Text Analytics (Text Mining)

Duen Horng (Polo) Chau
Associate Professor, College of Computing
Associate Director, MS Analytics
Georgia Tech

Mahdi Roozbahani
Lecturer, Computational Science & Engineering, Georgia Tech
Founder of Filio, a visual asset management platform

Partly based on materials by
Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos
Text is everywhere

We use documents as primary information artifact in our lives

Our access to documents has grown tremendously thanks to the Internet

- **WWW**: webpages, Twitter, Facebook, Wikipedia, Blogs, ...
- **Digital libraries**: Google books, ACM, IEEE, ...
- Lyrics, closed caption... (youtube)
- Police case reports
- Legislation (law)
- Reviews (products, rotten tomatoes)
- Medical reports (EHR - electronic health records)
- Job descriptions
Big (Research) Questions

... in understanding and gathering information from text and document collections

• establish authorship, authenticity; plagiarism detection

• classification of genres for narratives (e.g., books, articles)

• tone classification; sentiment analysis (online reviews, twitter, social media)

• code: syntax analysis (e.g., find common bugs from students’ answers)
Popular **Natural Language Processing (NLP)** libraries

- **Stanford NLP**
- **OpenNLP**
- **NLTK (python)**

  tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing

Image source: https://stanfordnlp.github.io/CoreNLP/
Outline

• **Preprocessing** (e.g., stemming, remove stop words)

• **Document representation** (most common: bag-of-words model)

• **Word importance** (e.g., word count, TF-IDF)

• **Latent Semantic Indexing** (find “concepts” among documents and words), which helps with retrieval

To learn more:
**CS 4650/7650 Natural Language Processing**
Stemming

Reduce words to their stems (or base forms)

Words: compute, computing, computer, ...

Stem: comput

Several classes of algorithms to do this:

• Stripping suffixes, lookup-based, etc.

http://en.wikipedia.org/wiki/Stemming
Stop words: http://en.wikipedia.org/wiki/Stop_words
Bag-of-words model

Represent each document as a bag of words, ignoring words’ ordering. Why? For simplicity.

Unstructured text becomes a vector of numbers
e.g., docs: “I like visualization”, “I like data”.

1 : “I”
2 : “like”
3 : “data”
4 : “visualization”

“I like visualization” → [1, 1, 0, 1]
“I like data” → [1, 1, 1, 0]
TF-IDF

A word’s importance score in a document, among N documents

When to use it? Everywhere you use “word count”, you can likely use TF-IDF.

**TF**: term frequency
= #appearance a document
(high, if terms appear many times in this document)

**IDF**: inverse document frequency
= log( N / #document containing that term)
(penalize “common” words appearing in almost any documents)

**Final score = TF * IDF**
(higher score → more “characteristic”)

Vector Space Model

Why?

Each document ➔ vector
Each query ➔ vector

Search for documents ➔ find “similar” vectors
Cluster documents ➔ cluster “similar” vectors
Latent Semantic Indexing (LSI)

Main idea
• map each document into some ‘concepts’
• map each term into some ‘concepts’

‘Concept’: ~ a set of terms, with weights.

For example, DBMS_concept:
“data” (0.8),
“system” (0.5),
“retrieval” (0.6)
Latent Semantic Indexing (LSI)

~ pictorially (before) ~

document-term matrix

<table>
<thead>
<tr>
<th></th>
<th>data</th>
<th>system</th>
<th>retrieval</th>
<th>lung</th>
<th>ear</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>doc2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc3</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Latent Semantic Indexing (LSI)

*~ pictorially (after) ~*

#### Term-concept matrix

<table>
<thead>
<tr>
<th>Database concept</th>
<th>Medical concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>system</td>
<td>1</td>
</tr>
<tr>
<td>retrieval</td>
<td>1</td>
</tr>
<tr>
<td>lung</td>
<td>1</td>
</tr>
<tr>
<td>ear</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Document-concept matrix

<table>
<thead>
<tr>
<th>Database concept</th>
<th>Medical concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
</tr>
<tr>
<td>doc2</td>
<td>1</td>
</tr>
<tr>
<td>doc3</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
</tr>
</tbody>
</table>

... and
**Latent Semantic Indexing (LSI)**

**Q:** How to search, e.g., for “system”?  
**A:** find the corresponding concept(s); and the corresponding documents

<table>
<thead>
<tr>
<th>database concept</th>
<th>medical concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>system</td>
<td>1</td>
</tr>
<tr>
<td>retrieval</td>
<td>1</td>
</tr>
<tr>
<td>lung</td>
<td>1</td>
</tr>
<tr>
<td>ear</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>database concept</th>
<th>medical concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1 ←</td>
</tr>
<tr>
<td>doc2</td>
<td>1 ←</td>
</tr>
<tr>
<td>doc3</td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
</tr>
</tbody>
</table>
Latent Semantic Indexing (LSI)

Works like an automatically constructed thesaurus

We may retrieve documents that DON’T have the term “system”, but they contain almost everything else (“data”, “retrieval”)
LSI - Discussion

Great idea,
• to derive ‘concepts’ from documents
• to build a ‘thesaurus’ automatically
• to reduce dimensionality (down to few “concepts”)

How does LSI work?
Uses **Singular Value Decomposition** (SVD)
Problem #1
Find “concepts” in matrices

Problem #2
Compression / dimensionality reduction

Motivation

<table>
<thead>
<tr>
<th></th>
<th>bread</th>
<th>lettuce</th>
<th>tomatoes</th>
<th>beef</th>
<th>chicken</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
SVD is a powerful, generalizable technique.

