U.S.A., 1965

CALCULATING THE SINGULAR VALUES AND PSEUDO-INVERSE OF A MATRIX*

G. GOLUB† AND W. KAHAN‡

Abstract. A numerically stable and fairly fast scheme is described to compute the unitary matrices U and V which transform a given matrix A into a diagonal form $\Sigma = U^*AV$, thus exhibiting A's singular values on Σ's diagonal. The scheme first transforms A to a bidiagonal matrix J, then diagonalizes J. The scheme described here is complicated but does not suffer from the computational difficulties which occasionally afflict some previously known methods. Some applications are mentioned, in particular the use of the pseudo-inverse $A^I = V\Sigma^I U^*$ to solve least squares problems in a way which dampens spurious oscillation and cancellation.

1. Introduction. This paper is concerned with a numerically stable and fairly fast method for obtaining the following decomposition of a given rectangular matrix A:

(1.1) $$A = U\Sigma V^*,$$

Cleve Moler (invented MATLAB, co-founded MathWorks)
Gene Golub has done more than anyone to make the singular value decomposition one of the most powerful and widely used tools in modern matrix computation.

In later years he drove a car with the license plate:

What is LSI?

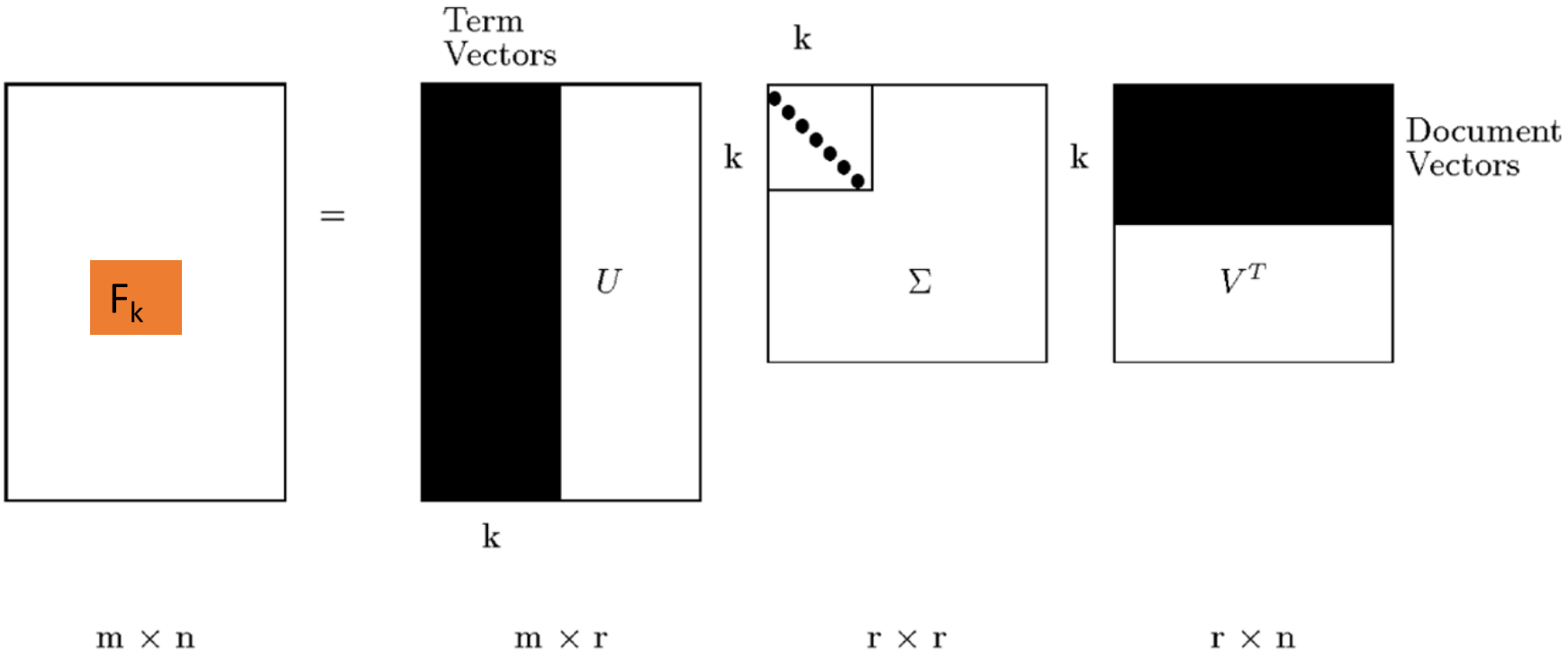
- LSI stands for latent semantic indexing. It was developed at Bellcore to improve information retrieval
- It uses SVD on term-document matrix to improve information retrieval by using hidden/latent similarities in words
- LSI also performs dimensionality reduction by working in a lower dimensional space
- LSI has found many applications including:
  - Image retrieval
  - Cross language information retrieval
  - Cross modality multimedia retrieval
  - Collaborative filtering

LSI Justification/Motivation

- Text corpus with many documents (docs)
- Given a query, find relevant docs
- Classical problems:
  - synonymy: For example, missing docs with reference to “automobile” when querying on “car”
  - polysemy: retrieving wrong docs, for example getting docs on internet when querying on “surfing”
- Solution: Represent docs (and queries) by their underlying latent concepts . How do we find such concepts? By utilizing context via LSI/SVD decomposition

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, Richard Harshman. 1990. Indexing by latent semantic analysis. JASIS 41(6):391—407.

LSI Representation



Term-Document matrix

LSI Example for IR

- We start with the term-document matrix for a given collection. As an example, consider the following term-document matrix

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

Example Contd.

- We next obtain the SVD decomposition of the term-document matrix. This will yield the following three matrices of the SD decomposition:

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
voyage	−0.70	0.35	0.15	−0.58	0.16
trip	−0.26	0.65	−0.41	0.58	−0.09

This is the U matrix; it is also known as the SVD term matrix

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

Matrix of singular values

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

V^t matrix, known as the SVD document matrix.

Example Contd.

- Next, we zero out all but the two largest singular values and write the zeroed singular matrix as:

By "zeroing out" all but the two largest singular values of Σ, we obtain $\Sigma_2 =$

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

- Using the zeroed singular matrix, we calculate approximations to the U and V matrices to obtain 2-dimensional representations of documents and terms.

Example Contd.

- The approximation to the V^t matrix is:

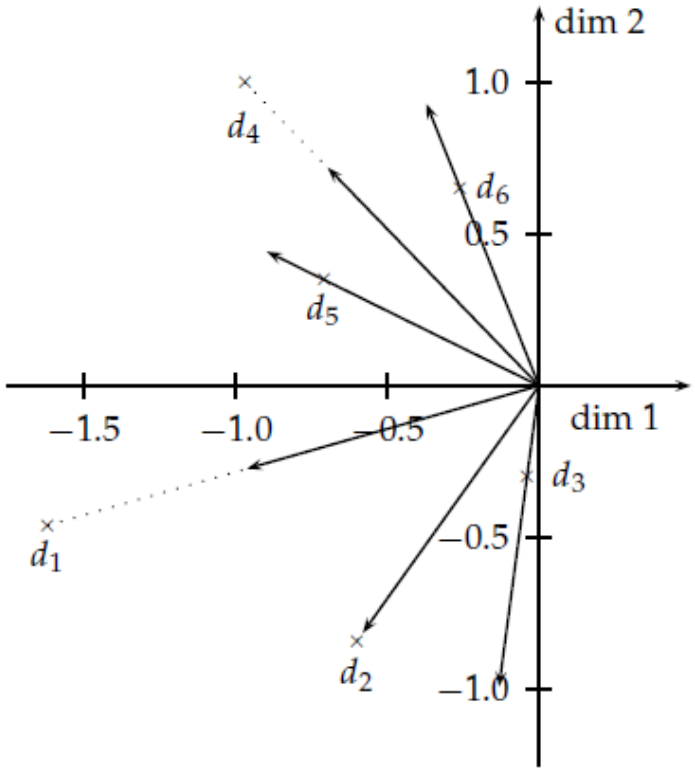
	d_1	d_2	d_3	d_4	d_5	d_6
1	−1.62	−0.60	−0.44	−0.97	−0.70	−0.26
2	−0.46	−0.84	−0.30	1.00	0.35	0.65

- The approximation to the U matrix is:

ship	-0.95	-0.48
boat	-0.28	-0.52
ocean	-1.04	-0.81
voyage	-1.51	0.56
trip	-0.56	1.03

Example Contd.

- Since we retained only two singular values, we can represent each document in a two-dimensional space. The same can be done for the terms (not shown)



Retrieval Using LSI

- To perform retrieval, the query (a collection of terms) is mapped into the lower dimensional space by using the following mapping:

$$q_k = q^T U_k \Sigma_k^{-1}$$

- The similarity of the mapped query vector is computed in the usual IR fashion to retrieve a set of matching documents.

