Text Preprocessing and Similarity

[DAT640] Information Retrieval and Text Mining

Krisztian Balog
University of Stavanger

August 20, 2024

In this module

1. Representing text
2. Text similarity
3. Text preprocessing

Representing text

Question

What makes working with text challenging?

Representing text

- How to represent text for machine understanding?
  - To measure the similarity between two texts (later in this lecture)
  - To decide whether some document belongs to a category (text classification)
  - To decide which groups of documents belong together (text clustering)
  - To decide if a document is relevant to a search query (search)
  - ...
- We need a mathematical representation for words (referred to as *terms*) and for sequences of words (referred to as *documents*)

Representing words

- Discrete (sparse) representation
  - Each word is given a unique **term ID** (integer)
  - The **vocabulary** V contains a mapping between terms and their IDs
  - $\Rightarrow$ This is the representation we use for now

ID	**term**
1	discrete
2	sparse
3	document
4	text
5	word
6	this
	...

Vocabulary

Representing words (2)

- Continuous (dense) representation
  - Each word is represented as a real-valued vector in a (relatively) small dimensional latent space, i.e., an **embedding vector**

    $\vec{w} =$

0.15	0.07	0.83	0.46	...	0.02

  - Similar/related words are close to each other in the embedding space
  - (To be covered in the second half of the course)

Representing documents

- Terms are represented by their respective IDs in the vocabulary
  - There might be out of vocabulary (OOV) words, i.e., terms not seen during the construction of the vocabulary
  - Those are substituted with a special OOV token (if term positions matter) or simply ignored
- A document might be represented as a **sequence of words**
- A document d can also be represented as a **term vector**
  - Each element of the vector corresponds to a term in the vocabulary
  - The value might represent
    - the presence/absence of a word (0/1)
    - the frequency of the word (int)
    - the word's importance (real)
  - This is a **sparse** representation, as most values will be zeros
  - Word position is ignored, therefore it is called the **bag-of-words** representation

ID	**term**
1	discrete
2	sparse
3	document
4	text
5	word
6	this
	...

Vocabulary

$d =$ "this is sparse text"

$d = [t_6, OOV, t_2, t_4]$

$\vec{d} = \langle 0, 1, 0, 1, 0, 1, \dots \rangle$

Representing a collection of documents

- Document-term matrix, where
  - Rows correspond to documents (n)
  - Columns correspond to terms in the vocabulary (m)
- The matrix is huge, but most of the values are zeros; stored as a sparse matrix

	t_1	t_2	t_3	$\dots$	t_m
d_1	1	0	2		0
d_2	0	1	0		2
d_3	0	0	1		0
$\dots$					
d_n	0	1	0		0

Document-term matrix

Text similarity

Text similarity

- Core ingredient in many text mining and information retrieval problems
- Need to express the similarity between two pieces of text (referred to as *documents*, for simplicity)
- The choice of the similarity measure is closely tied with how documents are represented

Jaccard similarity

- **Jaccard similarity**: only the presence/absence of terms in documents is considered, with no regard to magnitude
- Defined as the ratio of shared terms and total terms in two documents:

$$\mathit{sim}_{\text{Jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \ ,$$

  - where X and Y represent the terms that appear in documents d_1 and d_2, respectively

Jaccard similarity

- Jaccard similarity for term vector-based representations:

$$sim_{\text{Jaccard}}(\mathbf{x}, \mathbf{y}) = \frac{\sum_i \mathbb{1}(x_i) \times \mathbb{1}(y_i)}{\sum_i \mathbb{1}(x_i + y_i)} \ ,$$

  - here $\mathbb{1}(x)$ is an indicator function (1 if $x > 0$ and 0 otherwise).

Example

	term 1	term 2	term 3	term 4	term 5
doc x	1	0	1	0	3
doc y	0	2	4	0	1

Table: Document-term vectors with term frequencies.

$$\mathbf{x} = \langle 1, 0, 1, 0, 3 \rangle \qquad \mathbf{y} = \langle 0, 2, 4, 0, 1 \rangle$$

$$sim_{\text{Jaccard}}(\mathbf{x}, \mathbf{y}) = \frac{0 + 0 + 1 + 0 + 1}{1 + 1 + 1 + 0 + 1} = \frac{2}{4}$$

Cosine similarity

- **Cosine similarity**: the cosine of the angle between the two document vectors plotted in their high-dimensional space; the larger the angle, the more dissimilar the documents are:

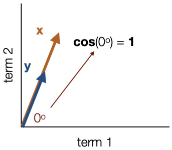
$$sim_{\cos}(x, y) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \ ,$$

  - where **x** and **y** are the term vectors corresponding to documents d_1 and d_2, respectively
  - Term weights (x_i and y_i) may be raw term counts or TF-IDF-weighted frequencies

Cosine similarity - Geometric interpretation

	term 1	term 2
doc x	1	2
doc y	2	4

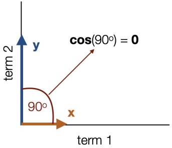
$$\mathit{sim}_{\cos}(x, y) = 1$$



Cosine similarity - Geometric interpretation

	term 1	term 2
doc x	1	0
doc y	0	2

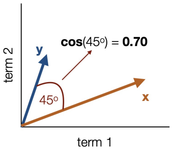
$$sim_{\text{cos}}(x, y) = 0$$



Cosine similarity - Geometric interpretation

	term 1	term 2
doc x	4	2
doc y	1	3

$$\mathit{sim}_{\text{cos}}(x, y) = 0.7$$



Cosine similarity

Example

	term 1	term 2	term 3	term 4	term 5
doc x	1	0	1	0	3
doc y	0	2	4	0	1

Table: Document-term vectors with term frequencies.

$$\mathbf{x} = \langle 1, 0, 1, 0, 3 \rangle \qquad \mathbf{y} = \langle 0, 2, 4, 0, 1 \rangle$$

$$\begin{aligned} \mathit{sim}_{\mathrm{cos}}(x, y) &= \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \\ &= \frac{1 \times 0 + 0 \times 2 + 1 \times 4 + 0 \times 0 + 3 \times 1}{\sqrt{1^2 + 0^2 + 1^2 + 0^2 + 3^2}\sqrt{0^2 + 2^2 + 4^2 + 0^2 + 1^2}} = \frac{7}{\sqrt{11}\sqrt{21}} \end{aligned}$$

Exercise

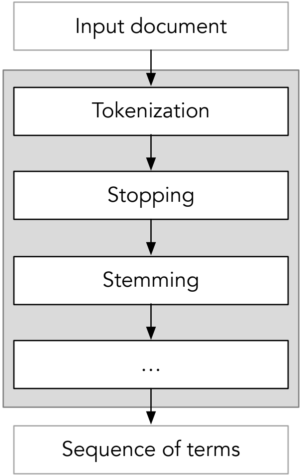
E1-1 Term vector similarity

Text preprocessing

Text preprocessing

$$
\begin{aligned}
&d_1 : [t_1, t_3, t_3] \\
&d_2 : [t_2, t_4, t_5, t_6, t_2] \\
&d_3 : [t_3, t_2, t_3] \\
&\ldots
\end{aligned}
$$

Text preprocessing pipeline



Tokenization

- Parsing a string into individual words (tokens)
- Splitting is usually done along white spaces, punctuation marks, or other types of content delimiters (e.g., HTML markup)
- Sounds easy, but can be surprisingly complex, even for English
  - Even worse for many other languages

Question

What could be the issues with tokenization along whitespace and punctuation marks?

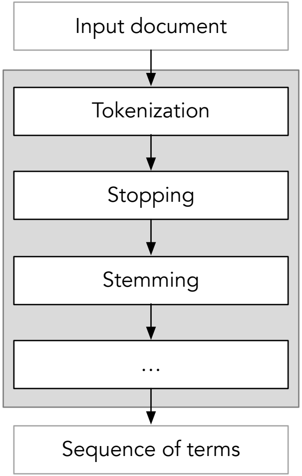
Tokenization issues

- Apostrophes can be a part of a word, a part of a possessive, or just a mistake
  - *rosie o'donnell, can't, 80's, 1890's, men's straw hats, master's degree, ...*
- Capitalized words can have different meaning from lower case words
  - *Bush, Apple, ...*
- Special characters are an important part of tags, URLs, email addresses, etc.
  - *C++, C#, ...*
- Numbers can be important, including decimals
  - *nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358, ...*
- Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
  - *I.B.M., Ph.D., www.uis.no, F.E.A.R., ...*

Common practice

- Process documents in two stages
    - First pass is focused on identifying markup or tags
    - Second pass is done on the appropriate parts of the document structure
- Treat hyphens, apostrophes, periods, etc. like spaces
- Ignore capitalization
- Index even single characters
    - *o'connor* $\Rightarrow$ *o connor*