<table>
<thead>
<tr>
<th>Songs / Movies / Products</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Customers</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
SVD Definition (pictorially)

$$A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} (V_{[m \times r]})^T$$

**Diagonal matrix**

Diagonal entries: concept strengths

- **n** documents
- **m** terms
- **r** concepts

- **m** terms
- **r** concepts
SVD Definition (in words)

\[ A_{[n \times m]} = U_{[n \times r]} \Lambda_{[r \times r]} (V_{[m \times r]})^T \]

**A**: n x m matrix
e.g., n documents, m terms

**U**: n x r matrix
e.g., n documents, r concepts

**Λ**: r x r diagonal matrix
r : rank of the matrix; strength of each ‘concept’

**V**: m x r matrix
e.g., m terms, r concepts
SVD - Properties

**THEOREM** [Press+92]:  
**always possible to decompose** matrix $A$ into  
$A = U \Lambda V^T$

$U$, $\Lambda$, $V$: **unique**, most of the time

$U$, $V$: column **orthonormal**

i.e., columns are **unit vectors**, and **orthogonal** to each other

$U^T U = I$

$V^T V = I$  
($I$: identity matrix)

$\Lambda$: **diagonal** matrix with non-negative diagonal elements, sorted in **decreasing order**
SVD - Example

\[ A = U \Lambda V^T \]

Matrix \( A \) is decomposed into matrices \( U \), \( \Lambda \), and \( V^T \).

The matrix \( A \) represents documents in terms of concepts.

Diagonal matrix \( \Lambda \) shows the concept strengths.

Number of documents: \( n = 4 \)
Number of terms: \( m = 4 \)
Number of concepts: \( r = 2 \)

Document vectors:
- Data
- Info
- Retrieval
- Brain
- Lung

Term vectors:
- CS docs
- MD docs

Concept vectors:
- Concept 1
- Concept 2

Matrix multiplication:
\[ X \cdot X^T = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \end{bmatrix} \cdot \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} = \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix} \]
SVD - Example

\[ \text{document-concept similarity matrix} \times \text{term-concept similarity matrix} = \text{MD docs} \times \text{CS docs} \]

The diagram showcases the application of Singular Value Decomposition (SVD) to a document-concept similarity matrix and a term-concept similarity matrix. The matrices are shown with values that represent the strength of the relationship between concepts and documents. The resulting matrix captures the 'strength' of CS-concept and MD-concept.
SVD - Interpretation #1

‘documents’, ‘terms’ and ‘concepts’:

\( U \): document-concept similarity matrix
\( V \): term-concept similarity matrix
\( \Lambda \): diagonal elements: concept “strengths”
SVD - Interpretation #1

‘documents’, ‘terms’ and ‘concepts’:
Q: if $A$ is the document-to-term matrix, what is the similarity matrix $A^T A$ ?
A:

Q: $A A^T$ ?
A:
‘documents’, ‘terms’ and ‘concepts’:
Q: if $A$ is the document-to-term matrix, what is the similarity matrix $A^T A$?
A: term-to-term ($[m \times m]$) similarity matrix

Q: $A \ A^T$?
A: document-to-document ($[n \times n]$) similarity matrix
SVD properties

\[ \mathbf{V} \text{ are the eigenvectors of the covariance matrix } \mathbf{A}^\top \mathbf{A} \text{ (term-to-term } [m \times m] \text{ similarity matrix)} \]

\[ \mathbf{A}^\top \mathbf{A} = (\mathbf{U} \Sigma \mathbf{V}^\top)^\top (\mathbf{U} \Sigma \mathbf{V}^\top) = \mathbf{V} \Sigma^2 \mathbf{V}^\top \]

\[ \mathbf{U} \text{ are the eigenvectors of the Gram (inner-product) matrix } \mathbf{A} \mathbf{A}^\top \text{ (doc-to-doc } [n \times n] \text{ similarity matrix)} \]

\[ \mathbf{A} \mathbf{A}^\top = (\mathbf{U} \Sigma \mathbf{V}^\top)(\mathbf{U} \Sigma \mathbf{V}^\top)^\top = \mathbf{U} \Sigma^2 \mathbf{U}^\top \]

SVD is closely related to PCA, and can be numerically more stable. For more info, see:


SVD - Interpretation #2

Find the best axis to project on.