Another example

We have a corpus of 17 documents. The terms used in creating the term-document matrix are underlined.

Label	Titles
B1	A Course on Integral Equations
B2	Attractors for Semigroups and Evolution Equations
B3	Automatic Differentiation of Algorithms: Theory, Implementation, and Application
B4	Geometrical Aspects of Partial Differential Equations
B5	Ideals, Varieties, and Algorithms – An Introduction to Computational Algebraic Geometry and Commutative Algebra
B6	Introduction to Hamiltonian Dynamical Systems and the N-Body Problem
B7	Knapsack Problems: Algorithms and Computer Implementations
B8	Methods of Solving Singular Systems of Ordinary Differential Equations
B9	Nonlinear Systems
B10	Ordinary Differential Equations
B11	Oscillation Theory for Neutral Differential Equations with Delay
B12	Oscillation Theory of Delay Differential Equations
B13	Pseudodifferential Operators and Nonlinear Partial Differential Equations
B14	Sinc Methods for Quadrature and Differential Equations
B15	Stability of Stochastic Differential Equations with Respect to Semi-Martingales
B16	The Boundary Integral Approach to Static and Dynamic Contact Problems
B17	The Double Mellin-Barnes Type Integrals and Their Applications to Convolution Theory

Example Contd.

TABLE 3

The 16 × 17 term-document matrix corresponding to the book titles in Table 2.

Terms	Documents																
	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
algorithms	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
application	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
delay	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
differential	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
equations	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
implementation	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
integral	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
introduction	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
methods	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
nonlinear	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
ordinary	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
oscillation	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
partial	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
problem	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
systems	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
theory	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1

Representation of documents and the terms in the two-dimensional space after mapping

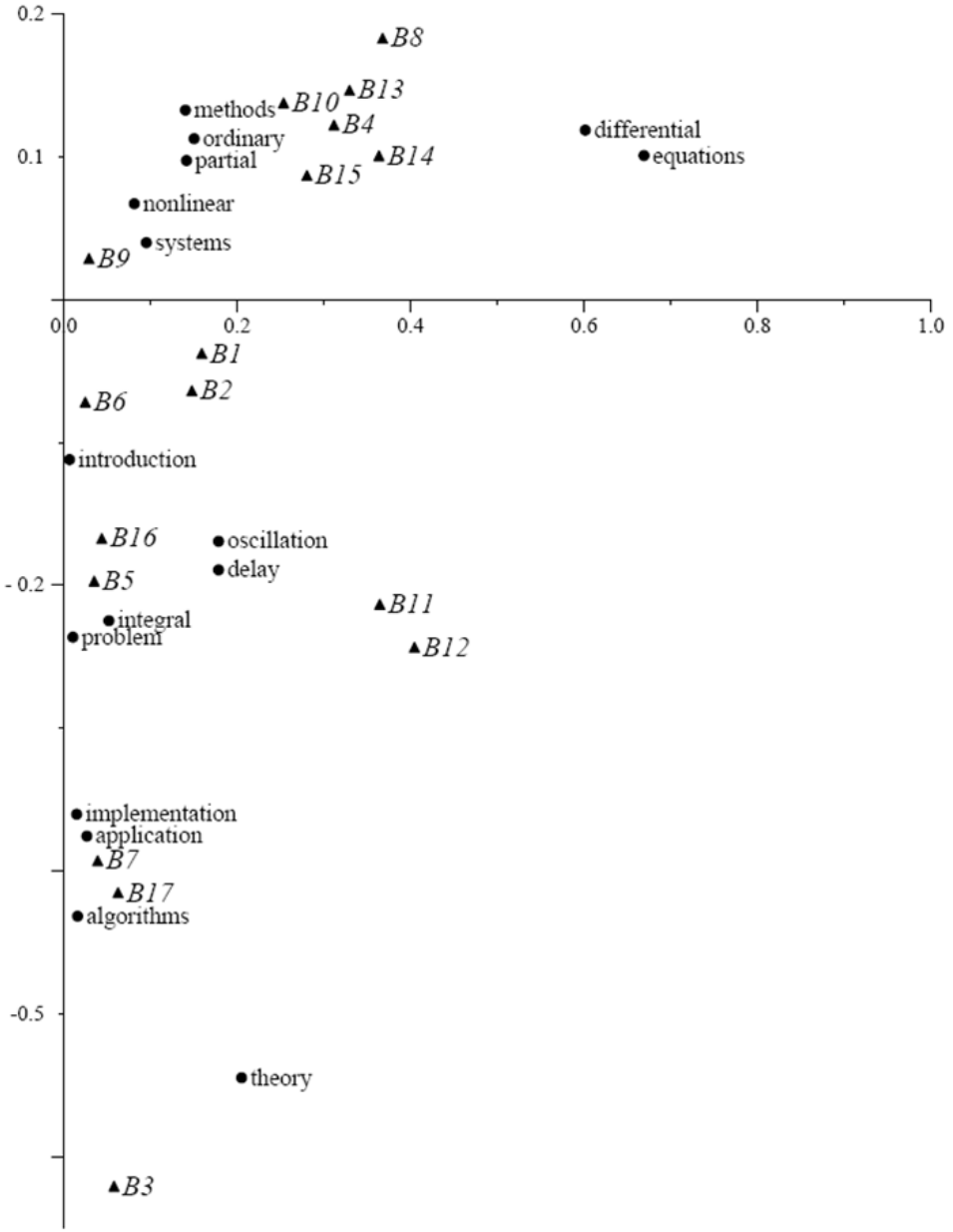


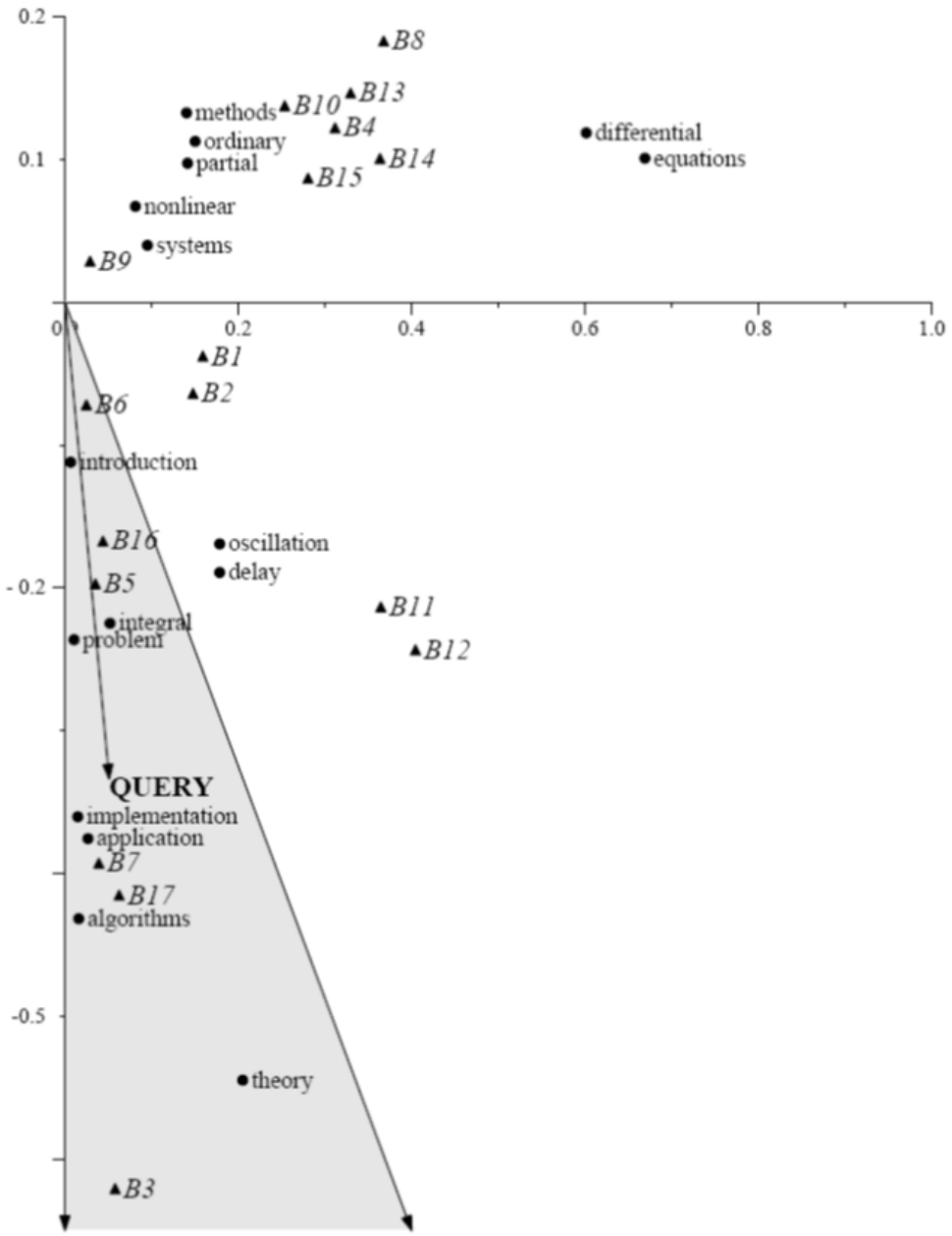
FIG. 4. *Two-dimensional plot of terms and documents for the* 16 × 17 *example.*

