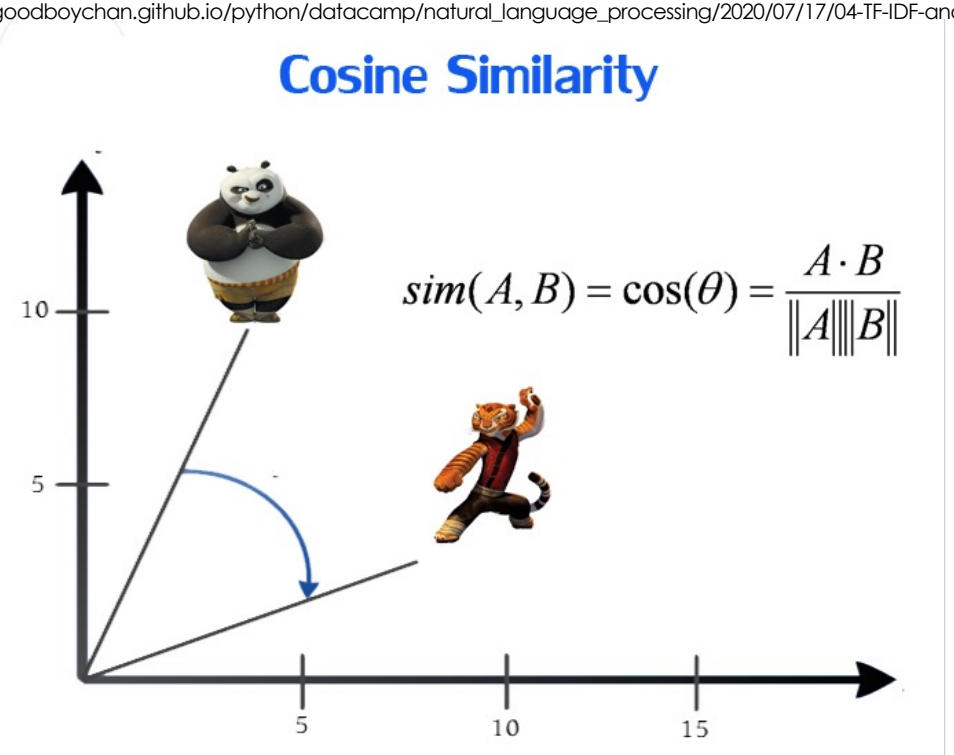
Image from https://goodboychan.github.io/python/datacamp/natural_language_processing/2020/07/17/04-TF-IDF-and-similarity-scores.html



LANGUAGE AS VECTORS & FEATURE-BASED MACHINE LEARNING APPROACH

Luísa Coheur

TÉCNICO LISBOA

inesc id lisboa

OVERVIEW

- Learning objectives
- Topics
  - Language Representation
    - Language as vectors
    - Documents as vectors
    - Words as vectors
    - Sentences as vectors
  - Feature Engineering Machine Learning Approach
    - Concept and examples of interesting features
    - Some widely used algorithms
    - End-to-end example
- Key takeaways
- Suggested readings

LEARNING OBJECTIVES

LEARNING OBJECTIVES

- After this class, students:
  - Should understand how language can be represented with vectors (sparse for now)
  - Describe different ways to create those vectors (including by using tf-idf)
  - Calculate the distance between given vectors by applying the cosine-similarity, including between sentences
  - Should be able to understand and define "feature engineering" machine learning and apply the Naïve Bayes algorithm
  - Reflect about how dangerous it can be to take decisions directly based on AI results

TOPICS

OVERVIEW

- Learning objectives
- Topics
  - Language Representation
    - Language as vectors
    - Documents as vectors
    - Words as vectors
    - Sentences as vectors
  - Feature Engineering Machine Learning Approach
    - Concept and examples of interesting features
    - Some widely used algorithms
    - End-to-end example
- Key takeaways
- Suggested readings

SNEAK PEEK

- Now we will see how to represent natural language as…

SNEAK PEEK

- We will see how to represent natural language as

VECTORS

LANGUAGE AS VECTORS (SPARSE)

- Distributional Semantics
  - Harris (1954)
    - "If A and B have almost identical environments […] we say that they are synonyms"
  - J.R.Firth (1957)
    - "You shall know a word by the company it keeps"

LANGUAGE AS VECTORS (SPARSE)

- Example from Jurafsky:

tasty tnassiorc
greasy tnassiorc
tnassiorc with butter
tnassiorc for breakfast

FOOD

LANGUAGE AS VECTORS (SPARSE)

