Introduction to Data Science

Text Analysis

Kerstin Bunte and **George Azzopardi**

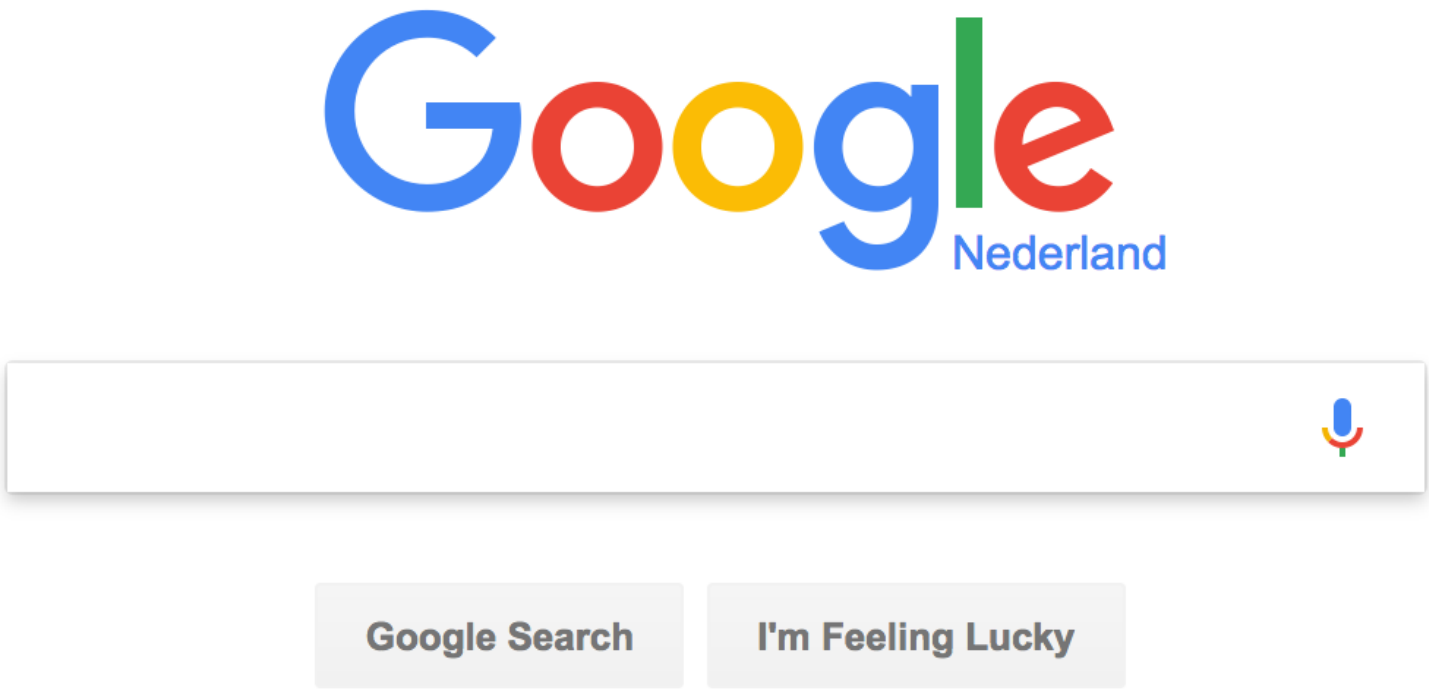
WMCS16002, semester Ia 2018

Mathematician Ted Kaczynski!

Learning goals

- Understand what text mining is
- Learn how to preprocess text
- Learn how to compare documents
- Compute the bag of words model
- Identify model topics
- Build bi-grams, . . . n-grams

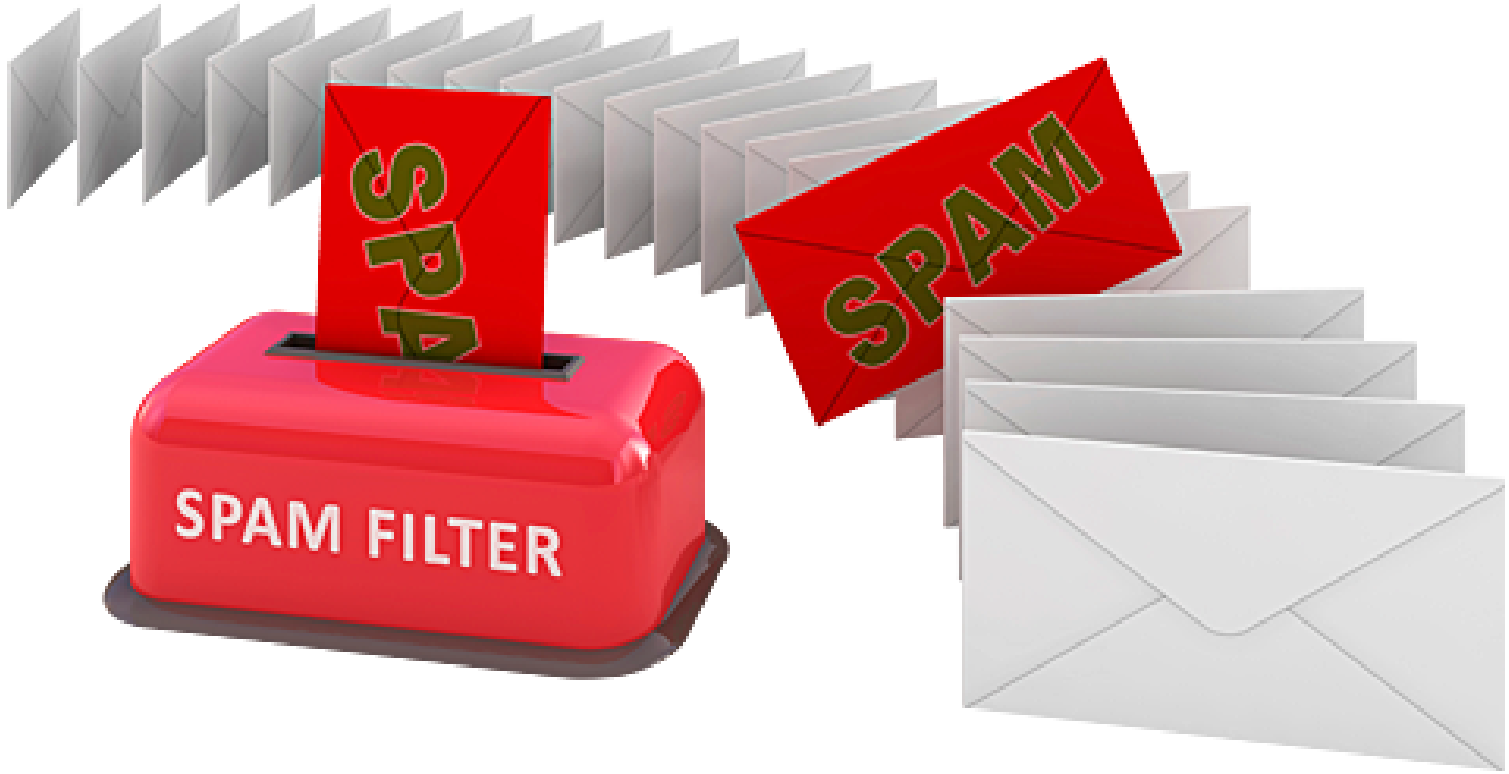
Search


Google
Nederland
Google Search
I'm Feeling Lucky
Google.nl offered in: Nederlands