Let the query terms be: Application , Theory
Then query vector is:

$$\begin{pmatrix} 0.0511 & -0.3337 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}^T \begin{pmatrix} 0.0159 & -0.4317 \\ 0.0266 & -0.3756 \\ 0.1785 & -0.1692 \\ 0.6014 & 0.1187 \\ 0.6691 & 0.1209 \\ 0.0148 & -0.3603 \\ 0.0520 & -0.2248 \\ 0.0066 & -0.1120 \\ 0.1503 & 0.1127 \\ 0.0813 & 0.0672 \\ 0.1503 & 0.1127 \\ 0.1785 & -0.1692 \\ 0.1415 & 0.0974 \\ 0.0105 & -0.2363 \\ 0.0952 & 0.0399 \\ 0.2051 & -0.5448 \end{pmatrix} \begin{pmatrix} 4.5314 & 0 \\ 0 & 2.7582 \end{pmatrix}^{-1}$$

Mapped query vector

$$q_k = q^T U_k \Sigma_k^{-1}$$


0.2
0.1
- 0.2
-0.5
0.0
0.2
0.4
0.6
0.8
1.0
B8
B13
B10
B4
B14
B15
B1
B2
B6
B16
B5
B11
B12
B7
B17
B9
B3
methods
ordinary
partial
nonlinear
systems
differential
equations
introduction
oscillation
delay
integral
problem
QUERY
implementation
application
algorithms
theory

Updating in LSI

- Updating might require adding a new document or a new term. Recalculating (recomposing) is expensive. Instead new documents/terms are *folded in* by the following equations. The preexisting documents and terms are not affected.

To fold-in a new $m \times 1$ document vector, d, into an existing LSI model, a projection, $\hat{d}$, of d onto the span of the current term vectors (columns of U_k) is computed by

$$\hat{d} = d^T U_k \Sigma_k^{-1}. \tag{7}$$

Similarly, to fold-in a new $1 \times n$ term vector, t, into an existing LSI model, a projection, $\hat{t}$, of t onto the span of the current document vectors (columns of V_k) is determined by

$$\hat{t} = t V_k \Sigma_k^{-1}. \tag{8}$$

Illustration of Folding in and Re-composition

Label	Titles
B18	Systems of Nonlinear Equations
B19	Ordinary Algorithms for Integral and Differential Equations
B20	Ordinary Applications of Oscillation Theory

Illustration of Folding in

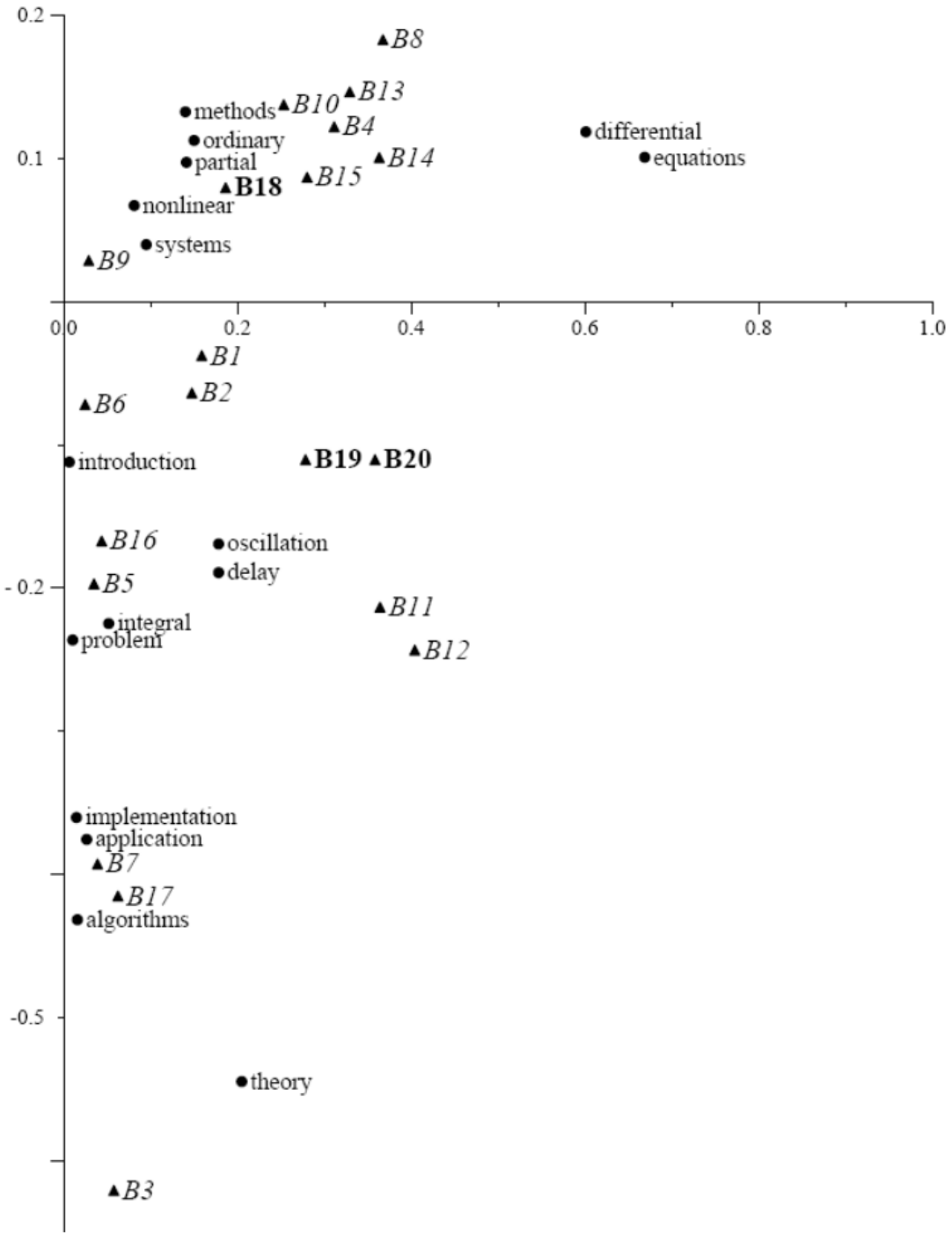
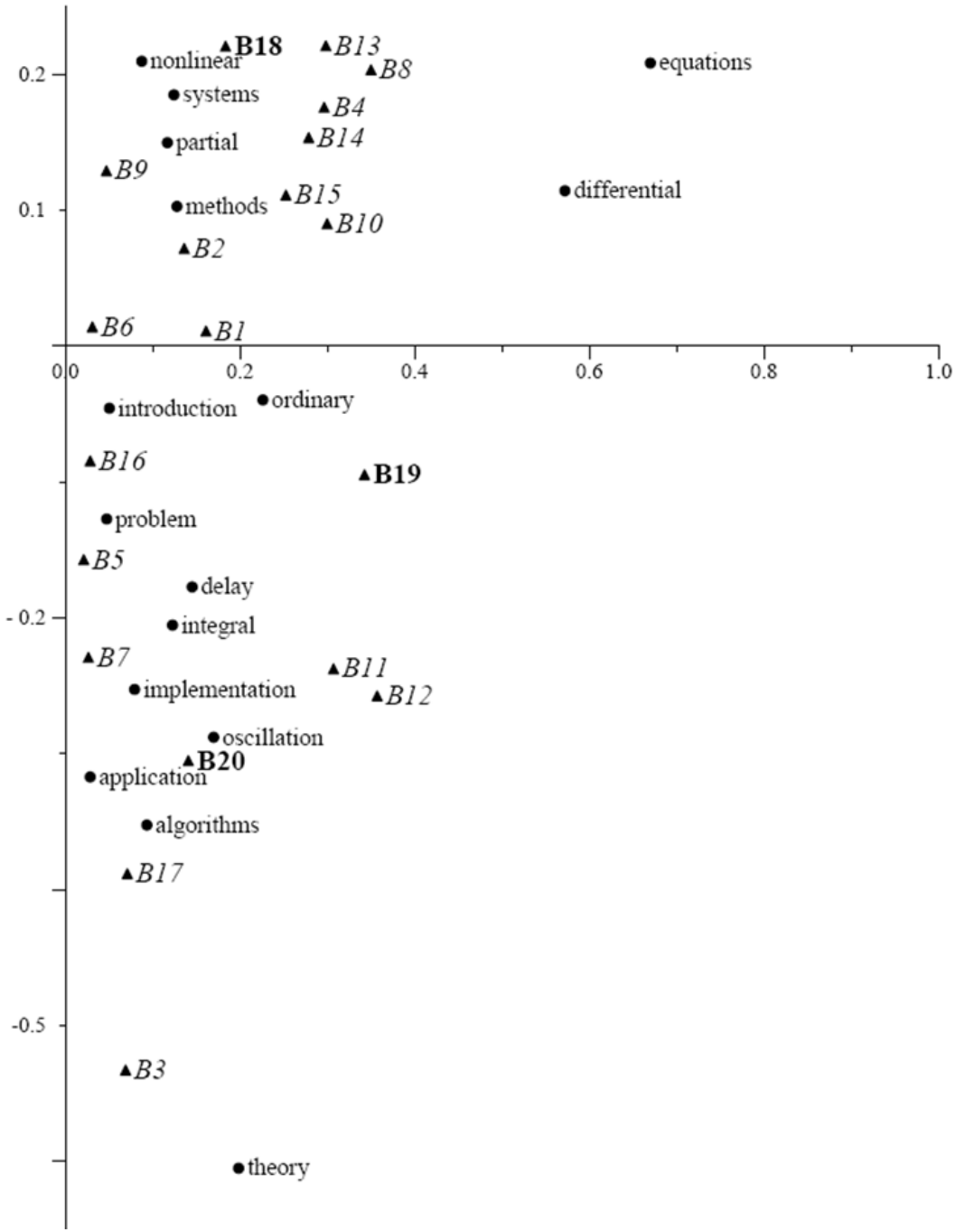