Text preprocessing pipeline



Stopword removal

- Function words that have little meaning apart from other words: *the*, *a*, *an*, *that*, *those*, ..
- These are considered **stopwords** and are removed
- A stopwords list can be constructed by taking the top-k (e.g., 50) most common words in a collection
    - May be customized for certain domains or applications

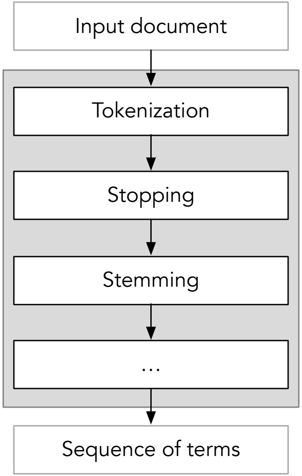
Example (minimal stopword list)

a	as	by	into	not	such	then	this	with
an	at	for	is	of	that	there	to	
and	be	it	it	on	the	these	was	
are	but	in	no	or	their	they	will	

Question

What about a text like “to be or not to be”?

Text preprocessing pipeline



Stemming

- Reduce the different forms of a word that occur to a common stem
  - Inflectional (plurals, tenses)
  - Derivational (making verbs nouns etc.)
- In most cases, these have the same or very similar meanings
- Basic types of stemmers
  - Algorithmic
  - Dictionary-based
  - Hybrid algorithmic-dictionary

Suffix-s stemmer

- Assumes that any word ending with an 's' is plural
  - *cakes* $\Rightarrow$ *cake*, *dogs* $\Rightarrow$ *dog*
- Cannot detect many plural relationships (false negative)
  - *centuries* $\Rightarrow$ *century*
- In rare cases it detects a relationship where it does not exist (false positive)
  - *is* $\Rightarrow$ *i*

Porter stemmer

- Most popular algorithmic stemmer
- Consists of 5 steps, each step containing a set of rules for removing suffixes
- Produces stems not words
- Makes a number of errors and difficult to modify

Example step (1 of 5)

Step 1a:

- Replace ***sses*** by ***ss*** (e.g., stresses → stress).
- Delete ***s*** if the preceding word part contains a vowel not immediately before the ***s*** (e.g., gaps → gap but gas → gas).
- Replace ***ied*** or ***ies*** by ***i*** if preceded by more than one letter, otherwise by ***ie*** (e.g., ties → tie, cries → cri).
- If suffix is ***us*** or ***ss*** do nothing (e.g., stress → stress).

Step 1b:

- Replace ***eed***, ***eedly*** by ***ee*** if it is in the part of the word after the first non-vowel following a vowel (e.g., agreed → agree, feed → feed).
- Delete ***ed***, ***edly***, ***ing***, ***ingly*** if the preceding word part contains a vowel, and then if the word ends in ***at***, ***bl***, or ***iz*** add ***e*** (e.g., fished → fish, pirating → pirate), or if the word ends with a double letter that is not ***ll***, ***ss*** or ***zz***, remove the last letter (e.g., falling→ fall, dripping → drip), or if the word is short, add ***e*** (e.g., hoping → hope).
- Whew!

Porter stemmer examples

False positives *(should not have the same stem)*	**False negatives** *(should have the same stem)*
organization/organ	european/europe
generalization/generic	cylinder/cylindrical
numerical/numerous	matrices/matrix
policy/police	urgency/urgent
university/universe	create/creation
addition/additive	analysis/analyses
negligible/negligent	useful/usefully
execute/executive	noise/noisy
past/paste	decompose/decomposition

Krovetz stemmer

- Hybrid algorithmic-dictionary
- Word checked in dictionary
  - If present, either left alone or replaced with exception stems
  - If not present, word is checked for suffixes that could be removed
- After removal, dictionary is checked again
- Produces words not stems

Stemmer comparison

Original text

Document will describe **marketing** strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for **agrochemicals**, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales

Porter stemmer

market strateg carr compan agricultur chemic report predict market share chemic report market statist **agrochem** pesticid herbicid fungicid insecticid fertil predict sale stimul demand price cut volum sale

Krovetz stemmer

marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic **agrochemic** pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale

Effect of stemming

- Generally a small (but significant) effectiveness improvement for English
- Can be crucial for some languages (e.g., Arabic, Russian)

Exercise

E1-2 Text preprocessing

Summary

- Representing text (terms, documents, collection of documents)
- Measuring text similarity (Jaccard and cosine)
- Text preprocessing (tokenization, stopwords removal, stemming)

Reading

- Text Data Management and Analysis (Zhai&Massung)
  - Section 8.1