Introduction to Text Mining

Applied Data Analysis and Visualization I

Anastasia Giachanou, Ayoub Bagheri
Department of Methodology and Statistics

Schedule

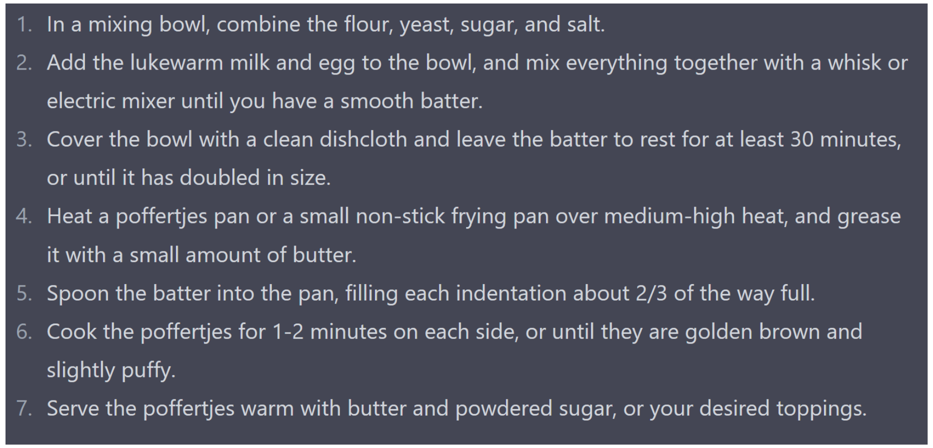
What	When
Classification	Week 5
Interactive visualizations with R shiny	Week 6
Tree-based Methods	Week 7
Text Mining	Week 8
Network Analysis	Week 9
Exam	Week 10

Outline

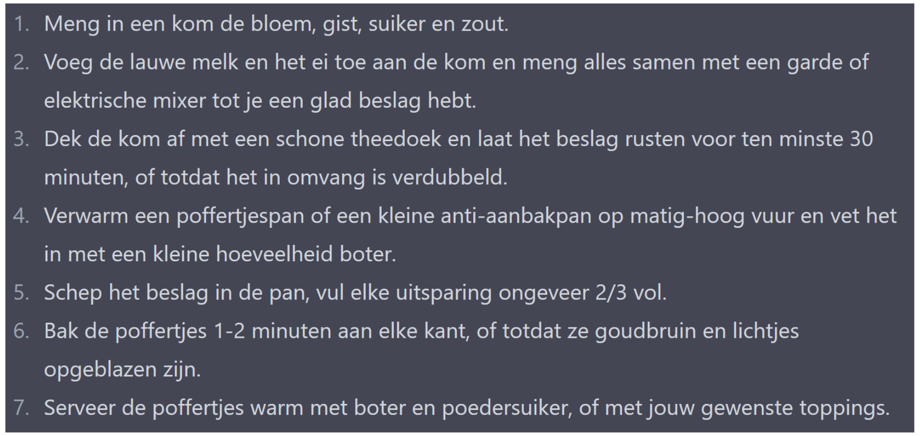
- Text mining
- Pre-processing text data
- Vector space model
- Text classification
- Word embedding

Introduction

Chat-GPT

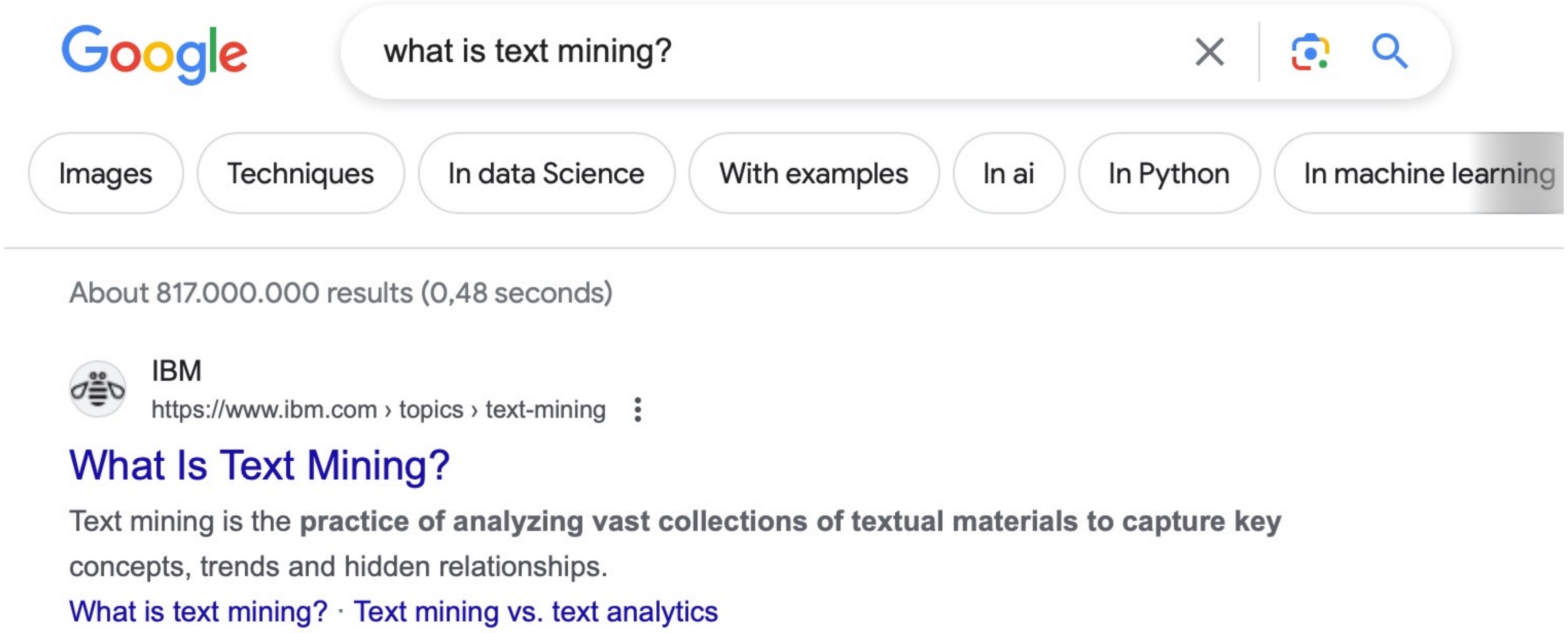


1. In a mixing bowl, combine the flour, yeast, sugar, and salt.
2. Add the lukewarm milk and egg to the bowl, and mix everything together with a whisk or electric mixer until you have a smooth batter.
3. Cover the bowl with a clean dishcloth and leave the batter to rest for at least 30 minutes, or until it has doubled in size.
4. Heat a poffertjes pan or a small non-stick frying pan over medium-high heat, and grease it with a small amount of butter.
5. Spoon the batter into the pan, filling each indentation about 2/3 of the way full.
6. Cook the poffertjes for 1-2 minutes on each side, or until they are golden brown and slightly puffy.
7. Serve the poffertjes warm with butter and powdered sugar, or your desired toppings.