Illustration of Recomposing



How Useful is LSI?

- LSI is useful only if k << 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.
- 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
  - Typical result of LSI is improved recall at lower precision
- SVD is computationally expensive; thus LSI has limited use for large document collections
- Inverted index not possible

Latent Semantic Analysis (LSA)

- The term LSI is used when document indexing/retrieval is involved. The term LSA is used for other applications of the SVD-based decomposition approach.

Measuring Word Similarities

Word Similarity

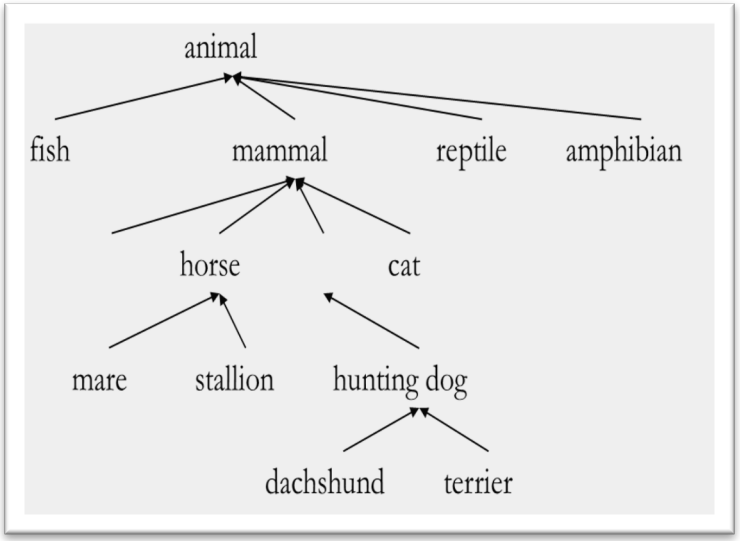
- Words can be similar if:
  - They mean the same thing (synonyms)
  - They mean the opposite (antonyms)
  - They are used in the same way (red, green)
  - They are used in the same context (doctor, hospital, scalpel)
  - One is a type of another (poodle, dog, mammal)
- Word similarity methods
  - Using a Lexical Database such as WordNet
  - Corpus Based Methods
    - Latent Semantic Analysis (LSA)
    - Explicit Semantic Analysis (Not covered)
  - Knowledge Based Methods
    - Google distance

WordNet

- WordNet, developed at Princeton, is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.
- Synsets are interlinked by means of conceptual-semantic and lexical relations.
- WordNet interlinks not just word forms—strings of letters—but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated.
- WordNet labels the semantic relations among words

WordNet Based Similarity

- WordNet organizes nouns and verbs into hierarchies of *is–a* relations.
- There are nine separate noun hierarchies that include 80,000 concepts, and 554 verb hierarchies that are made up of 13,500 concepts
- *Is–a* relations in WordNet do not cross part of speech boundaries, so similarity measures are limited to making judgments between noun pairs (e.g., *cat* and *dog*) and verb pairs (e.g., *run* and *walk*). Similarity is measured using variations of the path length between the two concepts
- While WordNet also includes adjectives and adverbs, these are not organized into *is–a* hierarchies so similarity measures can not be applied.



Using measures of semantic relatedness for word sense disambiguation, S Patwardhan, S Banerjee, T Pedersen
Computational linguistics and intelligent text processing, 241-257

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: surfing Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **surfing**, surfboarding, surfriding (the sport of riding a surfboard toward the shore on the crest of a wave)

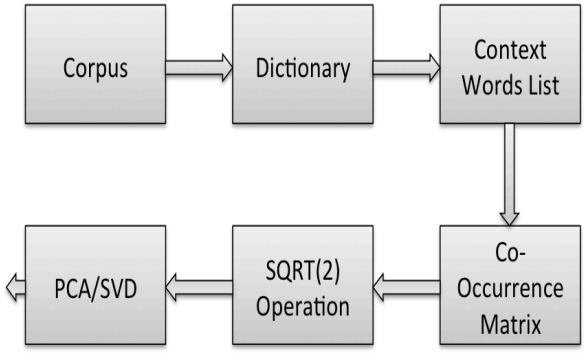
Verb

- S: (v) surfboard, surf (ride the waves of the sea with a surfboard) *"Californians love to surf"*
- S: (v) browse, surf (look around casually and randomly, without seeking anything in particular) *"browse a computer directory"; "surf the internet or the world wide web"*
- S: (v) surf, channel-surf (switch channels, on television)

Latent Semantic Analysis

- Finds words that co-occur within a window of a few words and forms an NxN cooccurrence matrix.
- Perform SVD to get a k-dimensional, k<N, representation for word vectors
- This technique learns related words due to their occurrence together in a context.
- Problem: Mapped dimensions are not well defined.

Words as Vectors using Co-occurrences of Words



https://iksinc.online/2015/06/23/how-to-use-words-co-occurrence-statistics-to-map-words-to-vectors/

Make sure to go over a small example given in the Comments section.