Extract Knowledge

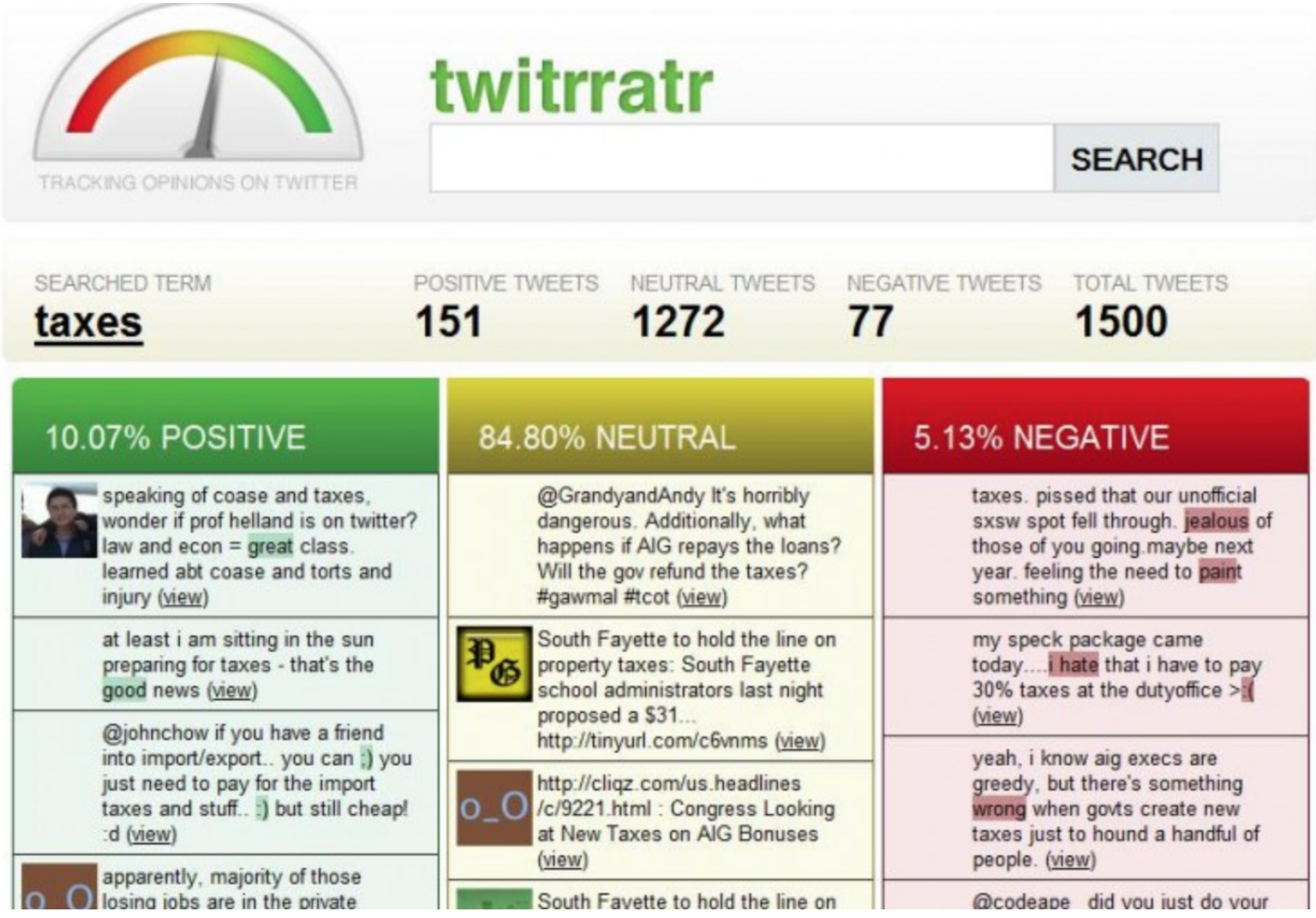
In 2011, Watson competed against the legendary human competitors and outperformed them! Watson had access to 200 million pages of structured and unstructured content consuming four terabytes of disk storage including the full text of Wikipedia

Classify



You can thank text classification algorithms for keeping spam out of your inbox

Detect Emotions



Trend of publications

Go to Web of Knowledge

Let us start small...

How do we extract features from text?

Let us start small...

How do we extract features from text?

1. Tokenization
2. Case folding
3. Removal of short and stop words
4. Stemming
5. Lemmatization

Tokenization

Tokenization

the isolation of word-like units from a text

- base for all further text processing
- accuracy of tokenisation affects higher level processing

Tokenization

Tokenization

the isolation of word-like units from a text

- base for all further text processing
- accuracy of tokenisation affects higher level processing

  ATTN: “garbage in, garbage out”

Tokenization

Tokenization

the isolation of word-like units from a text

- base for all further text processing
- accuracy of tokenisation affects higher level processing

  ATTN: “garbage in, garbage out”

- tokenizing by: spaces, spaces and punctuation, etc.

Tokenization is More Difficult than it seems

Tokenization is More Difficult than it seems

Ambiguous examples

- state-of-the-art
- Résumé vs. resume

Tokenization is More Difficult than it seems

Ambiguous examples

- state-of-the-art
- Résumé vs. resume
- computer science, San Francisco

  this is called a *noun phrase*

Tokenization Beyond English

- German/Dutch combine individual words to create longer compound words

  searching for Hund should retrieve Windhund while searching for Blau should likely not return Blauzungekrankheit

Tokenization Beyond English

- German/Dutch combine individual words to create longer compound words

  searching for Hund should retrieve Windhund while searching for Blau should likely not return Blauzungekrankheit

- Some Asian languages have no spaces

Tokenization Beyond English

- German/Dutch combine individual words to create longer compound words

  searching for Hund should retrieve Windhund while searching for Blau should likely not return Blauzungekrankheit

- Some Asian languages have no spaces
- Some languages have very complex morphology: e.g. Uygarlastiramadiklarimizdanmissinizcasina (tr)

Tokenization Best Practices

- No single tokenizer that will work in every scenario.

Tokenization Best Practices

- No single tokenizer that will work in every scenario.
- Common to pair a standard tokenization technique with a lookup table
  e.g. San Francisco

Case folding

Case folding

= reducing all the letters in a word to lowercase

Works

- "**Look**, I took the liberty to **look** at that parrot when I got it home..."

Case folding

Case folding

= reducing all the letters in a word to lowercase

Works

- "**Look**, I took the liberty to **look** at that parrot when I got it home..."

Does not work

- General Motors → general motors

Case folding

Case folding

= reducing all the letters in a word to lowercase

Works

- "**Look**, I took the liberty to **look** at that parrot when I got it home..."

Does not work

- General Motors → general motors
- The Who → the who

  Heuristic strategies (e.g. no case fold in mid-sentence)

Removal of short and stop words

- Typically, remove words that have fewer than 3 characters

Removal of short and stop words

- Typically, remove words that have fewer than 3 characters
- Remove the most common words in a language, known as **stop words**
  E.g. the, and, which, where, ...

Stemming

Stemming

= reducing a word to a stem (or root) form *algorithmically*

- Number: stemmer, stemmers
- Tense: stems, stemmed, stemming
- Gender: stemmer, *stemmeress*
- ...

Algorithmic stemmer: Porter

Rules for chopping the end of some words[1]

Step 1a

SSES	-> SS	caresses	-> caress
IES	-> I	ponies	-> poni
		ties	-> ti
SS	-> SS	caress	-> caress
S	->	cats	-> cat

Step 1b

(m>0) EED	-> EE	feed	-> feed
		agreed	-> agree
(*v*) ED	->	plastered	-> plaster
		bled	-> bled
(*v*) ING	->	motoring	-> motor
		sing	-> sing

Link: The -ing rule Link: The full description of the rules

[1] A good discussion at: http://snowballstem.org/texts/introduction.html and a place to try them online: http://textanalysisonline.com/

Porter Stemmer[2] – Example

Call me Ishmael . Some **year** ago **have littl** or no money in my **purs** , and **noth** particular to interest me on shore , I thought I would sail about a **littl** and see the **wateri** part of the world .

vs.

Call me Ishmael. Some years ago having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world.

[2] The rules of the stemmer:
`http://snowball.tartarus.org/algorithms/porter/stemmer.html`

Porter Stemmer[2] – Example

Call me Ishmael . Some **year** ago **have littl** or no money in my **purs** , and **noth** particular to interest me on shore , I thought I would sail about a **littl** and see the **wateri** part of the world .

vs.

Call me Ishmael. Some years ago having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world.