1. Meng in een kom de bloem, gist, suiker en zout.
2. Voeg de lauwe melk en het ei toe aan de kom en meng alles samen met een garde of elektrische mixer tot je een glad beslag hebt.
3. Dek de kom af met een schone theedoek en laat het beslag rusten voor ten minste 30 minuten, of totdat het in omvang is verdubbeld.
4. Verwarm een poffertjespan of een kleine anti-aanbakpan op matig-hoog vuur en vet het in met een kleine hoeveelheid boter.
5. Schep het beslag in de pan, vul elke uitsparing ongeveer 2/3 vol.
6. Bak de poffertjes 1-2 minuten aan elke kant, of totdat ze goudbruin en lichtjes opgeblazen zijn.
7. Serveer de poffertjes warm met boter en poedersuiker, of met jouw gewenste toppings.

Search engines



what is text mining?

About 817.000.000 results (0,48 seconds)

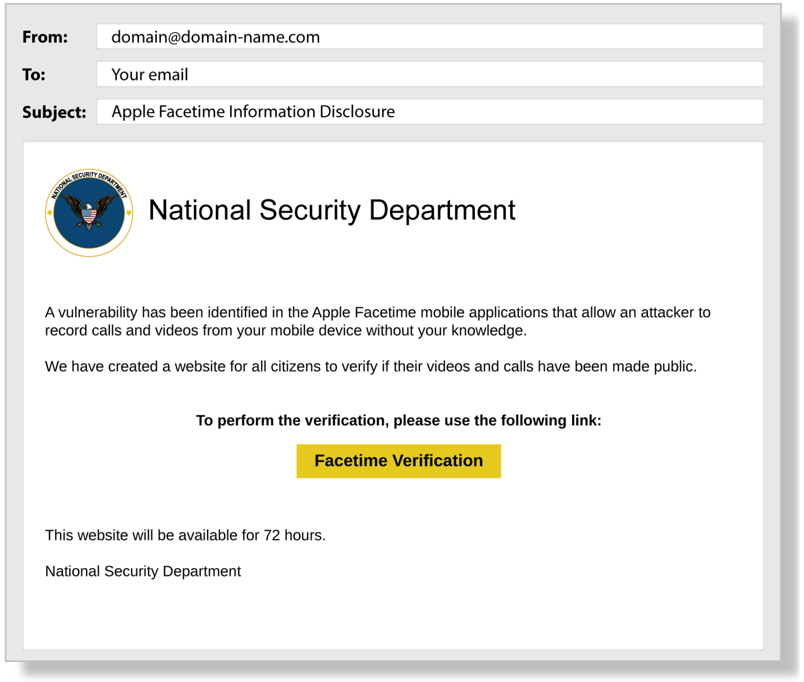
IBM
https://www.ibm.com › topics › text-mining

What Is Text Mining?

Text mining is the **practice of analyzing vast collections of textual materials to capture key** concepts, trends and hidden relationships.

What is text mining? · Text mining vs. text analytics

Email spam filtering



From: domain@domain-name.com

To: Your email

Subject: Apple Facetime Information Disclosure

National Security Department

A vulnerability has been identified in the Apple Facetime mobile applications that allow an attacker to record calls and videos from your mobile device without your knowledge.

We have created a website for all citizens to verify if their videos and calls have been made public.

To perform the verification, please use the following link:

Facetime Verification

This website will be available for 72 hours.

National Security Department

Stock market prediction

- Can the stock market be predicted by extracting and analyzing large collections of publicly available textual documents?

- A study used StockTwits (a platform where people share ideas about the stock market) as a data source and applied sentiment analysis and five different algorithms to check that

Renault T (2019) Sentiment analysis and machine learning in finance: a comparison of methods and models on one million messages. Digit Finance

Text mining

- “The discovery by computer of **new, previously unknown information**, by automatically extracting information from different written resources”, Hearst (1999).
- Text mining describes a set of linguistic, statistical, and machine learning techniques that model and **structure the information content of textual sources**. (Wikipedia)

Why text mining?

- **Text data is everywhere**, websites (e.g., news), social media (e.g., twitter), databases (e.g., doctors' notes), digital scans of printed materials, …
- A lot of world's data is in **unstructured text format**
- Applications in industry: search, machine translation, sentiment analysis, question answering, …
- Applications in science: cognitive modeling, understanding bias in language, automated systematic literature reviews, …

Language is hard!

- Different things can mean more or less the same (“data science” vs. “statistics”)
- Context dependency (“The queue was long”);
- Same words with different meanings (“bank”);
- Lexical ambiguity (“we saw her duck”)
- Irony, sarcasm (“You should swallow disinfectant”?)
- Figurative language (“He has a heart of stone”)
- Negation (“not good” vs. “good”), spelling variations, jargon, abbreviations
- All the above is different over languages, 99% of work is on English!

Pre-processing Text Data

Text preprocessing

- Is an approach for cleaning and noise removal of text data.
- Brings your text into a form that is analyzable for your task.
- Transforms text into a more digestible form so that machine learning algorithms can perform better.

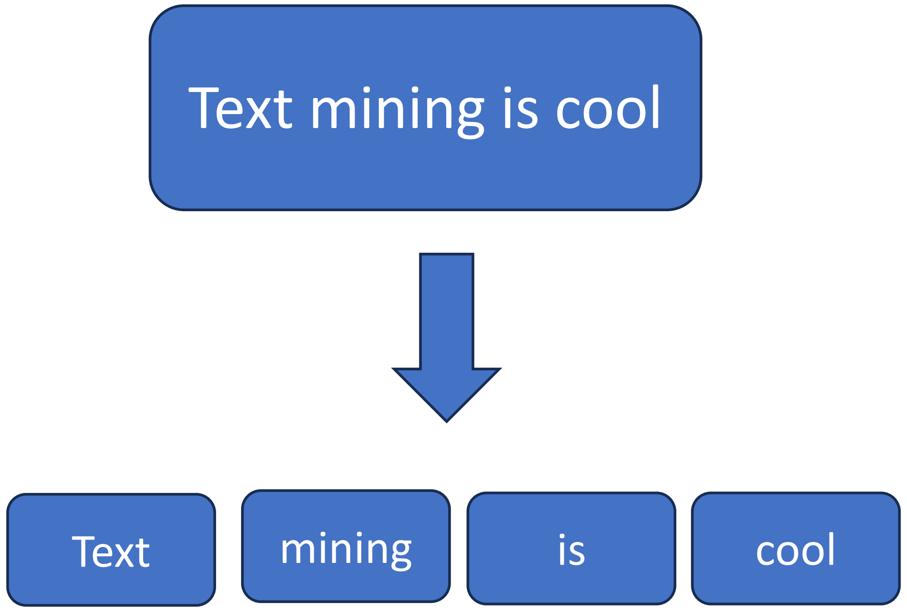
Why preprocessing?