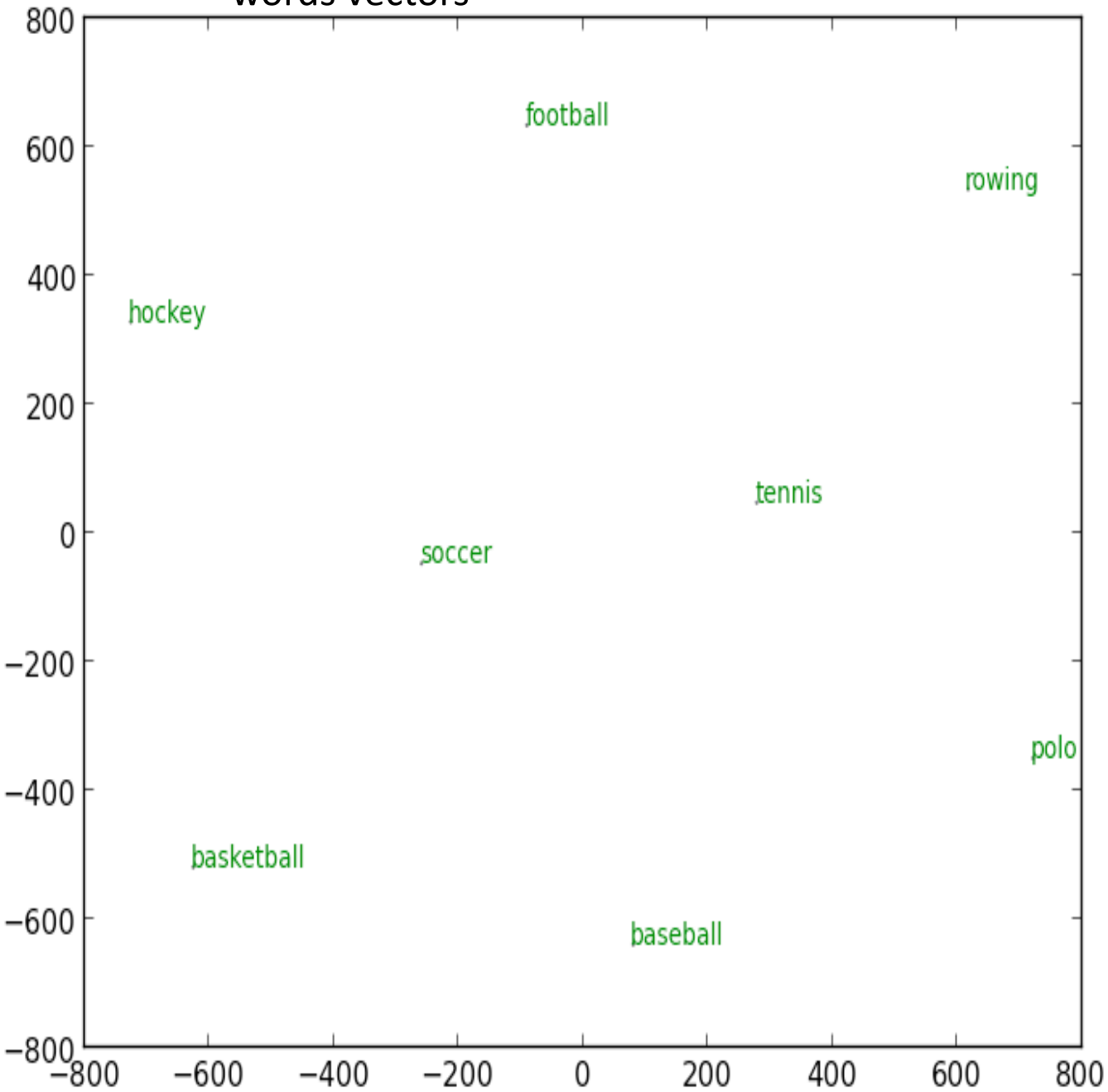
Hellinger Distance

Hellinger distance is a measure of similarity between two probability distributions. Given two discrete probability distributions $P = (p_1, \ldots, p_k)$ and $Q = (q_1, \ldots, q_k)$, the Hellinger distance $H(P, Q)$ between the distributions is defined as:

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})^2}$$

Hellinger distance is a metric satisfying triangle inequality. The reason for including $\sqrt{(2)}$ in the definition of Hellinger distance is to ensure that the distance value is always between 0 and 1. When comparing a pair of discrete probability distributions the Hellinger distance is preferred because P and Q are vectors of unit length as per Hellinger scale.

Mapped 2-dimensional similarity of words vectors



Words as Vectors

- Some recent methods use a large amount of text to create high-dimensional (50 to 300 dimensional) representations of words capturing relationships between words unaided by external annotations.
- Such a representation seems to capture many linguistic regularities. For example, it yields a vector approximating the representation for *vec('Rome')* as a result of the vector operation *vec('Paris') – vec('France') + vec('Italy')*

For a simple intro, see the blog at https://iksinc.online/2015/04/13/words-as-vectors/

Normalized Google Distance

- It is a similarity measure that takes advantage of Google search
- The basic idea is that if two words/phrases occur on a same web page many times, then the words bear some similarity or relationship
- The NGD measure is defined as:

$$\mathrm{NGD}(x,y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log N - \min\{\log f(x), \log f(y)\}}$$

N : Total number of web pages searched multiplied by the average number of words per page

f(x) : Number of web pages with word *x*

f(x, y) : Number of web pages with both *x* and *y*

F(y) : Number of web pages with word *y*

Properties of NGD

- *NGD(x,x) = 0*
- *NGD(x,y) = NGD(y,x)*
- NGD lies between 0 and ∞
- NGD is not a metric; it doesn't satisfy the triangle inequality

Illustration of NGD Use

- Binary classification problem
- Two sets of words describing two classes of situations: *emergencies (Positive class)* and *almost emergencies* are used for training
- A six-dimensional feature vector is generated for each word using its NGD from six anchor words.

Training Data

Positive Training	(22 cases)			
avalanche	bomb threat	broken leg	burglary	car collision
death threat	fire	flood	gas leak	heart attack
hurricane	landslide	murder	overdose	pneumonia
rape	roof collapse	sinking ship	stroke	tornado
train wreck	trapped miners			
Negative Training	(25 cases)			
arthritis	broken dishwasher	broken toe	cat in tree	contempt of court
dandruff	delayed train	dizziness	drunkenness	enumeration
flat tire	frog	headache	leaky faucet	littering
missing dog	paper cut	practical joke	rain	roof leak
sore throat	sunset	truancy	vagrancy	vulgarity
Anchors	(6 dimensions)			
crime	happy	help	safe	urgent
wash				

NGD Use Illustration

- SVM classifier used for training

Testing Results

	Positive tests	Negative tests
Positive Predictions	assault, coma, electrocution, heat stroke, homicide, looting, meningitis, robbery, suicide	menopause, prank call, pregnancy, traffic jam
Negative Predictions	sprained ankle	acne, annoying sister, campfire, desk, mayday, meal
Accuracy	15/20 = 75.00%	

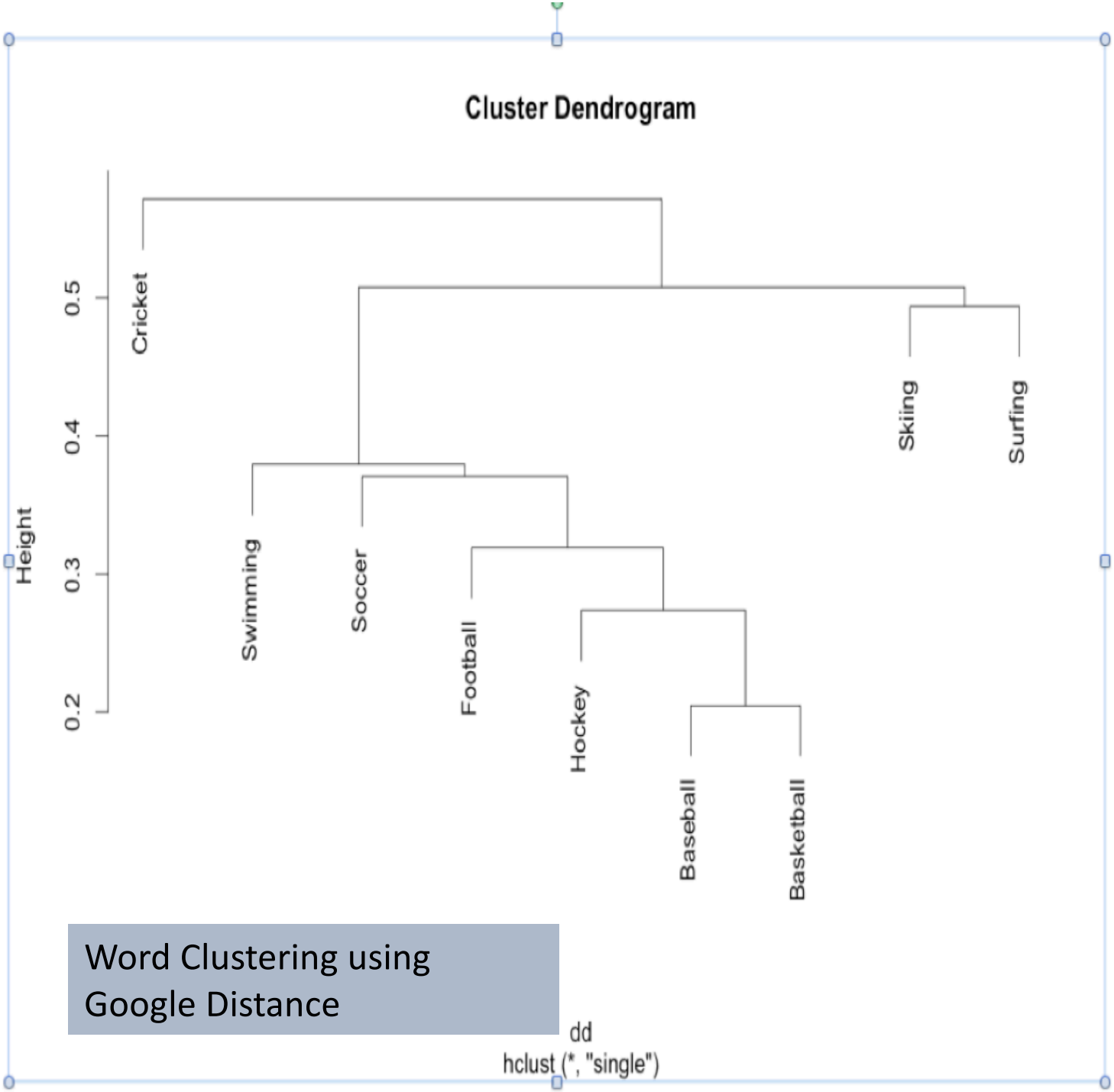
Hierarchical Clustering using NGD

- Given a set of words, this example uses NGD to construct an inter-word distance matrix
- Hierarchical clustering is performed using the NGD distance matrix
- Distance matrix and clusters shown on the following slides