Go to `http://localhost:8888/notebooks/Tokeniser_Stemming.ipynb`

[2] The rules of the stemmer:
`http://snowball.tartarus.org/algorithms/porter/stemmer.html`

Limiting Porter Damage...

```
new_word = PorterStemmer(old_word)
dUS=Enchanted_Dicitonary(en-US)
if  new_word != old_word  and  len(old_word)>3:
    if  new_word exist in dUS:
        Replacing the old_word by new_word
```

See: PyEnchant

Limiting Porter Damage...

```
new_word = PorterStemmer(old_word)
dUS=Enchanted_Dicitonary(en-US)
if  new_word != old_word  and  len(old_word)>3:
    if  new_word exist in dUS:
        Replacing the old_word by new_word
```

See: PyEnchant

Surely, Porter was born before Twitter / SMS:

This is soooooo LOOOVELY, sooo goooood, noooo waaay!!!

Another Headache... HTML

"this is really important text"

A **chunker** is a special tokenizer function that breaks text up into large chunks rather than individual tokens.

```
>>> from enchant.tokenize import get_tokenizer, HTMLChunker
>>>
>>> tknzr = get_tokenizer("en_US"])
>>> [w for w in tknzr("this is <span class='important'>really important</span> text")]
[('this', 0), ('is', 5), ('span', 9), ('class', 14), ('important', 21), ('really', 32), ('important', 39),
>>>
>>>
>>> tknzr = get_tokenizer("en_US",chunkers=(HTMLChunker,))
>>> [w for w in tknzr("this is <span class='important'>really important</span> text")]
[('this', 0), ('is', 5), ('really', 32), ('important', 39), ('text', 56)]
```

Lemmatization

Lemmatization

= obtaining the single word that allows you to group together a bunch of inflected forms

- Harder than stemming
- Takes context into account
- Uses some kind of dictionary

Lemmatization

Lemmatization

= obtaining the single word that allows you to group together a bunch of inflected forms

- Harder than stemming
- Takes context into account
- Uses some kind of dictionary

Pattern = web mining module for Python

`http://www.clips.ua.ac.be/pages/pattern`

WordNet Lemmatization Example

Call me Ishmael . Some **year** ago **have** little or no money in my purse , and nothing particular to interest me on shore , I **think** I would sail about a little and see the watery part of the world .

vs.

Call me Ishmael. Some years ago having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world.

Stemming vs. Lemmatization

- **Common goal**: To reduce inflectional forms and sometimes derivationally related forms of a word to a common base form

Stemming vs. Lemmatization

- **Common goal**: To reduce inflectional forms and sometimes derivationally related forms of a word to a common base form
- Stemming
  - crude heuristic process that chops off the end of words
  - correct most of the time
  - simple to implement and fast

Stemming vs. Lemmatization

- **Common goal**: To reduce inflectional forms and sometimes derivationally related forms of a word to a common base form
- Stemming
  - crude heuristic process that chops off the end of words
  - correct most of the time
  - simple to implement and fast
- Lemmatization
  - Use vocabulary but slower than stemming
  - Morphological analysis of words
  - Return the base or dictionary form of a word

Stemming vs. Lemmatization

- **Common goal**: To reduce inflectional forms and sometimes derivationally related forms of a word to a common base form
- Stemming
  - crude heuristic process that chops off the end of words
  - correct most of the time
  - simple to implement and fast
- Lemmatization
  - Use vocabulary but slower than stemming
  - Morphological analysis of words
  - Return the base or dictionary form of a word
- Examples
  - "better" -> lemma: "good". It is missed from stemming
  - "walking" -> lemma: "walk", stem: "walk"
  - "meeting" -> two meanings. Lemmatization takes context into account

NLP Word Features

1. Root words
2. Part of speech (POS) tags
   Ex: POSTagging
3. Named entities

NLP Word Features

1. Root words
2. Part of speech (POS) tags
   Ex: POSTagging
3. Named entities

Traditionally these features were manually annotated in **language corpora**

Language Corpora

Corpus

= a public and frozen set of (usually annotated) documents

- Nice for training and benchmarking
- Often NLP libraries come with corpora

  Ex: NLTKCorpora

Word Feature: Information Content

- IC = metric to denote the importance of a term in a corpus

Word Feature: Information Content

- IC = metric to denote the importance of a term in a corpus
- Combines
  - 1. knowledge of its hierarchical structure from an ontology
  - 2. statistics on its actual usage in text derived from a corpus

Word Feature: Information Content

- IC = metric to denote the importance of a term in a corpus
- Combines
  - 1. knowledge of its hierarchical structure from an ontology
  - 2. statistics on its actual usage in text derived from a corpus
- e.g. $IC(necklace) > IC(jewelry)$ in an English corpus
  jewelry is more general and thus less interesting

Word Feature: Information Content

- IC = metric to denote the importance of a term in a corpus
- Combines
  - 1. knowledge of its hierarchical structure from an ontology
  - 2. statistics on its actual usage in text derived from a corpus
- e.g. $IC(necklace) > IC(jewelry)$ in an English corpus
  jewelry is more general and thus less interesting

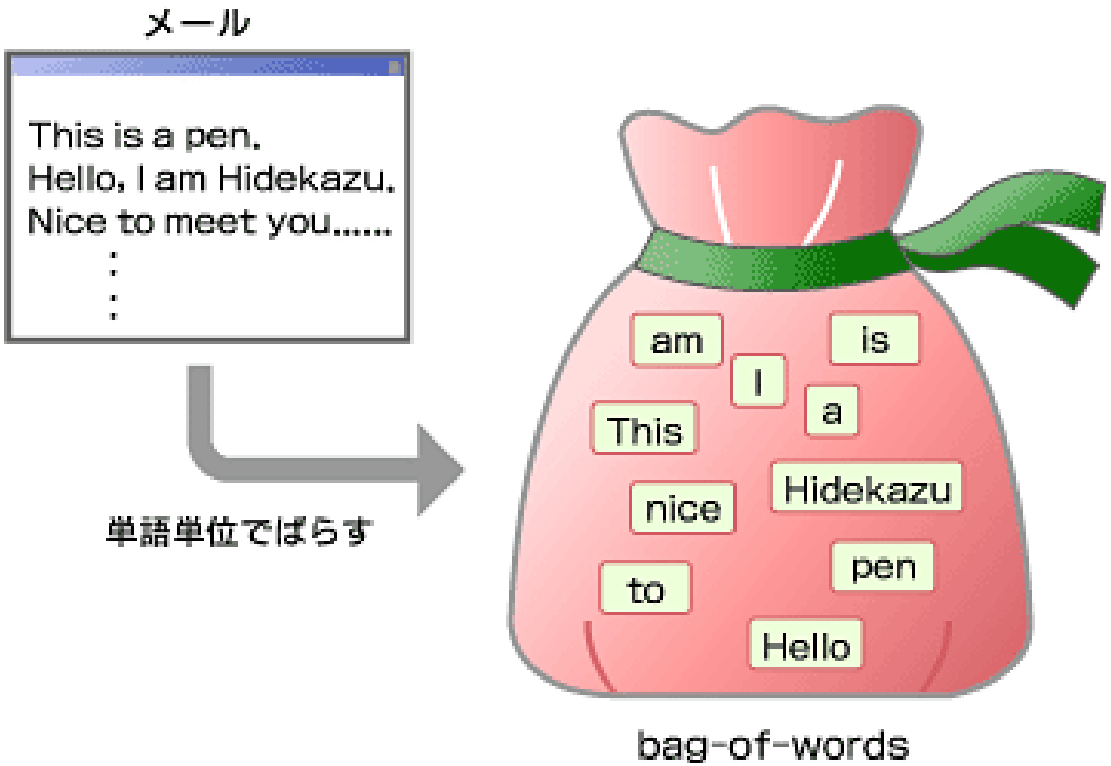
Problems

- terms not in the corpus automatically receive zero IC
  analyzing U2 with the Brown corpus

Enough about words!

Documents

Document Representation: Bag-of-words



- A set of terms ignoring: order, context, etc.
- Each word is considered a term or token

Clear Limitations but still Powerful

"the dog bites the man" vs. "the man bites the dog"!

$\{bites, dog, man, the\}$

Clear Limitations but still Powerful

"the dog bites the man" vs. "the man bites the dog"!