- Why do we need to preprocess text data?

- Dimensionality
- Memory allocation
- Performance

Tokenization

- Converts sequence of text into smaller parts, known as tokens. Used for:
  - **Text preprocessing:** Most of the times the first step during preprocessing; important for the next steps of preprocessing
  - **Feature extraction:** Tokens are one of the input features to machine learning models
  - **Statistical analysis:** Necessary to compute statistics such as word frequency, document frequency



Most common pre-processing steps

- Tokenization (“text”, “ming”, “is”, “the”, “best” , “!”)
- Stemming (“lungs”→“lung”) or Lemmatization (“were”→“is”)
- Stopword removal (“text minig is best!”)

Additional steps

- Lowercasing (“Disease”→“disease”)
- Punctuation removal (“text mining is the best”)
- Number removal (“I42”→“I”)
- Spell correction (“hart”→“heart”)

Not all of pre-processing steps are appropriate at all times!

Typical steps

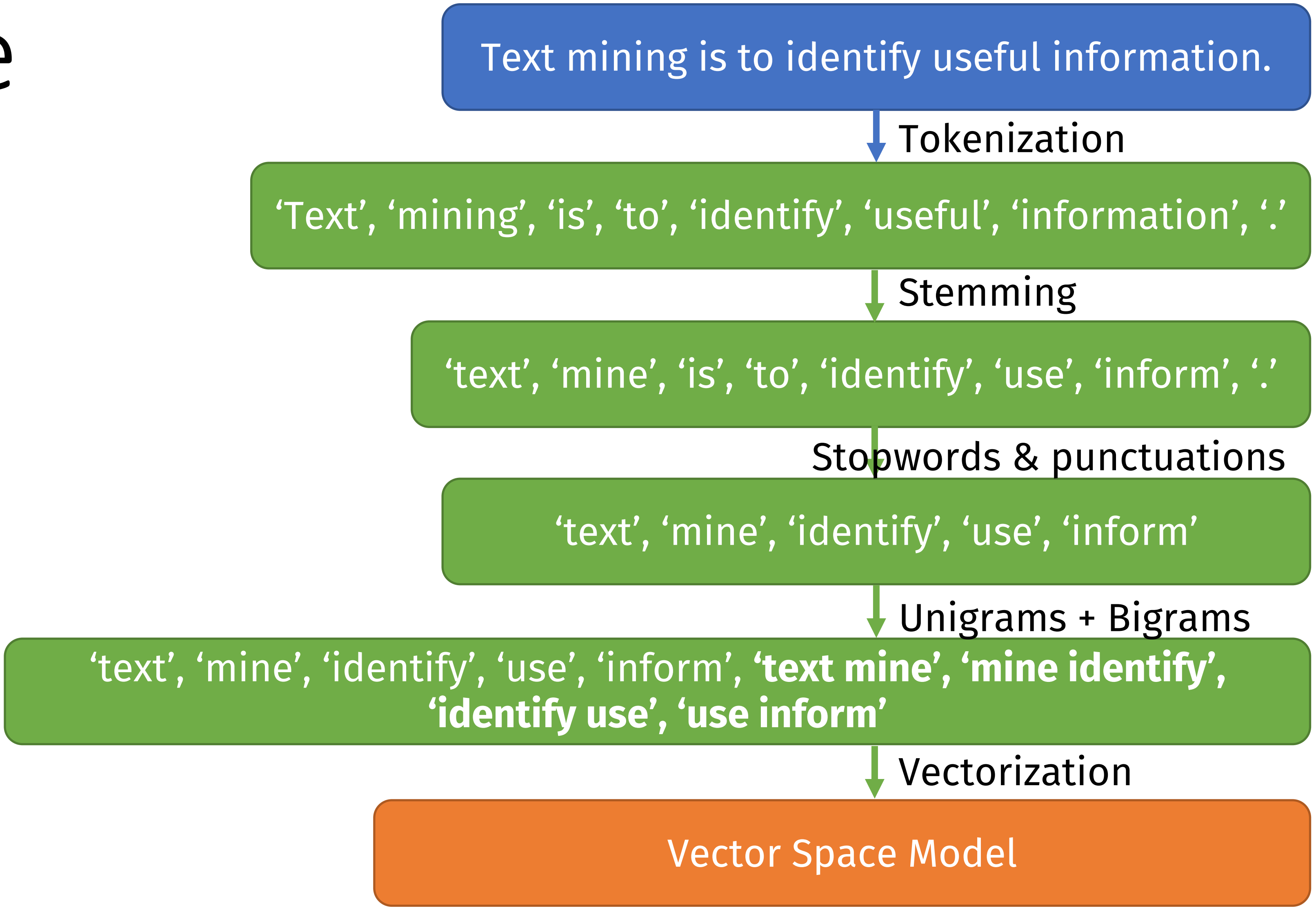
Not all of these are appropriate at all times!

Text mining is the best!!!

Text mining is the best

In tasks such as sentiment analysis or emotion detection, removing punctuation as a preprocessing step may not be appropriate

Example


Text mining is to identify useful information.
Tokenization
‘Text’, ‘mining’, ‘is’, ‘to’, ‘identify’, ‘useful’, ‘information’, ‘.’
Stemming
‘text’, ‘mine’, ‘is’, ‘to’, ‘identify’, ‘use’, ‘inform’, ‘.’
Stopwords & punctuations
‘text’, ‘mine’, ‘identify’, ‘use’, ‘inform’
Unigrams + Bigrams
‘text’, ‘mine’, ‘identify’, ‘use’, ‘inform’, **‘text mine’, ‘mine identify’, ‘identify use’, ‘use inform’**
Vectorization
Vector Space Model