Word Similarity, Google Distance, and Baseball and Cricket

Another Google Word Distance Example

	Baseball	Cricket	Soccer	Football	Basketball	Hockey	Skiing	Surfing	Swimming
Baseball	0.00	0.57	0.40	0.37	0.20	0.27	0.51	0.92	0.56
Cricket	0.57	0.00	1.75	1.24	1.63	1.15	1.26	0.87	1.47
Soccer	0.40	1.75	0.00	0.39	0.37	0.49	0.82	1.07	0.50
Football	0.37	1.24	0.39	0.00	0.32	0.42	0.62	0.70	0.38
Basketball	0.20	1.63	0.37	0.32	0.00	0.38	0.72	1.02	0.46
Hockey	0.27	1.15	0.49	0.42	0.38	0.00	0.71	0.98	0.85
Skiing	0.51	1.26	0.82	0.62	0.72	0.71	0.00	0.49	0.61
Surfing	0.92	0.87	1.07	0.70	1.02	0.98	0.49	0.00	0.68
Swimming	0.56	1.47	0.50	0.38	0.46	0.85	0.61	0.68	0.00

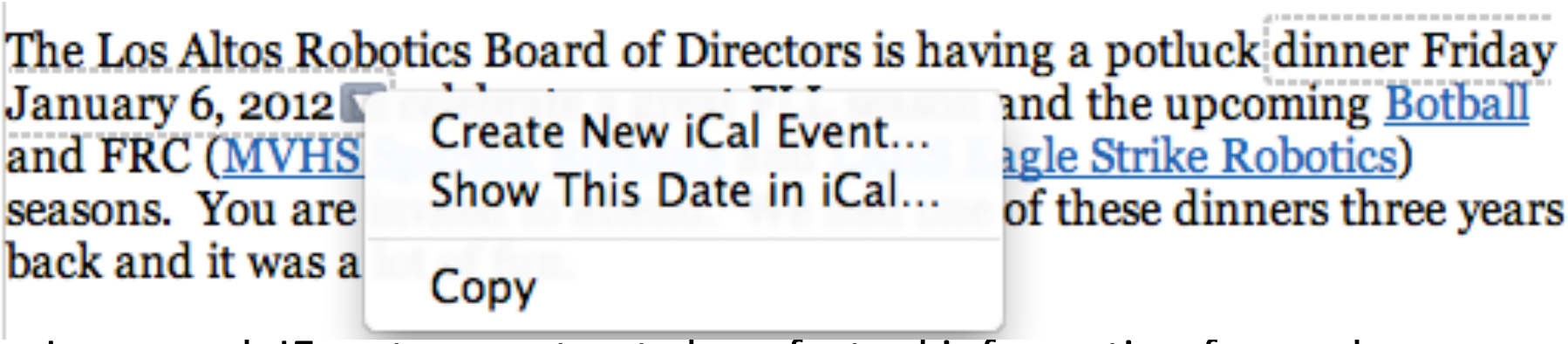


Word Clustering using Google Distance

Information Extraction

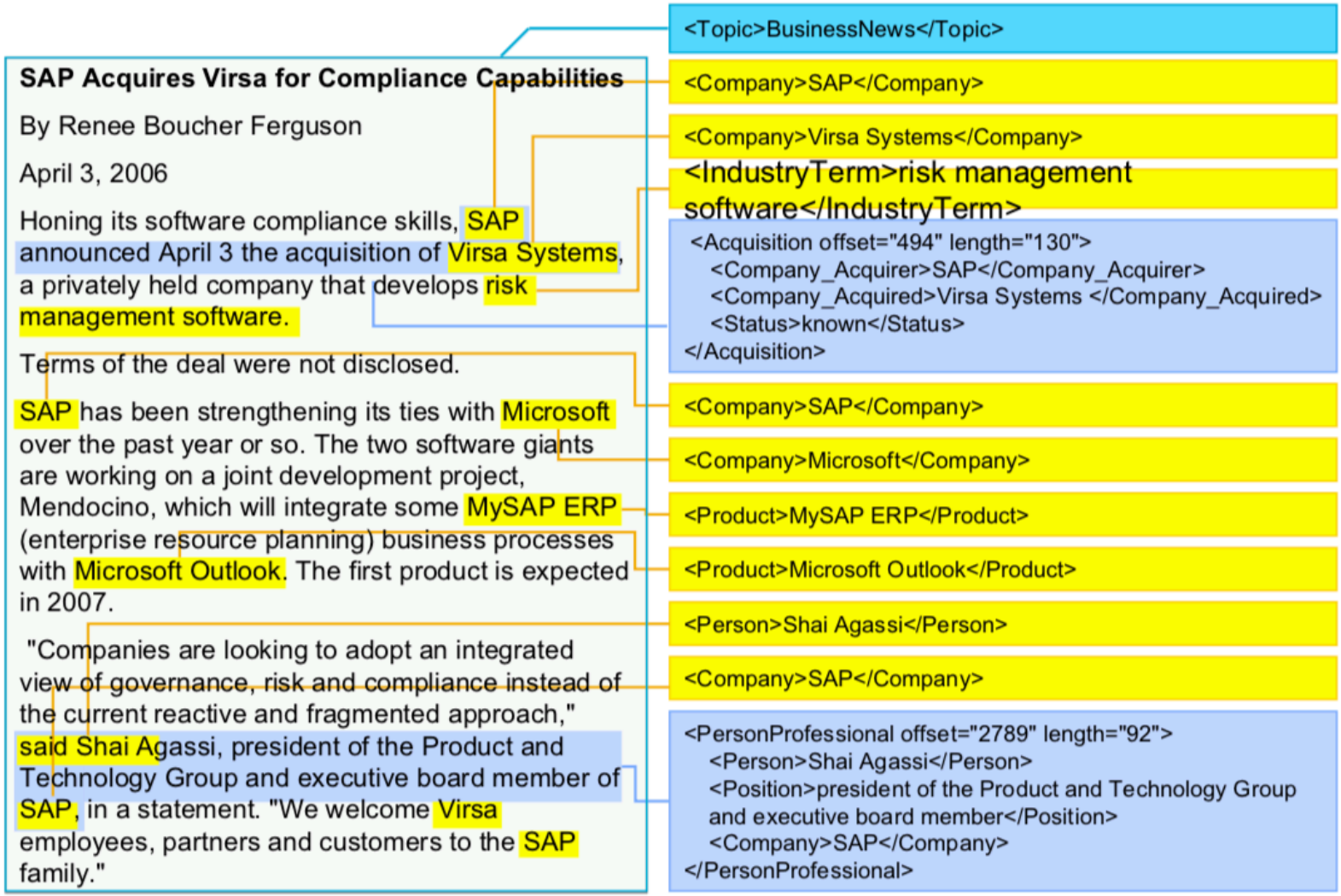
Information Extraction (IE)

- Information extraction is the process of extracting specific (pre-specified) information from textual sources.
  - A simple example is when your email system extracts relevant information from a message for you to add in your Calendar.



- In general, IE systems extract clear, factual information from a large collection of documents to build a DB to provide answers for queries of the type
  - Roughly: *Who did what to whom when?*