$\{bites, dog, man, the\}$

"the overwhelming evidence is that the **judicious use of single-term identifiers is preferable to the incorporation of more complex entities** extracted from the texts themselves..." [SB88]

Term Frequency

- The weight of each term in a document

  Proportional to the number of occurrences of the term in that document

Term Frequency

- The weight of each term in a document

  Proportional to the number of occurrences of the term in that document

$$TF_1(term, doc) = \frac{\texttt{count of term in doc}}{\texttt{count words in doc}}$$

Term Frequency

- The weight of each term in a document

  Proportional to the number of occurrences of the term in that document

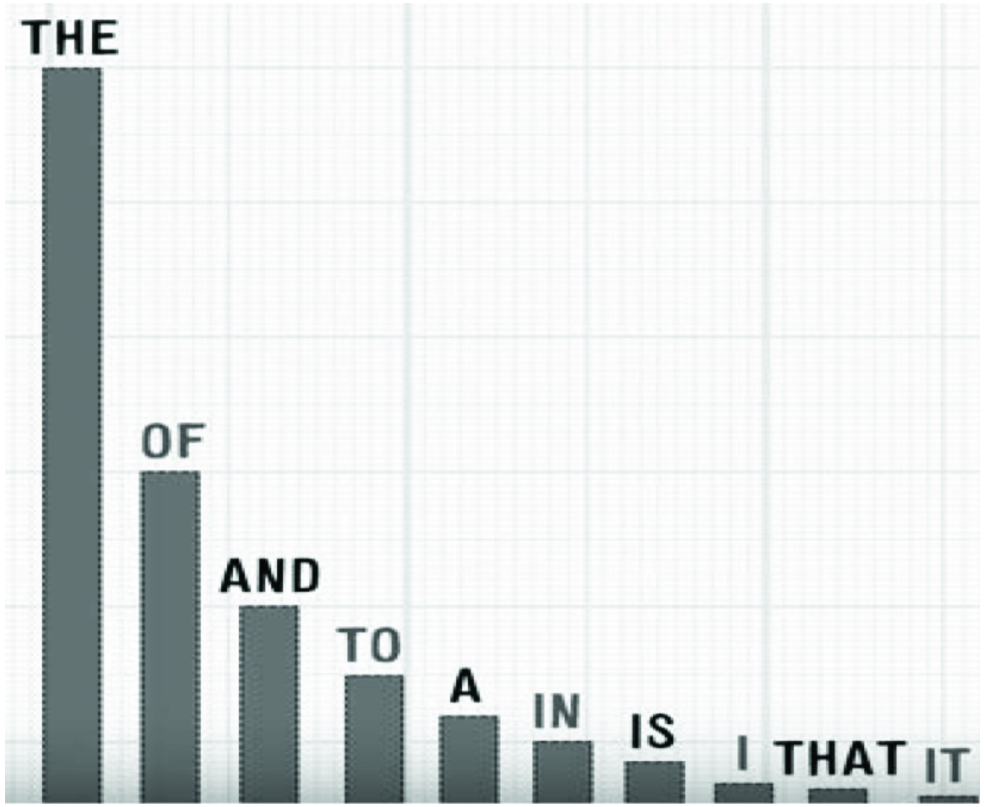
$$TF_1(term, doc) = \frac{\texttt{count of term in doc}}{\texttt{count words in doc}}$$

- Roughly follows Zipf's Law

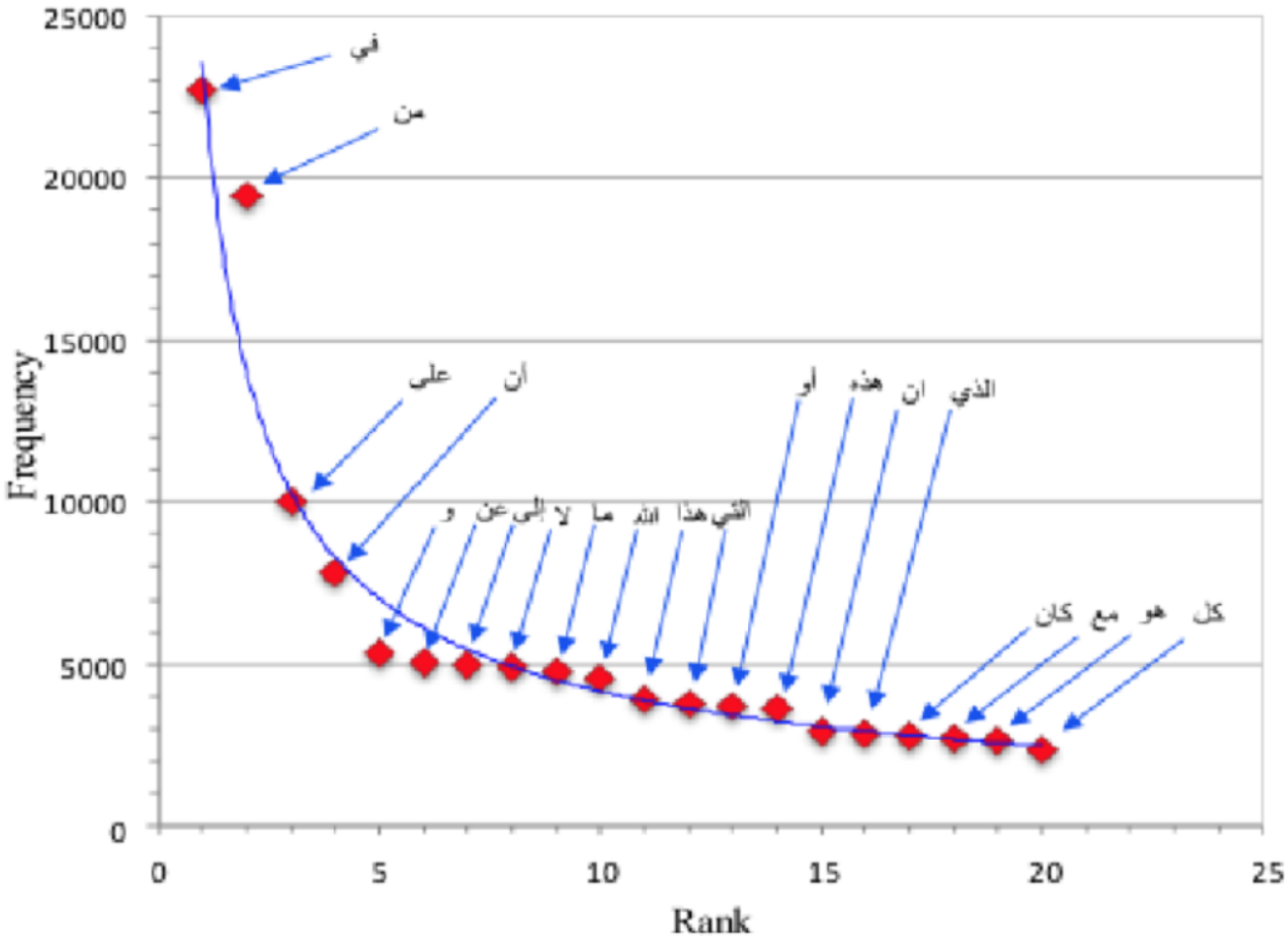
  the i-th most common item has freq of approx. 1/i of most frequent – see next slides!