Vector Space Model

Basic idea

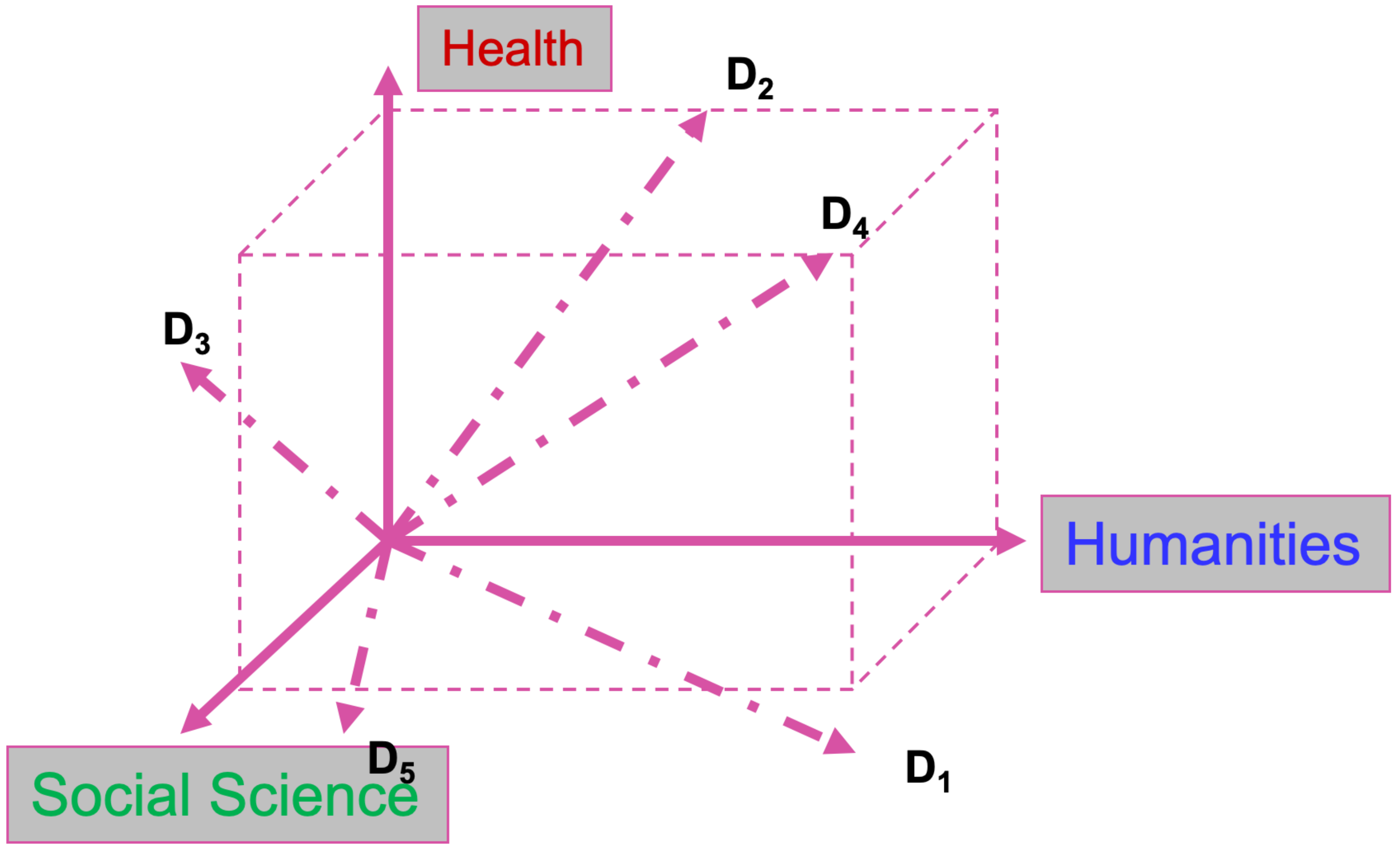
- Text is “unstructured data”
- How do we get to something structured that we can compute with?
- **Text must be represented somehow**
- Represent the text as something that makes sense to a computer -> represent it as numbers

Vector space model

- Terms / words are generic features that can be extracted from text
- Typically, terms are single words, keywords, n-grams, or phrases
- Documents are represented as vectors of terms
- Each dimension corresponds to a separate term

$$d = (w_1, \ldots, w_n)$$

An illustration

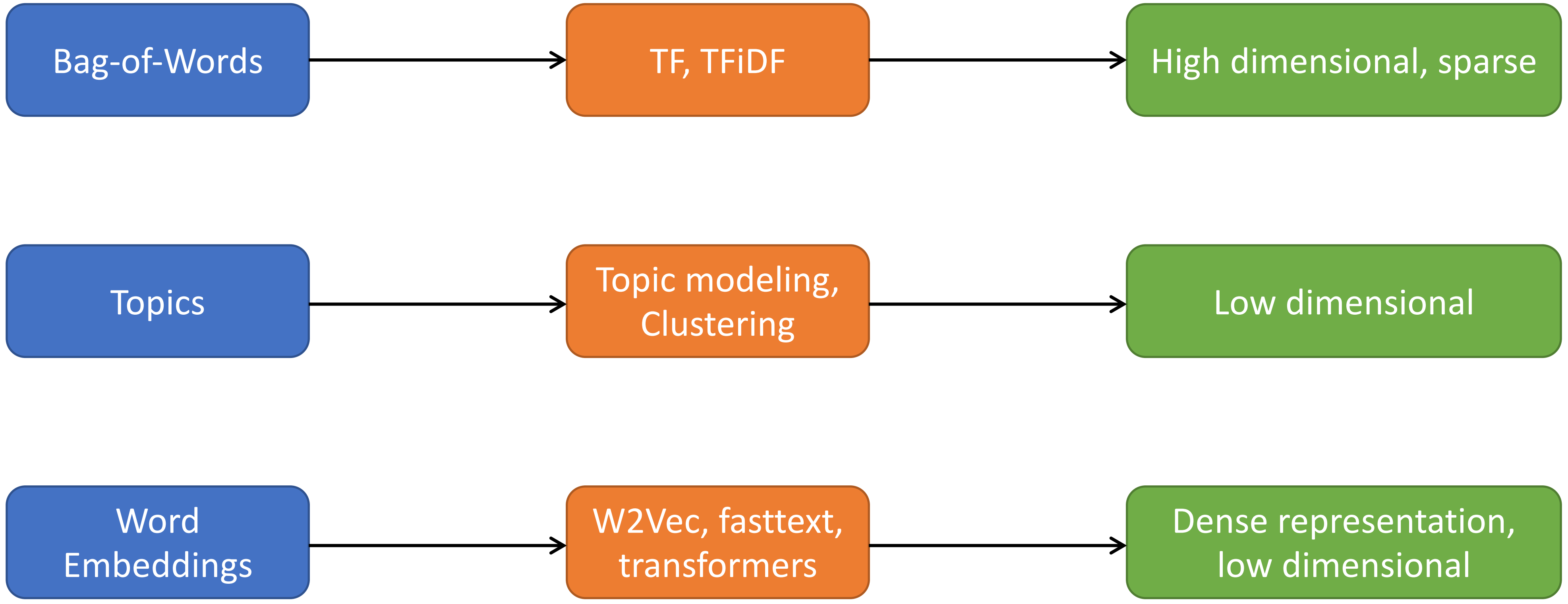


Vectorization

- The process of converting text into numbers is called **vectorization**

- Distance between the vectors in this concept space
    - Relationship among documents

VSM representations


Bag-of-Words
TF, TFiDF
High dimensional, sparse
Topics
Topic modeling, Clustering
Low dimensional
Word Embeddings
W2Vec, fasttext, transformers
Dense representation, low dimensional

Bag-of-Words

- **Terms** are words (more generally we can use n-grams)
- The unique terms in the collection of documents is the **vocabulary**
- **Weights** are number of occurrences of the terms in the document
  - Binary (absence/presence of term or 0/1)
  - Term Frequency (TF)
  - Term Frequency inverse Document Frequency (TFiDF)

Example

Doc1: Text mining is to identify useful information.