("best" = minimize sum of squares of projection errors)

Beautiful visualization explaining PCA:
http://setosa.io/ev/principal-component-analysis/
**SVD - Interpretation #2**

$\mathbf{U} \Lambda$ gives the **coordinates** of the points in the projection axis

$$\mathbf{A} = \mathbf{U} \Lambda \mathbf{V}^T$$

- **U**: Coordinates of the points in the projection axis
- **Lambda**: Variance (‘spread’) on the v1 axis
- **V**: Directions of the singular values

![Diagram](image-url)
SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?
**SVD - Interpretation #2**

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
= \begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27 \\
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71 \\
\end{bmatrix}
\]
SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?
A: set the smallest singular values to zero:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0.53 & 0 \\
0.80 & 0 \\
0.27 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
9.64 & 0 & 0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 6.9 & 0 & 0 & 0 & 0.71 & 0.71 \\
\end{bmatrix}
\]
SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?
A: set the smallest singular values to zero:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
= \begin{bmatrix}
0.18 \\
0.36 \\
0.18 \\
0.90 \\
0 \\
0 \\
0
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
\end{bmatrix}
\]
**SVD - Interpretation #2**

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero:

```
 1 1 1 0 0
2 2 2 0 0
1 1 1 0 0
5 5 5 0 0
0 0 0 2 2
0 0 0 3 3
0 0 0 1 1
```

~

```
 1 1 1 0 0
2 2 2 0 0
1 1 1 0 0
5 5 5 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

```
SVD - Interpretation #3
finds non-zero ‘blobs’ in a data matrix

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
= \begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27 \\
\end{bmatrix}
\times \begin{bmatrix}
9.64 & 0 \\
0 & 5.29 \\
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71 \\
\end{bmatrix}
\times
\]
SVD - Interpretation #3

finds non-zero ‘blobs’ in a data matrix

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix} = \begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27 \\
\end{bmatrix} \times \begin{bmatrix}
9.64 & 0 \\
0 & 5.29 \\
\end{bmatrix} \times \begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71 \\
\end{bmatrix}
\]
SVD - Interpretation #3

• finds non-zero ‘blobs’ in a data matrix =
• ‘communities’ (bi-partite cores, here)

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

Row 1
Row 4
Row 5
Row 7

Col 1
Col 3
Col 4
SVD - Complexity

$O(n^*m^*m)$ or $O(n^*n^*m)$ (whichever is less)

Faster version, if just want singular values
or if we want first $k$ singular vectors
or if the matrix is sparse [Berry]

No need to write your own!
Available in most linear algebra packages
(LINPACK, matlab, Splus/R, mathematica ...
...
Case Study

How to do queries with LSI?
Case Study

How to do queries with LSI?

For example, how to find documents with ‘data’?

data  info  retrieval  brain  lung
1 1 1  0 0
2 2 2  0 0
1 1 1  0 0
5 5 5  0 0
0 0 0  2 2
0 0 0  3 3
0 0 0  1 1

CS docs

= x

MD docs

9.64 0
0 5.29
0 0.53
0 0.80
0 0.27

0.18 0
0.36 0
0.18 0
0.90 0

0.58 0.58 0.58 0 0
0 0 0 0.71 0.71
Case Study

How to do queries with LSI?

For example, how to find documents with ‘data’?
A: map query vectors into ‘concept space’ – how?

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
= 
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27 \\
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71 \\
\end{bmatrix}
\]
Case Study

How to do queries with LSI?

For example, how to find documents with ‘data’?

A: map query vectors into ‘concept space’, using \textbf{inner product} (cosine similarity) with each ‘concept’ vector $v_i$

\begin{equation}
q = \begin{bmatrix}
data 
info 
retrieval 
brain 
lung 
\end{bmatrix} = \begin{bmatrix}
1 
0 
0 
0 
0 
0 
\end{bmatrix}
\end{equation}
Case Study
How to do queries with LSI?
Compactly, we have:

$$q \cdot V = q_{\text{concept}}$$

<table>
<thead>
<tr>
<th>data</th>
<th>info</th>
<th>retrieval</th>
<th>brain</th>
<th>lung</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

term-concept similarity matrix

$$\begin{pmatrix}
0.5 & 8 & 0 \\
0.5 & 8 & 0 \\
0 & 0.7 & 1 \\
0 & 0.7 & 1 \\
\end{pmatrix}$$
Case Study

How would the document (‘information’, ‘retrieval’) be handled?
Case Study

How would the document ('information', 'retrieval') be handled?

\[ d \ V = \ d_{\text{concept}} \]

SAME!
Document (‘information’, ‘retrieval’) will be retrieved by query (‘data’), even though it does not contain ‘data’!!
Switch Gear to
Text Visualization
Word/Tag Cloud (still popular?)

http://www.wordle.net
Word Counts (words as bubbles)

http://www.infocaptor.com/bubble-my-page
Phrase Net

Visualize pairs of words satisfying a pattern ("X and Y")

http://hint.fm/projects/phrasenet/
Termite: Topic Model Visualization
http://vis.stanford.edu/papers/termite
Termite: Topic Model Visualization

http://vis.stanford.edu/papers/termite