- this is why we often work with $TF = log(TF_1)$

Zipf's Law - Top 10 English Words

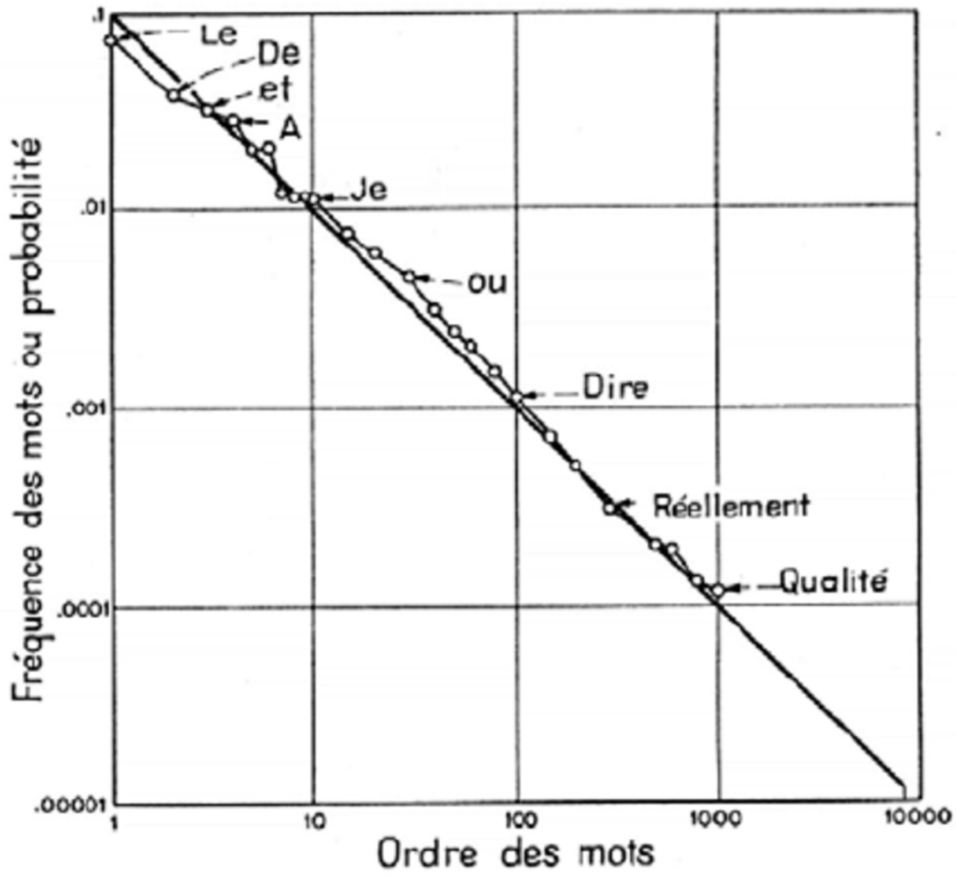


Zipf's Law - Top 20 Arabic Words



[source]

Zipf's Law - Top 10K French



"a good approximation to what is demonstrably a much more complicated distribution of word frequencies" (S. Piantadosi)

Demo

TF in Shakespeare

(Ex: TFShake)

Term Frequency Inverse Document Frequency (TFxIDF)

TF.IDF

= numerical statistic reflecting the importance of a word in a doc in a corpus

Considering **a word important for a document** if it appears

- often within that document (TF), and

Term Frequency Inverse Document Frequency (TFxIDF)

TF.IDF

= numerical statistic reflecting the importance of a word in a doc in a corpus

Considering **a word important for a document** if it appears

- often within that document (TF), and
- rarely in other documents of the corpus (IDF)

Computing TF.IDF

$$N = |docs|$$

Computing TF.IDF

$$N = |docs|$$

$$DF(term, docs) = \texttt{<count docs in which term occurs>}$$

Computing TF.IDF

$$N = |docs|$$

$$DF(term, docs) = \texttt{<count docs in which term occurs>}$$

$$IDF(term, docs) = log\left(\frac{N+1}{DF(term, docs)+1}\right) + 1$$

Computing TF.IDF

$$N = |docs|$$

$$DF(term, docs) = \texttt{<count docs in which term occurs>}$$

$$IDF(term, docs) = log\left(\frac{N+1}{DF(term, docs)+1}\right) + 1$$

$$TF.IDF(term, doc, docs) = TF(term, doc) * IDF(term, docs)$$

Demo

Simple TF.IDF Example

Ex: TF.IDF.Example

Vector Space Models[3]

Vector Space Model

= the representation of a set of documents as vectors in a common space

[3] parts of some following slides are based on slides by Malvina Nissim

Vector Space Model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term: $w(term, doc)$:

$$\boldsymbol{V}(d) \;=\; (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$

- Note that also raw frequency could be used (that's what we are doing)

Vector Space Model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term: $w(term, doc)$:

$$\boldsymbol{V}(d) \;=\; (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$

- Note that also raw frequency could be used (that's what we are doing)

representation of three documents using term raw frequencies:
d_1, d_2, d_3

	d_1	d_2	d_3
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

Vector Space Model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term: $w(term, doc)$:

$$\boldsymbol{V}(d) \;=\; (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$

- Note that also raw frequency could be used (that's what we are doing)

representation of three documents using term raw frequencies:
d_1, d_2, d_3

	d_1	d_2	d_3
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

- **Question**: how do we compute the **similarity between documents**?

Vector normalization and similarity

- Similarity between vectors
  $\rightarrow$ inner product $\boldsymbol{V}(d_1) \cdot \boldsymbol{V}(d_2)$

Vector normalization and similarity

- Similarity between vectors
  $\rightarrow$ inner product $\boldsymbol{V}(d_1) \cdot \boldsymbol{V}(d_2)$

- What about the length of a vector?
  Longer documents will be represented with longer vectors, but that does not mean they are more important

- Euclidian normalization (vector length normalization):

$$\boldsymbol{v}(d) = \frac{\boldsymbol{V}(d)}{\|\boldsymbol{V}(d)\|} \qquad \text{where } \|\boldsymbol{V}(d)\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Vector normalization and similarity

- Similarity between vectors
  $\rightarrow$ inner product $\boldsymbol{V}(d_1) \cdot \boldsymbol{V}(d_2)$
- Similarity given by the *cosine* measure between normalized vectors:

$$sim(d_1, d_2) \;=\; \boldsymbol{v}(d_1) \cdot \boldsymbol{v}(d_2)$$

Example (Manning et al[4])

- $sim(d1, d2) = \boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2)$
  let's backtrack this:
- $\boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\boldsymbol{v}(d_i) = \frac{\boldsymbol{V}(d_i)}{\|\boldsymbol{V}(d)\|}$$

- Euclidean length:

$$\|\boldsymbol{V}(d)\| = \sqrt{\sum_{i=1}^{n} \boldsymbol{V}_i^2(d)}$$

[4] See [MRS08]

Example (Manning et al[4])

- $sim(d1, d2) = \boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2)$
- $\boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\boldsymbol{v}(d_i) = \frac{\boldsymbol{V}(d_i)}{\|\boldsymbol{V}(d)\|}$$

- Euclidean length:

$$\|\boldsymbol{V}(d)\| = \sqrt{\sum_{i=1}^{n} \boldsymbol{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

[4] See [MRS08]

Example (Manning et al[4])

- $sim(d1, d2) = \boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2)$
- $\boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\boldsymbol{v}(d_i) = \frac{\boldsymbol{V}(d_i)}{\|\boldsymbol{V}(d)\|}$$

- Euclidean length:

$$\|\boldsymbol{V}(d)\| = \sqrt{\sum_{i=1}^{n} \boldsymbol{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\boldsymbol{V}(d1)\| = \sqrt{115^2 + 10^2 + 2^2}$$

[4] See [MRS08]

Example (Manning et al[4])

- $sim(d1, d2) = \boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2)$
- $\boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\boldsymbol{v}(d_i) = \frac{\boldsymbol{V}(d_i)}{\|\boldsymbol{V}(d)\|}$$

- Euclidean length:

$$\|\boldsymbol{V}(d)\| = \sqrt{\sum_{i=1}^{n} \boldsymbol{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\boldsymbol{V}(d1)\| = \sqrt{115^2 + 10^2 + 2^2}$$

$$\boldsymbol{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$

$$\boldsymbol{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$

$$\boldsymbol{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

[4] See [MRS08]

Example (Manning et al[4])

- $sim(d1, d2) = \boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2)$
- $\boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\boldsymbol{v}(d_i) = \frac{\boldsymbol{V}(d_i)}{\|\boldsymbol{V}(d)\|}$$