Doc2: Useful information is mined from text.

Doc3: Apple is delicious.

Vocabulary: {text, mining, is, to, identify, useful, information, mined, from, apple, delicious}

Example

Doc1: Text mining is to identify useful information.

Doc2: Useful information is mined from text.

Doc3: Apple is delicious.

Vocabulary: {text, mining, is, to, identify, useful, information, mined, from, apple, delicious}

Document-Term Matrix (DTM):

	text	mining	is	to	identify	useful	information	mined	from	apple	delicious
Doc1											
Doc2											
Doc3											

Example

Doc1: **Text mining is to identify useful information**.

Doc2: Useful information is mined from text.

Doc3: Apple is delicious.

Vocabulary: {text, mining, is, to, identify, useful, information, mined, from, apple, delicious}

Document-Term Matrix (DTM):

	text	mining	is	to	identify	useful	information	mined	from	apple	delicious
Doc1	1	1	1	1	1	1	1	0	0	0	0
Doc2											
Doc3											

Example

Doc1: Text mining is to identify useful information.

Doc2: Useful information is mined from text.

Doc3: Apple is delicious.

Vocabulary: {text, mining, is, to, identify, useful, information, mined, from, apple, delicious}

Document-Term Matrix (DTM):

	text	mining	is	to	identify	useful	information	mined	from	apple	delicious
Doc1	1	1	1	1	1	1	1	0	0	0	0
Doc2	1	0	1	0	0	1	1	1	1	0	0
Doc3	0	0	1	0	0	0	0	0	0	1	1

Question

How is the DTM going to change if I first apply stop-words removal?

What about the number dimensions?

	text	mining	is	to	identify	useful	information	mined	from	apple	delicious
Doc1	1	1	1	1	1	1	1	0	0	0	0
Doc2	1	0	1	0	0	1	1	1	1	0	0
Doc3	0	0	1	0	0	0	0	0	0	1	1

	text	mining	identify	useful	information	mined	apple	delicious
Doc1	1	1	1	1	1	0	0	0
Doc2	1	0	0	1	1	1	0	0
Doc3	0	0	0	0	0	0	1	1

Limitation of BoW

- **Problem with Frequency**: BoW treats all words in the document equally regardless of their importance or relevance to the document's meaning
- Scenario: Collection of financial documents that "profit" or ”stock" appears multiple times in every document. BoW will give high frequencies to those terms, even if they are not necessarily indicative of the document's content (since they appear in all documents)
- TF-IDF -> less weight to common terms in the collection and more weight to terms that are less frequent across documents

TFiDF

- A term is more discriminative if it occurs a lot but only in fewer documents

TFiDF weight: $w_{d,t} = TF_{d,t} \cdot IDF_t$

TF and iDF

- Term frequency: Let $n_{d,t}$ denote the number of times the t-th term appears in the d-th document.

$$TF_{d,t} = \frac{n_{d,t}}{\sum_i n_{d,i}}$$

- Inverse document frequency: Let N denote the number of documents and N_t denote the number of documents containing the t-th term.

$$IDF_t = log_{10}(\frac{N}{N_t})$$

Example

Doc1: Text mining is to identify useful information. (length = 7)

Doc2: Useful information is mined from text. (length = 6)

Doc3: Apple is delicious. (length = 3)

$TF_{doc1,\ text} = 1/7$
$TF_{doc2,\ text} = 1/6$
$TF_{doc3,\ text} = 0/3$

$IDF_{text} = \log(N/N_t) = \log(3/2) = 0.17$

	text	mining	…
Doc1	?		
Doc2	?		
Doc3	?		

Example

Doc1: Text mining is to identify useful information. (length = 7)

Doc2: Useful information is mined from text. (length = 6)

Doc3: Apple is delicious. (length = 3)

$TF_{doc1,\ text} = 1/7$
$TF_{doc2,\ text} = 1/6$
$TF_{doc3,\ text} = 0/3$

$IDF_{text} = \log(N/N_t) = \log(3/2) = 0.17$

$w_{doc1,\ text} = 1/7 * 0.17 = 0.024$
$w_{doc2,\ text} = 1/6 * 0.17 = 0.028$
$w_{doc3,\ text} = 0/3 * 0.17 = 0$

	text	mining	…
Doc1	0.024		
Doc2	0.028		
Doc3	0		

Example

Doc1: Text mining is to identify useful information. (length = 7)

Doc2: Useful information is mined from text. (length = 6)

Doc3: Apple is delicious. (length = 3)

IDF_{is} = ?

Example

Doc1: Text mining is to identify useful information. (length = 7)

Doc2: Useful information is mined from text. (length = 6)

Doc3: Apple is delicious. (length = 3)

$$IDF_t = log_{10}(\frac{N}{N_t})$$

$\mathrm{IDF}_{\mathrm{is}} = \log(3/3) = 0$

Terminology

- **Corpus/Collection:** is a large and structured set of texts/documents
- **Vocabulary**: the set of words that appear in the corpus
- **Unstructured text:** information that either does not have a pre-defined data model or is not organized in a pre-defined manner.
- **Tokenizing:** process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens (see also lexical analysis)
- **Term document (or document term) matrix:** is a mathematical matrix that describes the frequency of terms that occur in a collection of documents
- **Stop words:** words which are filtered out before or after processing of natural language data (text)
- **Stemming:** the process for reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form