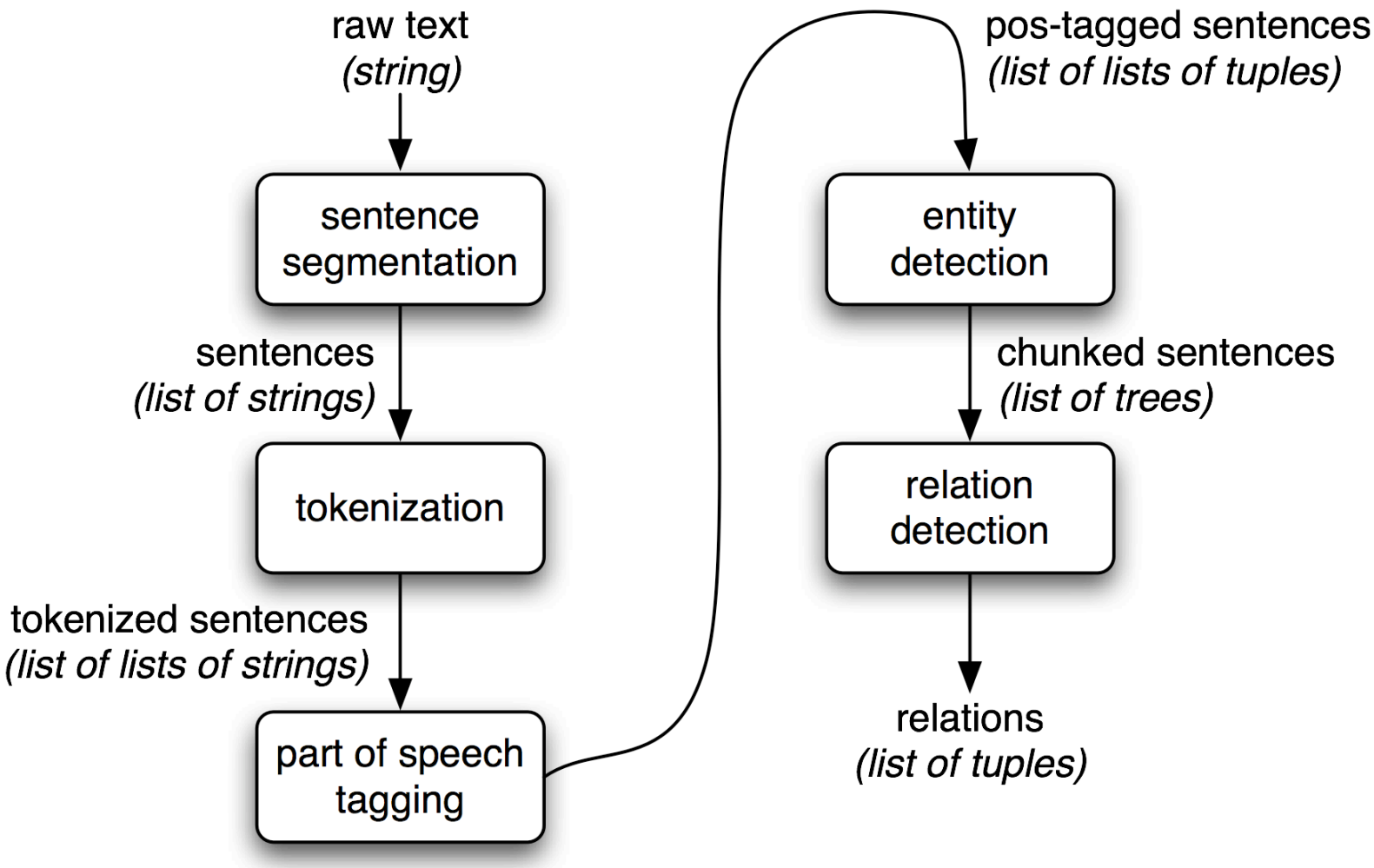
Information Extraction Example



Why IE

- Gathering detailed structured data from texts, information extraction enables:
  - The automation of tasks such as smart content classification, integrated search, management and delivery;
  - Data-driven activities such as mining for patterns and trends, uncovering hidden relationships, etc.
- Typical Information Extraction Applications
  - **Business intelligence** (for enabling analysts to gather structured information from multiple sources);
  - **Financial investigation** (for analysis and discovery of hidden relationships);
  - **Scientific research** (for automated references discovery or relevant papers suggestion);
  - **Media monitoring** (for mentions of companies, brands, people);
  - **Healthcare records management** (for structuring and summarizing patients records);
  - **Pharma research** (for drug discovery, adverse effects discovery and clinical trials automated analysis).

Information Extraction Steps



POS Tagging

- Parts-of-Speech labels such as noun, verb, adjective, preposition, etc. are assigned to tokens
- Part-of-Speech tagging approaches can generally fall into two categories: Rule based approaches and statistical approaches.
- Rule based approaches apply language rules to improve the accuracy of tagging. The limitation of this approach lies in requirement of large annotated data which require expert lin- guistic knowledge, labor and cost.
- Machine learning based approaches wherein a trained classifier is used to perform POS tagging

Named Entity Recognition (NER)

- The task here is to recognize those tokens that represent persons, organizations, locations and geo-political entities.
- Two basic approaches to NER:
  - Rule-based approach using Regular Expressions
    - Uses hand-coded rules
    - Domain dependent
    - Expensive
    - Changes over time cause difficulties
  - Machine learning approach
    - Inexpensive
    - Large training data
    - Cheap annotation (Mechanical Turk)
    - The features for a word are typically designed to reflect the local context of the word. Examples of local context are neighboring k words, appearing before and after and their respective part-of-speech tags. It is the choice of the features that determines the accuracy of the NER.

An Example of Rule-Based NER

Hillary Clinton tore into Donald Trump's tax maneuvering, business skills and trustworthiness Monday as she sought to capitalize on news that the New York real estate mogul may have paid no federal taxes for years.

```
> mytext<-scan("~/Desktop/Hillary.txt", character(0))
Read 35 items
> grep("^[A-Z]",mytext,perl=TRUE,value=TRUE)
[1] "Hillary" "Clinton" "Donald"  "Trump's" "Monday"  "New"     "York"
```

Named Entity Recognition with ANNIE

GATE is an open source infrastructure for developing and deploying software components that process human language. GATE excels at text analysis of all shapes and sizes. From large corporations to small startups, from multi-million research consortia to undergraduate projects. More than €5 million has been invested in GATE development and our objective is to make sure that this continues to be money well spent for all GATE's users.

GATE is distributed with an example Information Extraction system, known as ANNIE, which has formed the basis of many commercial and research systems. While ANNIE is capable of recognising a number of different entity types this simple demo focuses on the annotion of people, locations, and organizations.

To try the demo please enter some free text to process:

```
The New England Patriots have identified the fan who threw beer in the face of Kansas City Chiefs wide receiver Tyreek Hill during Sunday's night's game. The fan has been banned from Gillette Stadium and the Patriots have turned the matter over to law enforcement.
```

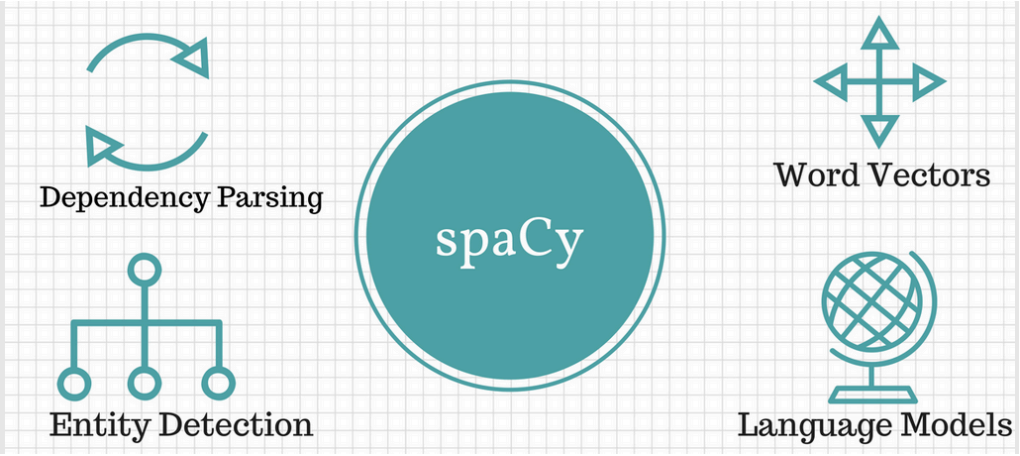
↓ Process Text ↓

The New England Patriots have identified the fan who threw beer in the face of Kansas City Chiefs wide receiver Tyreek Hill during Sundays nights game. The fan has been banned from Gillette Stadium and the Patriots have turned the matter over to law enforcement.

Please note that ANNIE was initially developed to process English language documents, mostly American news articles, and as such would require tuning to other languages, locales, or domains.

spaCy for NLP in Python

- The information extraction steps shown earlier can be easily carried out using spaCy, a python library for NLP. Another library for the same purpose is the Natural Language Toolkit library, NTLK.
- spaCy also has built-in word vectors. This means you can get better results for various text processing tasks.
- It comes with built-in language models for many languages.



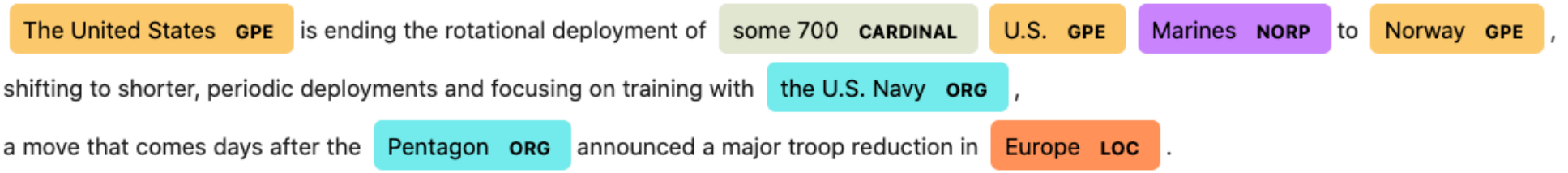
Tokenization with spaCy

```
# Word tokenization
from spacy.lang.en import English

# Load English tokenizer, tagger, parser, NER and word vectors
nlp = English()

text = """The United States is ending the rotational deployment of some 700 U.S. Marines to Norway,
shifting to shorter, periodic deployments and focusing on training with the U.S. Navy,
a move that comes days after the Pentagon announced a major troop reduction in Europe."""

#  "nlp" Object is used to create documents with linguistic annotations.
my_doc = nlp(text)

# Create list of word tokens
token_list = []
for token in my_doc:
    token_list.append(token.text)
print(token_list)
```

```
['The', 'United', 'States', 'is', 'ending', 'the', 'rotational', 'deployment', 'of', 'some', '700', 'U.S.', 'Marines', 'to',
'Norway', ',', '\n', 'shifting', 'to', 'shorter', ',', 'periodic', 'deployments', 'and', 'focusing', 'on', 'training', 'with
', 'the', 'U.S.', 'Navy', ',', '\n', 'a', 'move', 'that', 'comes', 'days', 'after', 'the', 'Pentagon', 'announced', 'a', 'maj
or', 'troop', 'reduction', 'in', 'Europe', '.']
```

Named Entity Recognition with spaCy

```
# Named Entity Recognition
import en_core_web_sm
from spacy import displacy
nlp = en_core_web_sm.load()
doc = nlp(text)
displacy.render(doc, style="ent")
#for ent in doc.ents:
    #print(ent.text, ent.label_)
```



You can train your own NER if you are working in a special domain.