- The meaning of a word is given by the set of contexts in which it occurs
- Principles:
  - Meanings are locations in a semantic space
  - Words with similar distributional properties have similar meanings

- But, before moving into "Words as vectors", let us check "Documents as vectors"

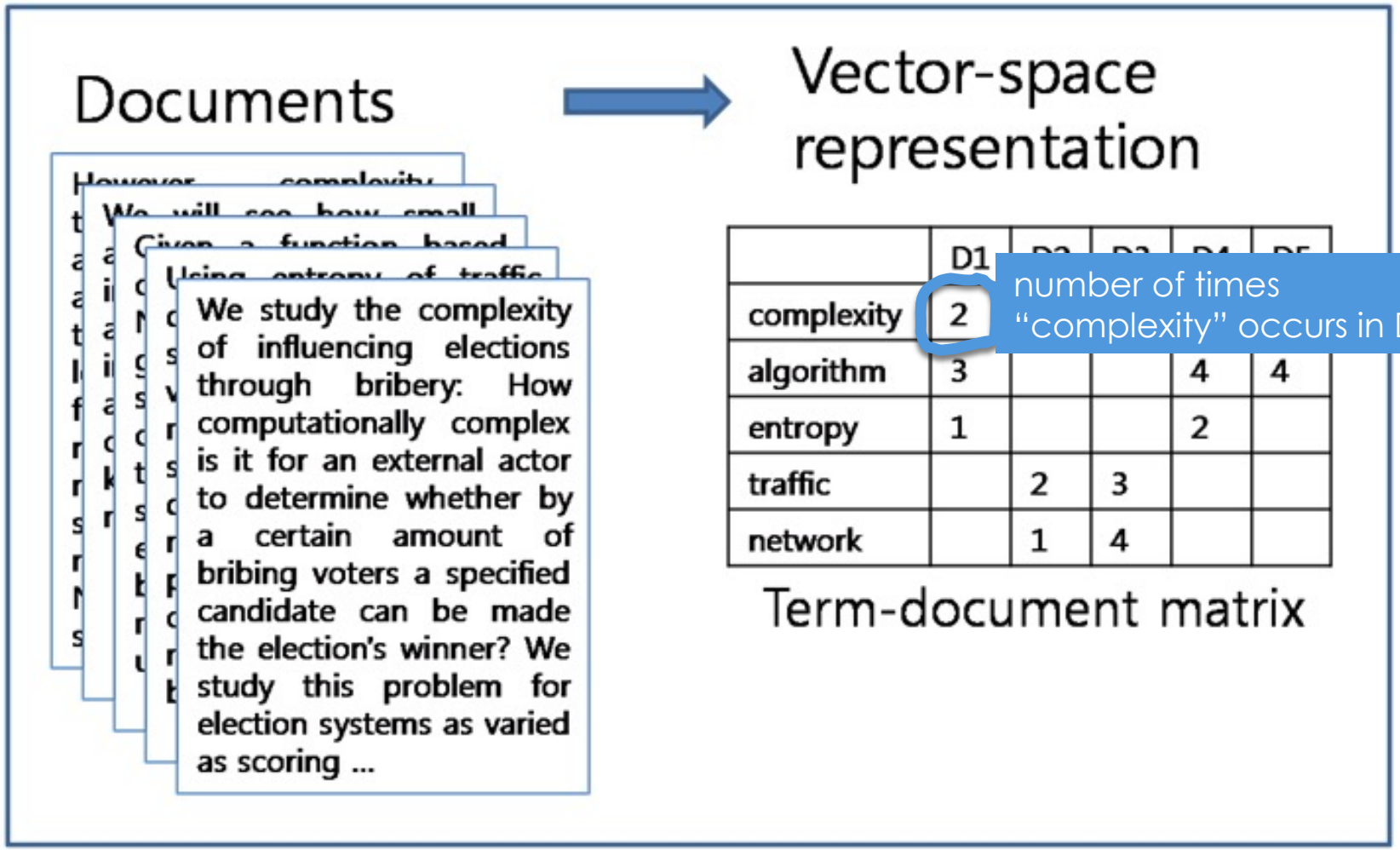
OVERVIEW

- Learning objectives
- Topics
  - Language Representation:
    - Language as vectors
    - Documents as vectors
    - Words as vectors
    - Sentences as vectors
  - Feature Engineering Machine Learning Approach
    - Concept and examples of interesting features
    - Some widely used algorithms
    - End-to-end example
- Key takeaways
- Suggested readings