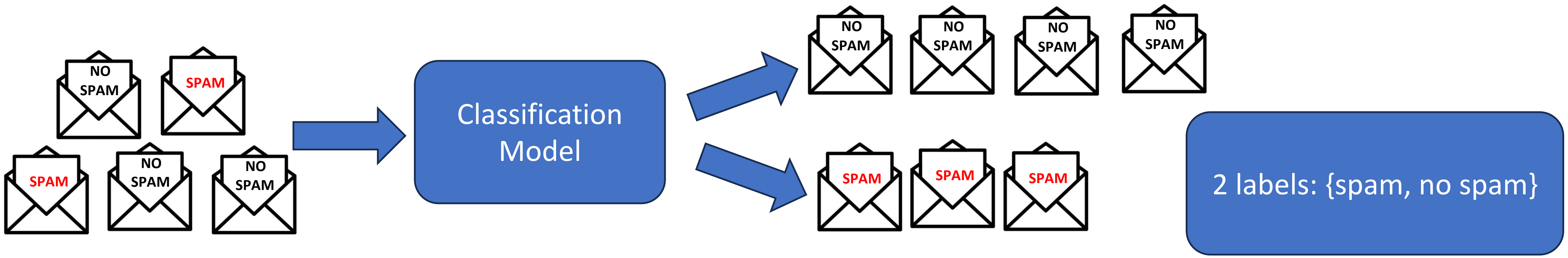
10-minute break

Text Classification

Text classification

Text classification is the **task of assigning a label or class to a given text**

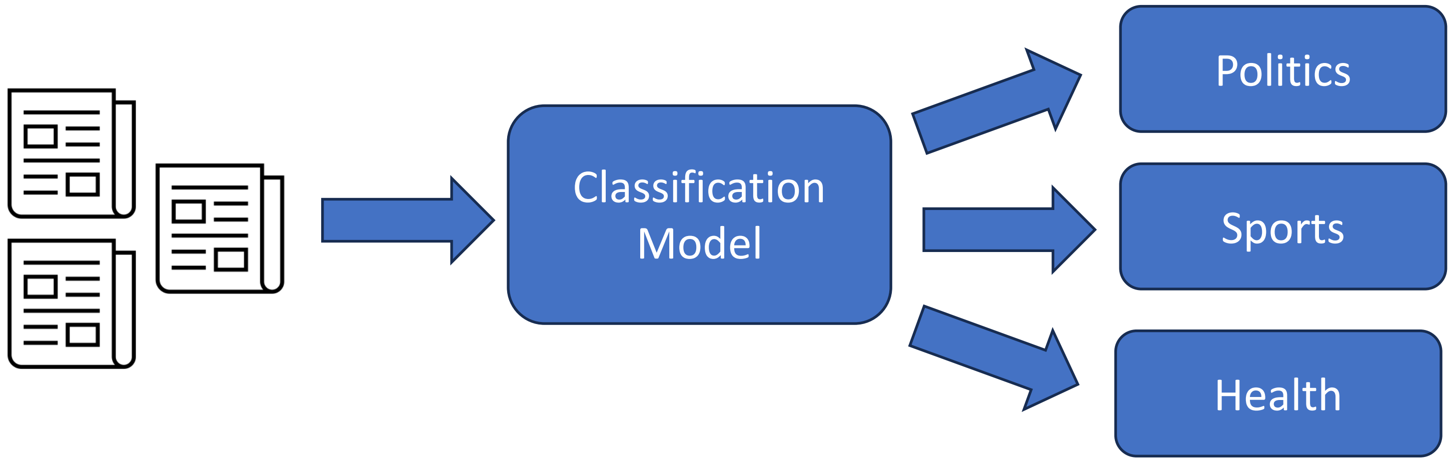
Example: Classify emails as spam or not spam

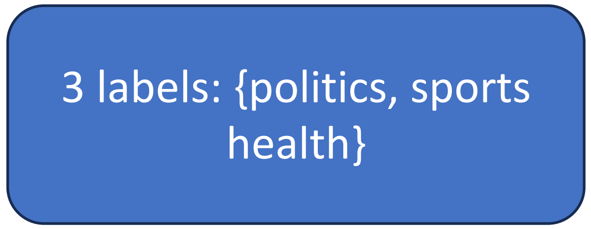


Text classification

Text classification is the **task of assigning a label or class to a given text**

Example: Assign documents to predefined categories





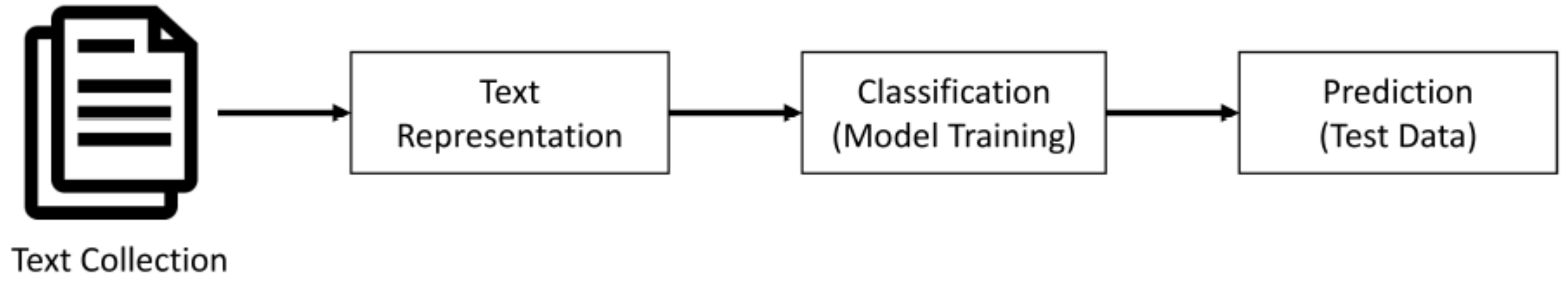
Text classification as supervised learning

- **Supervised learning**: Learning a function that maps an input to an output based on example input-output pairs.
  - infer a function from labeled training data
  - use the inferred function to label new instances