- Euclidean length:

$$\|\boldsymbol{V}(d)\| = \sqrt{\sum_{i=1}^{n} \boldsymbol{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\boldsymbol{V}(d2)\| = \sqrt{58^2 + 7^2 + 0}$$

$$\boldsymbol{v}(d2_1) = \frac{58}{\sqrt{58^2+7^2+0}} = 0.993$$

$$\boldsymbol{v}(d2_2) = \frac{7}{\sqrt{58^2+7^2+0}} = 0.120$$

$$\boldsymbol{v}(d2_3) = 0$$

[4] See [MRS08]

Example (Manning et al[4])

- $sim(d1, d2) = \boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2)$
- $\boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\boldsymbol{v}(d_i) = \frac{\boldsymbol{V}(d_i)}{\|\boldsymbol{V}(d)\|}$$

- Euclidean length:

$$\|\boldsymbol{V}(d)\| = \sqrt{\sum_{i=1}^{n} \boldsymbol{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\boldsymbol{V}(d3)\| = \sqrt{20^2 + 11^2 + 6^2}$$

$$\boldsymbol{v}(d3_1) = \frac{20}{\sqrt{20^2+11^2+6^2}} = 0.847$$
$$\boldsymbol{v}(d3_2) = \frac{11}{\sqrt{20^2+11^2+6^2}} = 0.466$$
$$\boldsymbol{v}(d3_3) = \frac{6}{\sqrt{20^2+11^2+6^2}} = 0.254$$

[4] See [MRS08]

Example (Manning et al[4])

- $sim(d1, d3) = \boldsymbol{v}(d1) \cdot \boldsymbol{v}(d3)$
- $\boldsymbol{v}(d1) \cdot \boldsymbol{v}(d3) = \sum_{i=1}^{n} d1_i d3_i$

$$\boldsymbol{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$
$$\boldsymbol{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$
$$\boldsymbol{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

[i=1] $0.996 * 0.847+$
[i=2] $0.087 * 0.466+$
[i=3] $0.017 * 0.254 =$
$= 0.888$

$$\boldsymbol{v}(d3_1) = \frac{20}{\sqrt{20^2+11^2+6^2}} = 0.847$$
$$\boldsymbol{v}(d3_2) = \frac{11}{\sqrt{20^2+11^2+6^2}} = 0.466$$
$$\boldsymbol{v}(d3_3) = \frac{6}{\sqrt{20^2+11^2+6^2}} = 0.254$$

[4] See [MRS08]

Example (Manning et al[4])

- $sim(d1, d2) = \boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2)$
- $\boldsymbol{v}(d1) \cdot \boldsymbol{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$

$$\boldsymbol{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$
$$\boldsymbol{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$
$$\boldsymbol{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

[i=1] $0.996 * 0.993+$
[i=2] $0.087 * 0.120+$
[i=3] $0.017 * 0.000 =$
$= 0.999$

$$\boldsymbol{v}(d2_1) = \frac{58}{\sqrt{58^2+7^2+0}} = 0.993$$
$$\boldsymbol{v}(d2_2) = \frac{7}{\sqrt{58^2+7^2+0}} = 0.120$$
$$\boldsymbol{v}(d2_3) = 0$$

[4] See [MRS08]

Example (Manning et al[4])

summing up:

dictionary	$\boldsymbol{v}(d_1)$	$\boldsymbol{v}(d_2)$	$\boldsymbol{v}(d_3)$
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254

$$sim(d_1, d_2) = 0.999$$
$$sim(d_1, d_3) = 0.888$$

[4] See [MRS08]

New documents

- each new document n is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1

- $sim(n, d) = \boldsymbol{v}(n) \cdot \boldsymbol{v}(d)$

New documents

- each new document n is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1

- $sim(n, d) = \boldsymbol{v}(n) \cdot \boldsymbol{v}(d)$

New documents

- each new document n is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1

- $sim(n, d) = \boldsymbol{v}(n) \cdot \boldsymbol{v}(d)$
- with $n =< jealous, gossip >$

  we obtain:

$$\begin{aligned} \boldsymbol{v}(n) \cdot \boldsymbol{v}(d_1) &= 0.074 \\ \boldsymbol{v}(n) \cdot \boldsymbol{v}(d_2) &= 0.085 \\ \boldsymbol{v}(n) \cdot \boldsymbol{v}(d_3) &= 0.509 \end{aligned}$$

Vector Space Models

Useful for

- Search
- Clustering
- **Discovering the main topics in a corpus**

Topic Models

Statistical models that

- Determine the themes over the documents / corpus

- Useful for: organizing, searching, summarizing texts

Topic Models

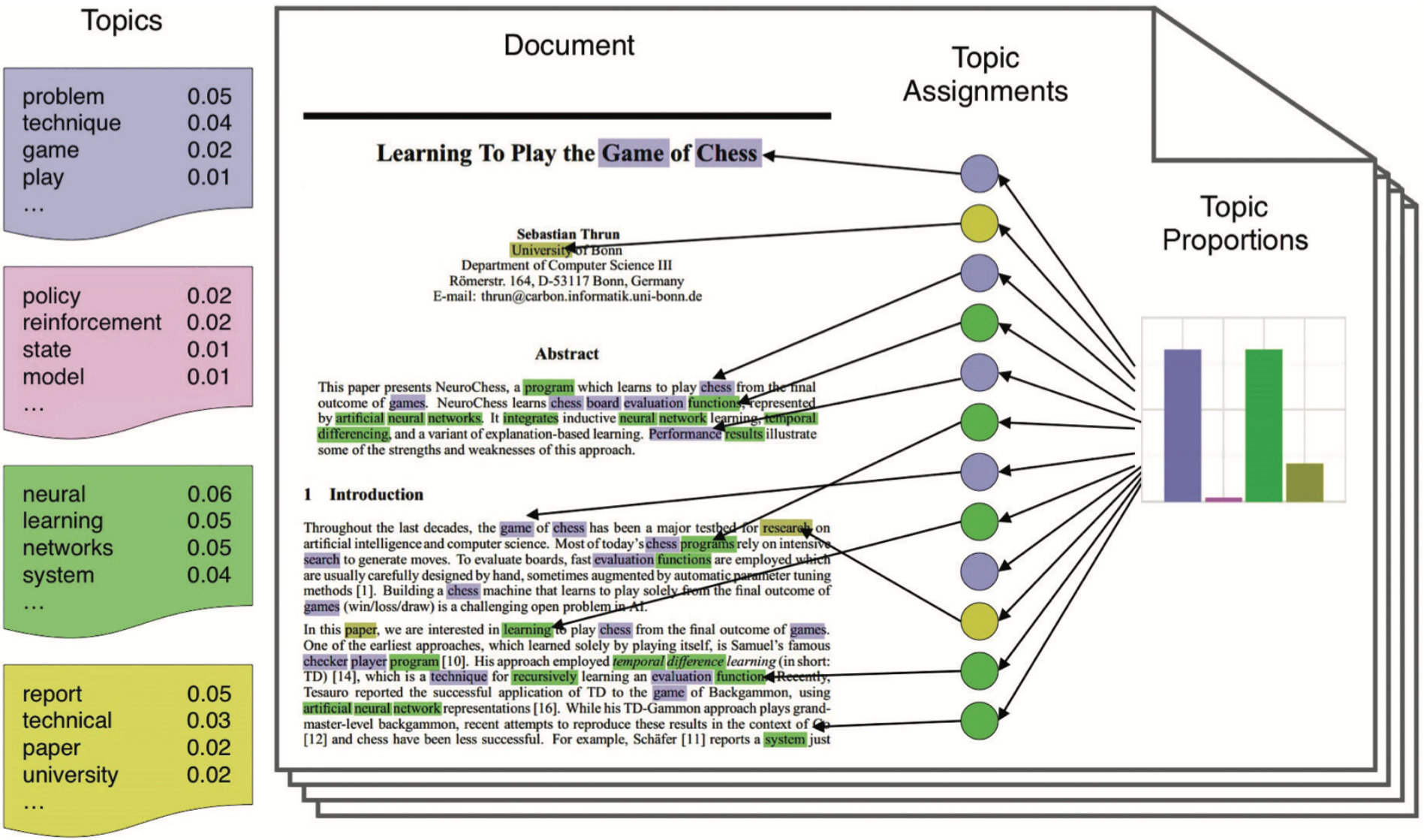
Statistical models that

- Determine the themes over the documents / corpus

- Useful for: organizing, searching, summarizing texts

Based on the **distributional hypothesis**: words close in meaning will occur in similar pieces of text.