PoS Tagging in spaCy

```
doc = nlp("Tesla reports fifth consecutive quarter of profits")
for token in doc:
    print(token.text,token.pos_)
```

```
Tesla PROPN
reports VERB
fifth ADJ
consecutive ADJ
quarter NOUN
of ADP
profits NOUN
```

Word Vectors in spaCy

- The spaCy library comes with built-in pre-trained word vectors. You will need to load the large language model for best results. Here is an example of the word vector for ‘pet’. It is a 300-dimensional vector.

```
pet.vector
```

```
array([-3.2021e-01,  2.3677e-01, -1.8965e-01, -4.3808e-01,  3.2134e-01,
        2.2599e-01, -2.9738e-01, -3.3284e-01,  1.3276e-02,  1.8255e+00,
       -6.8326e-01, -2.1144e-01, -5.1686e-01,  2.1188e-01, -3.0855e-01,
       -1.8966e-01,  5.9407e-01,  1.3767e+00, -3.7601e-02, -1.1015e-02,
       -5.1636e-01,  4.4497e-01,  1.1237e-01, -2.0388e-01,  4.3725e-01,
        4.3707e-02, -4.2008e-01, -4.1261e-01,  1.2107e-01,  2.3464e-02,
       -4.4328e-01, -5.1614e-03, -3.0632e-02,  5.9136e-01,  1.1151e-01,
        2.5462e-01,  4.6570e-01, -5.2234e-01, -2.0479e-02,  1.4698e-01,
        2.7550e-01,  1.3786e-02, -1.6621e-01, -6.2409e-01, -8.9611e-02,
       -5.0172e-02, -6.6342e-01,  1.2410e-01,  1.0190e-01, -1.6332e-01,
       -2.8740e-01, -4.3852e-01,  6.1361e-01,  6.3687e-01, -5.3210e-01,
        1.8384e-01,  1.6551e-01,  2.6447e-01, -2.5032e-01,  3.5439e-01,
        4.7892e-01, -1.6863e-01, -1.7733e-01, -2.1600e-01,  3.8453e-01,
        2.9101e-02,  4.5810e-01, -8.9102e-02, -2.3719e-01,  2.2335e-02,
       -5.4510e-01, -1.4196e-01, -6.0862e-01, -6.7093e-01,  3.8037e-01,
        4.5201e-01, -1.6732e-02, -2.6140e-01,  3.2731e-01, -3.1379e-01,
       -1.8549e-01,  6.5261e-01, -5.1746e-01, -3.2792e-01, -5.2962e-01,
       -5.3587e-01,  7.6511e-01,  1.2306e+00,  8.7570e-01, -3.9827e-01,
        2.5127e-01,  5.5742e-01, -4.9160e-02, -4.3051e-01, -5.8603e-01,
       -1.1521e-01, -1.8876e-01, -3.8349e-01, -4.4211e-03, -2.8757e-02,
        5.1534e-02,  4.5303e-01,  2.0841e-01, -1.7852e-02,  2.3401e-01,
       -8.4980e-01,  1.9855e-01, -3.6685e-01, -7.5310e-01, -2.1640e-01,
        2.7641e-01, -3.5316e-01,  6.3785e-01, -1.1467e-01,  2.2812e-02,
       -1.4970e-01,  2.8123e-01,  5.5078e-01, -2.6871e-01,  4.4300e-01,
        3.4717e-01,  7.8961e-02, -6.5047e-01, -1.2673e-01, -6.5096e-02,
       -2.9423e-01, -1.2397e-01, -2.4408e-01,  3.0055e-01,  7.3845e-03,
        7.2950e-01, -8.0815e-02,  1.8033e-01,  4.3336e-01,  7.3240e-02],
      dtype=float32)
```

Measuring Similarity using Word Vectors

```
nlp = spacy.load('en_core_web_lg')
cat = nlp('cat')
dog = nlp('dog')
cat.similarity(dog)
```

```
0.8016855517329495
```

```
pet = nlp('pet')
```

```
pet.similarity(cat)
```

```
0.7505455881628591
```

```
pet.similarity(dog)
```

```
0.8057452229094295
```

You can note that word vector representation of words captures semantics much better.

Sentiment Analysis

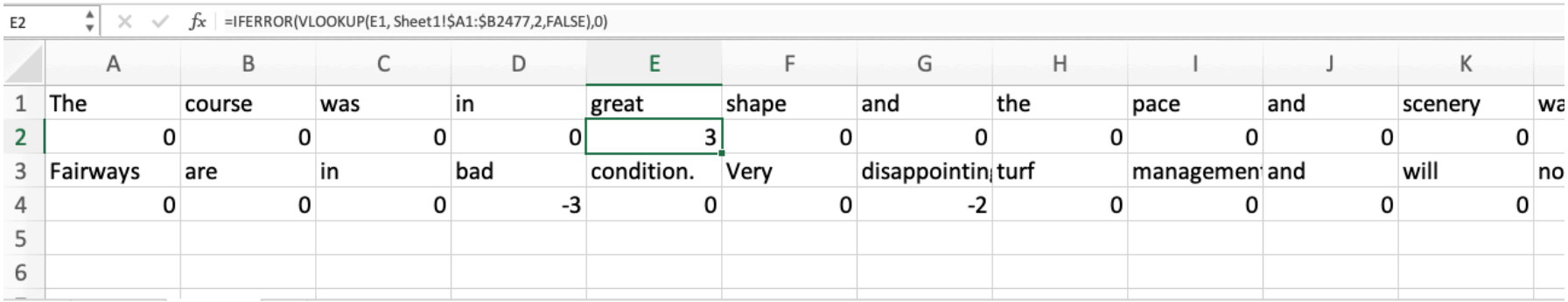
- Also known with many other names
  - Opinion extraction
  - Opinion mining
  - Sentiment mining
  - Subjectivity analysis

- Why sentiment analysis
  - Helps companies/entities to respond in a timely manner
  - Helps in prediction, e.g. elections, stock market

Sentiment Analysis Approaches

- Simplest sentiment analysis
  - Use a list of positive and negative words to count their occurrences in the given text and compute overall polarity score to mark the given text as showing positive or negative sentiment
  - A variation is to provide different weights to the words in the list of positive and negative words
- Sentiment analysis as a classification task
  - Collect lots of labeled text
  - Design a set of features and use NB or any other classifier

Example of a Very Simple Sentiment Analyzer



E2 =IFERROR(VLOOKUP(E1, Sheet1!$A1:$B2477,2,FALSE),0)

	A	B	C	D	E	F	G	H	I	J	K	
1	The	course	was	in	great	shape	and	the	pace	and	scenery	wa
2	0	0	0	0	3	0	0	0	0	0	0	
3	Fairways	are	in	bad	condition.	Very	disappointin	turf	managemen	and	will	no
4	0	0	0	-3	0	0	-2	0	0	0	0	
5												
6												

The list is from here:
http://www2.imm.dtu.dk/pubdb/pubs/6010-full.html

The best results are obtained by using word vector (word embeddings) representation.

Topic Modeling & Document Clustering

- When you have a huge collection of documents you need a method to organize the collection. It turns out that you can do so by topic modeling or by clustering.
- In topic modeling, a topic is defined by a cluster of words with each word in the cluster having a probability of occurrence for the given topic, and different topics have their respective clusters of words along with corresponding probabilities. Different topics may share some words and a document can have more than one topic associated with it.
- In clustering, the basic idea is to group documents into different groups based on some suitable similarity measure. The result of clustering is a list of clusters with every document showing up in one of the clusters

https://iksinc.online/2016/05/16/topic-modeling-and-document-clustering-whats-the-difference/

The performance analysis of a Chi-square similarity measure for topic related clustering of noisy transcripts

Summary

- Numerous applications of text mining. Some examples are:
  - Building knowledge-bases for different domains such as recording and tracking of adverse drug reactions
  - Monitoring social media posts for various purposes such as reputation management, suicide prevention etc.

- Heavy use of NLP for some of the text mining tasks is needed