- Human experts annotate a set of text data
  - Training set

Document	Class
note1…..	Not cancer
note2….	Not cancer
note3….	cancer
…	…

Simple pipeline



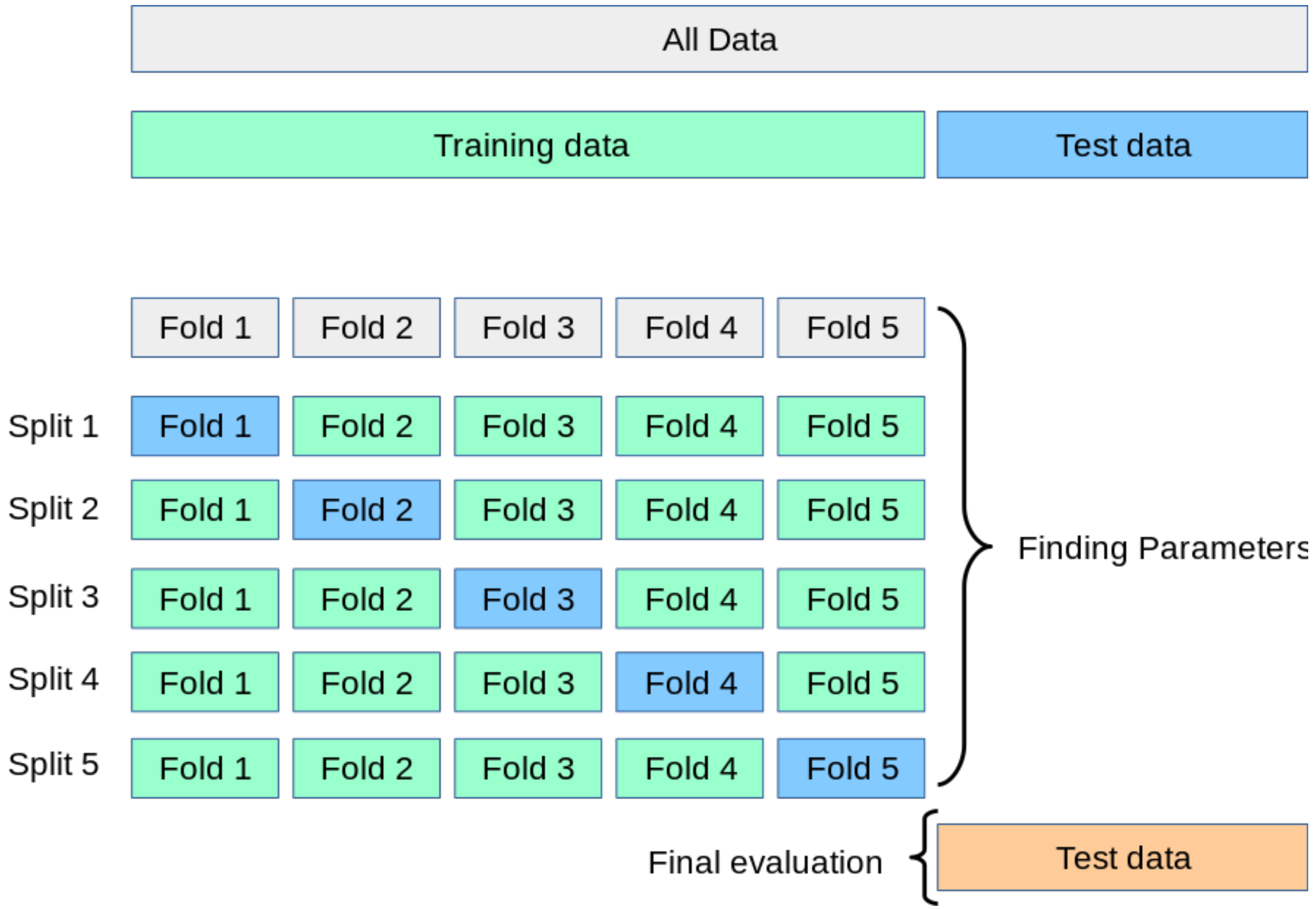
Preparing data

- Text preprocessing
- Data splitting
    - Training
    - Validation (development)
        - to tune the hyperparameters
    - Test
- Text representation

Algorithms

- Naïve Bayes
- Logistic regression
- Support vector machines
- K-nearest neighbors
- Neural networks
- Deep learning

K-fold cross validation



https://scikit-learn.org/stable/modules/cross_validation.html

Hand-coded rules

- Rules based on combinations of words or other features
- Accuracy can be high: If rules carefully refined by expert
- But building and maintaining these rules is expensive
- Data/Domain specifics

Example:

```
• If "congratulations" & "prize" in email:
    email is spam
```

Word Embeddings

Word as vectors

The vector representations should:

- capture semantics
  - similar words should be close to each other in the vector space
  - relation between two vectors should reflect the relationship between the two words
- be efficient (vectors with fewer dimensions are easier to work with)
- be interpretable

Word representations

How can we represent the meaning of words?

So, we can ask:

- How similar is heart to lung, or finance to economy?
- How similar is document A to document B?

One hot encoding

- Map each word to a unique identifier
- e.g. cat (3) and dog (5).
  → Vector representation: all zeros, except 1 at the ID

cat	**0**	**0**	**0**	**0**	**0**	**1**
dog	**0**	**0**	**0**	**0**	**1**	**0**
train	**0**	**0**	**0**	**1**	**0**	**0**

But:
Even related words have distinct vectors!
High number of dimensions
Order of words does not matter

Distributional hypothesis

…how much better a sense of smell of a **dog** is compared to that of humans…
Jacobson's organ is found in the roofs of **dogs**' mouths that helps them smell
…how far a **dog** can smell a scent depends on a lot of factors…

Distributional hypothesis: Words that occur in similar contexts tend to have similar meanings.

You shall know a word by the company it keeps.
(Firth, J. R. 1957:11)

Word embeddings

- Vectors are short; typically 50-1024 dimensions
- Vectors are dense (mostly non-zero values)
- Very effective for many NLP tasks
- Individual dimensions are less interpretable