Topic Models Example[5]



[5] See [ES15]

Topic

Topic

- cluster of words that frequently occur together and share the same theme.
- (formally) a distribution over a fixed vocabulary of words

  Different topics = different distributions over the same vocabulary

- can emerge solely based on text analysis

Latent Dirichlet Allocation

- Document is represented as a **mix of latent topics**

Latent Dirichlet Allocation

- Document is represented as a **mix of latent topics**
- Number of latent topics (k) is predefined

  unlike k-means a document can belong to multiple topics

  unlike GMM features do not need to be normally distributed

Latent Dirichlet Allocation

- Document is represented as a **mix of latent topics**
- Number of latent topics (k) is predefined

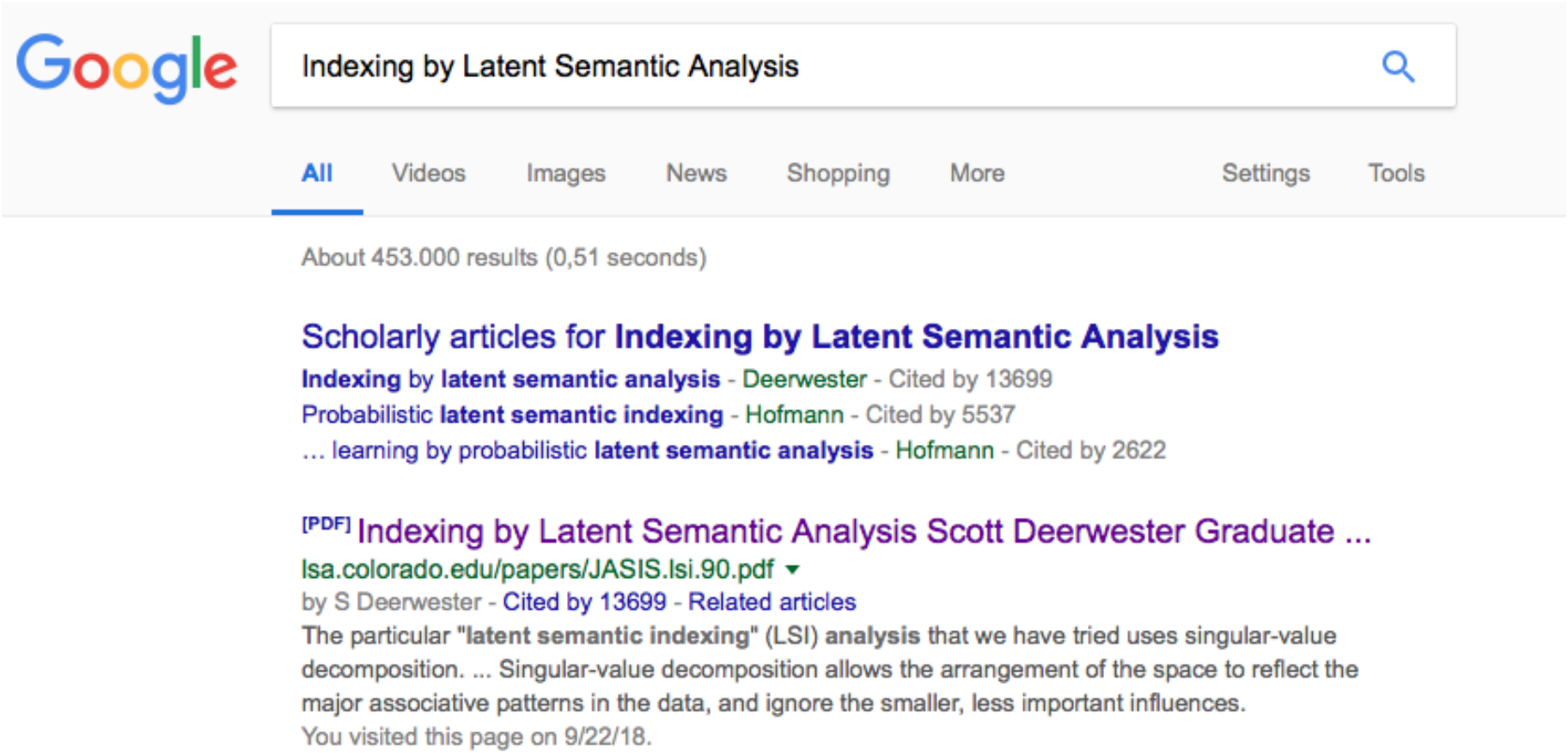
  unlike k-means a document can belong to multiple topics

  unlike GMM features do not need to be normally distributed

One of the simplest topic models

implemented in many libraries

Latent Semantic Analysis



Example

Titles:

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
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

Titles:

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
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

- Query: "Human Computer Interaction"

Example

Titles:

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
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

- Query: "Human Computer Interaction"
- Simple term matching techniques would return documents c1, c2 and c4 - they share one or more terms with the query.

Example

Titles:

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
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

- Query: "Human Computer Interaction"
- Simple term matching techniques would return documents c1, c2 and c4 - they share one or more terms with the query.
- Documents c3 and c5, which are also relevant, are missed as they have no terms in common with the query.

Latent Semantic Analysis

- Create a term-document matrix (M)

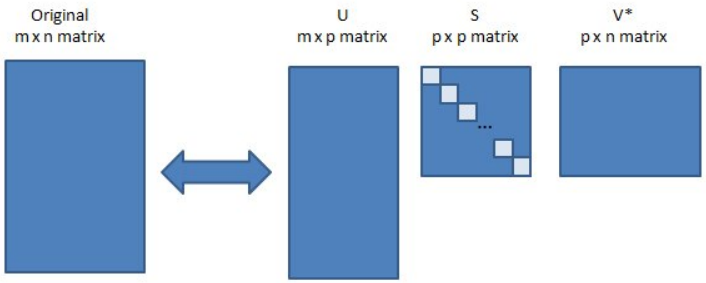
  word counts = rows / documents = columns

Latent Semantic Analysis

- Create a term-document matrix (M)

  word counts = rows / documents = columns

- Perform Singular Value Decomposition on M
- Reduce the dimension of the Σ in the SVD

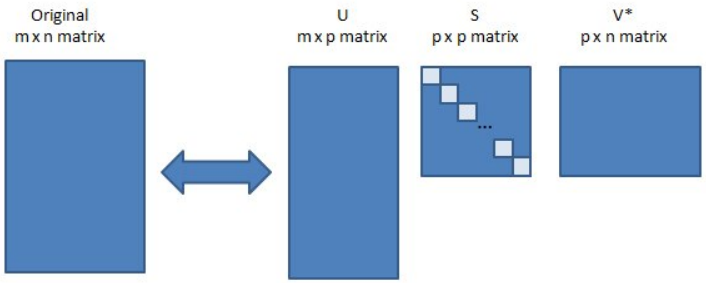


Latent Semantic Analysis

- Create a term-document matrix (M)

  word counts = rows / documents = columns

- Perform Singular Value Decomposition on M
- Reduce the dimension of the Σ in the SVD



- Create a new matrix M' based on the reduced Σ

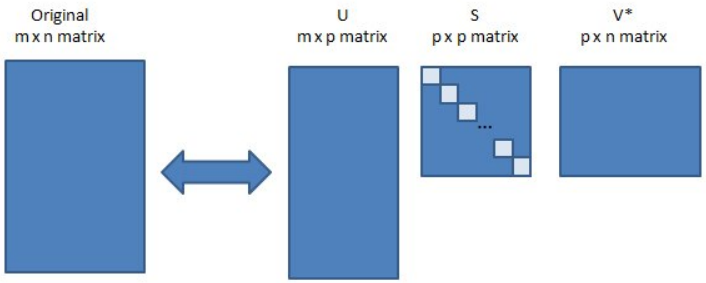
  words can be compared using cosine distance