cat	0,52	-0,23	....	0,15
dog	0,45	-0,34	....	0,05

Properties of word embeddings

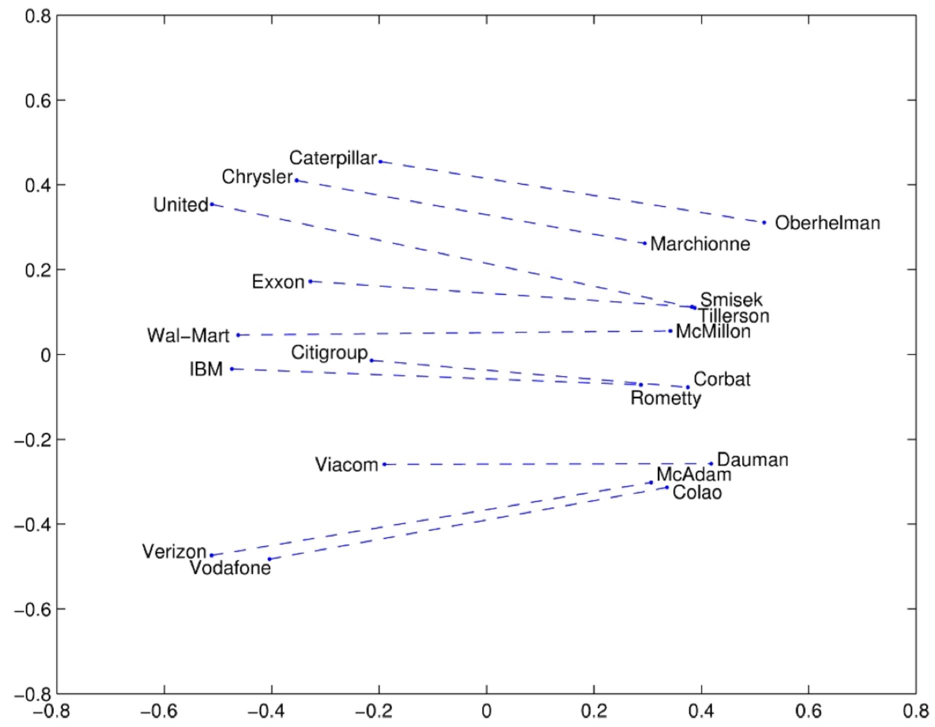


Figure: company - ceo

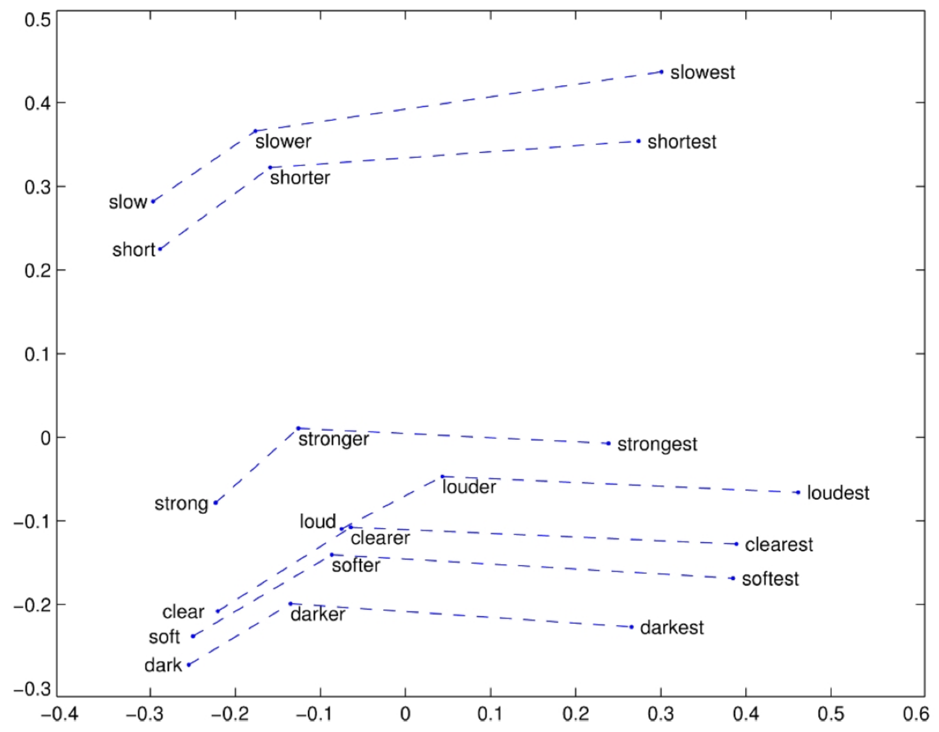


Figure: comparative - superlative

Source: https://nlp.stanford.edu/projects/glove/

Properties of word embeddings

- We can look at analogies in the vector space, for example:
- king - man + woman ≈ queen

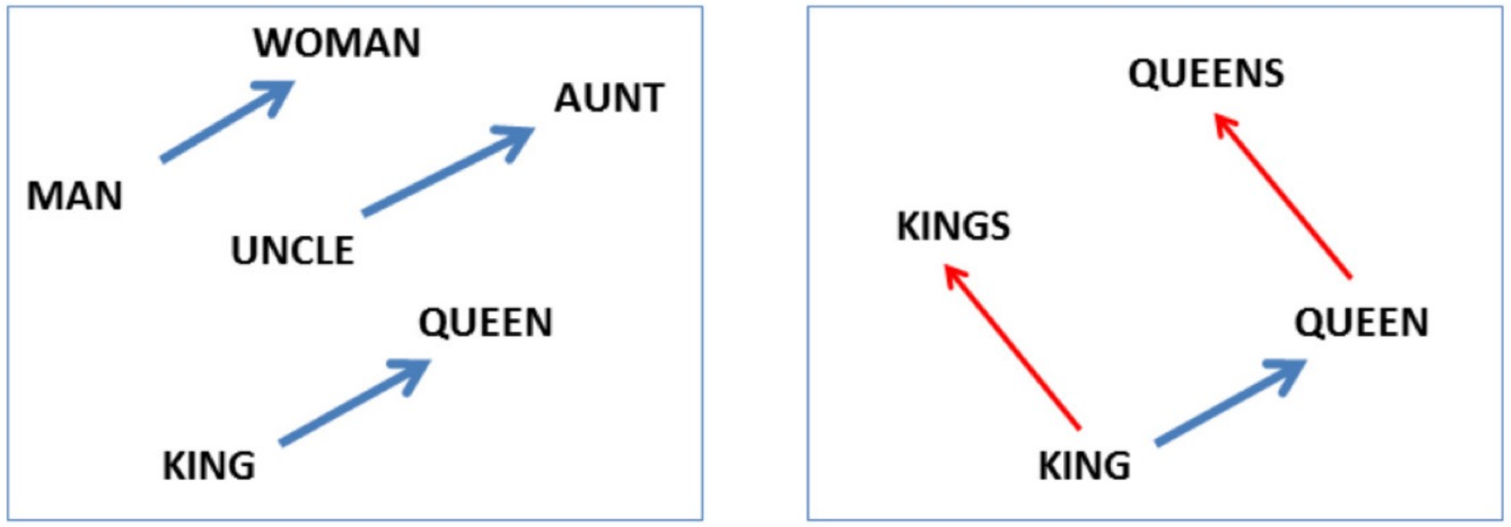


Figure: Figure 2 from Linguistic Regularities in Continuous Space Word Representations, Mikolov et al. NAACL 2013

Training data for word embeddings

- Use **text itself** as training data for the model!
  - A form of self-supervision.

- Train a **classifier** (neural network, logistic regression, or SVM, etc.) to predict the next word given previous words.

Word2Vec

- Popular embedding method
- Very fast to train

- **Can you find 5 nearest words to “finance”? (use Word2Vec all)**
- **https://projector.tensorflow.org/**

Conclusion

- The basic problem of text mining is that text is not a neat data set
- The solution to this problem is preprocessing and representation

Conclusion

- VSM: BOW vs Word Embedding
- BOW leads to high dimensional vector space, sparse vectors with many 0s
- Embeddings:
  - Key idea is the “distributional hypothesis” -> “you will know a word by the company it keeps”;
  - Maps each word into a low-dimensional vector space; dense vectors
  - Or, in other words: assign a bunch of numbers to each word, in such a way that “similar” words are closer together

Next

- Practical on Thursday
- Preparation: make sure to have finished the first part.

- Next week: Network analysis (on location, 11.00-12.45)

Have a nice day!

Questions?