Latent Semantic Analysis

- Create a term-document matrix (M)

  word counts = rows / documents = columns

- Perform Singular Value Decomposition on M
- Reduce the dimension of the Σ in the SVD



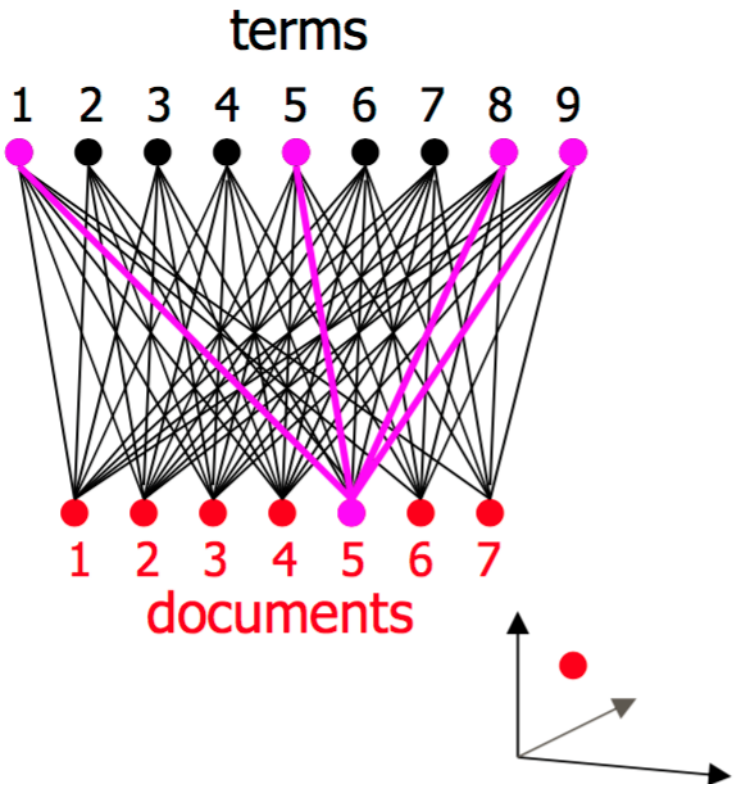
- Create a new matrix M' based on the reduced Σ

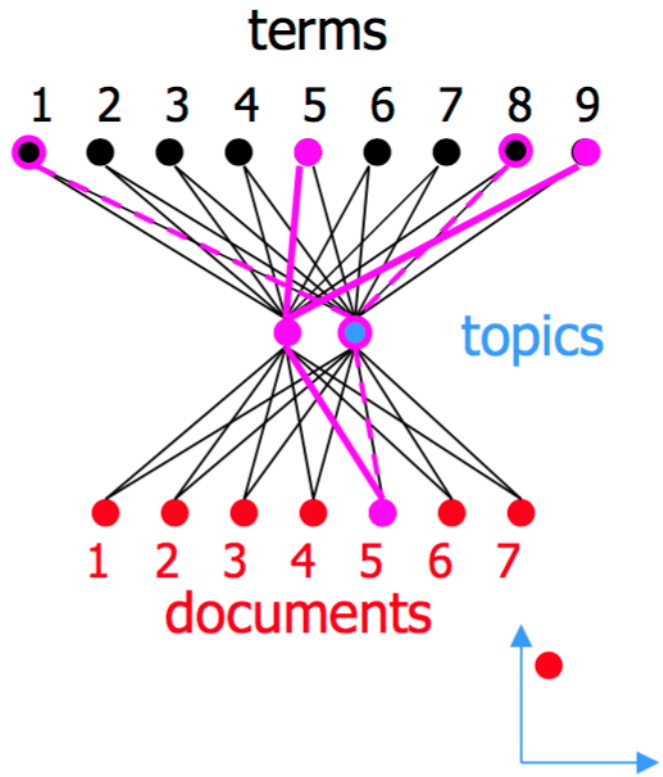
  words can be compared using cosine distance

- Now we can

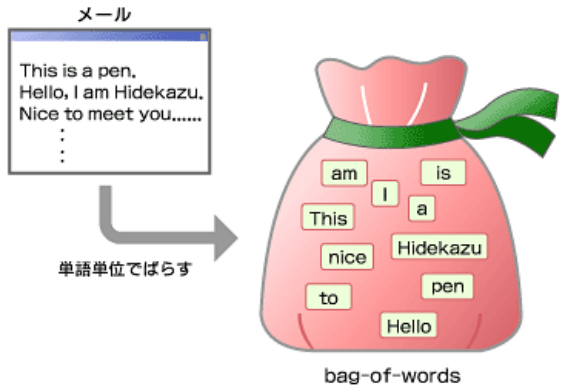
  - Search as we did in vector space models
  - Cluster the document vectors to discover topics

Singular Value Decomposition





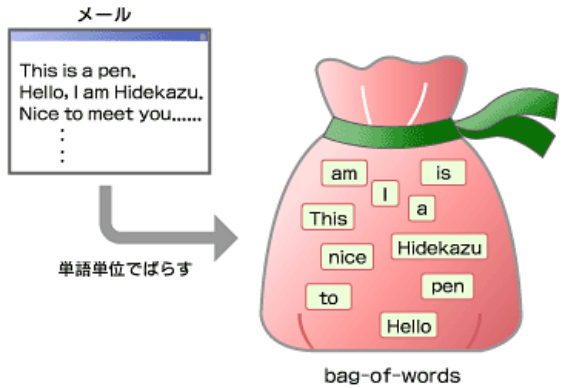
Limitations of the Bag-of-Words



- Cannot capture phrases and **multi-word expressions**

  e.g. San Juan, a priori, etc.

Limitations of the Bag-of-Words



- Cannot capture phrases and **multi-word expressions**

  e.g. San Juan, a priori, etc.

- Doesn't account for potential misspellings or word derivations.

N-grams

- Given a corpus of text, the n-grams are the sequences of n consecutive words that are in the corpus.
- Can be with words or letters (characters)

Sliding windows

- the cat that sat on the sofa also sat on the mat.
- the cat that sat on the sofa also sat on the mat.
- the cat that sat on the sofa also sat on the mat.

there are 3-grams, 4-grams, etc...

the sliding window can be larger than n!

Sliding windows

- the cat that sat on the sofa also sat on the mat.
- the cat that sat on the sofa also sat on the mat.
- the cat that sat on the sofa also sat on the mat.

there are 3-grams, 4-grams, etc...

the sliding window can be larger than n!

- are all combinations relevant?
- are the frequent ones better?

Bigrams beyond chance: *collocations*

Collocations: recurrent combinations of words that co-occur more often than chance, sometimes with non-compositional meaning

- strong tea (NB: ???powerful tea)
- best practice
- ...

Bigrams beyond chance: *collocations*

What does “more often than chance” mean?

Bigrams beyond chance: *collocations*

What does "more often than chance" mean?

We want to compare the likelihood of 2 words next to each other being a **chance event** vs. being a **surprise**[6][7]

- Do the two words appear next to each other more than we might expect, based on what we know about their individual frequencies?

[6] `tdunning.blogspot.nl/2008/03/surprise-and-coincidence.html`
[7] See [Dun93]

Determining strength of collocation

We use association measures to determine whether two words occur together by chance or in a significant way

- pointwise mutual information (PMI)
- chi-square (χ^2)
- log-likelihood ratio (LL)

.

Ex: Collocations

References I

[Dun93] Ted Dunning, *Accurate methods for the statistics of surprise and coincidence*, Comput. Linguist. **19** (1993), no. 1, 61–74.

[ES15] EMC Education Services, *Data science and big data analytics*, John Wiley and Sons, Inc., Indianapolis, Indiana, 2015.

[LKGL05] Mircea Lungu, Adrian Kuhn, Tudor Gîrba, and Michele Lanza, *Interactive exploration of semantic clusters*, 3rd International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2005), 2005, pp. 95–100.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, *Introduction to information retrieval*, Cambridge University Press, New York, NY, USA, 2008.

[SB88] Gerard Salton and Christopher Buckley, *Term-weighting approaches in automatic text retrieval*, Inf. Process. Manage. **24** (1988), no. 5, 513–523.