DOCUMENTS AS VECTORS

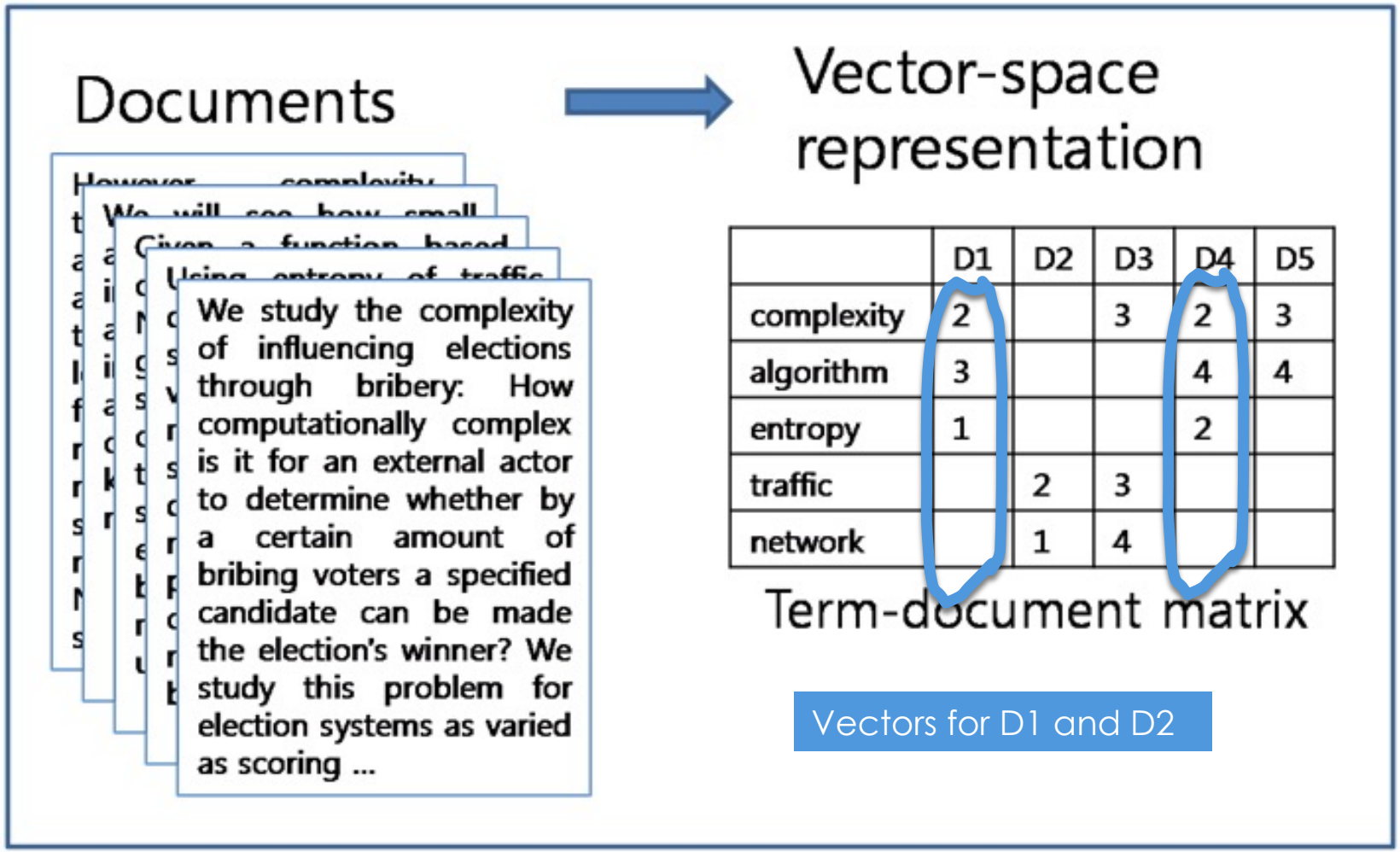
- Documents as vectors main idea (from Information Retrieval (IR)):
  - A vector space model is a model for representing a text document as a vector

DOCUMENTS AS VECTORS



Example from SpeedLab

DOCUMENTS AS VECTORS
MAIN IDEA



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Example from SpeedLab

DOCUMENTS AS VECTORS MAIN IDEA

- Vector space model
  - Being given a collection of documents, build a term-document matrix in which:
    - Each row represents a term in the vocabulary
    - Each column represents a document

But what do we put inside each cell of that matrix?

DOCUMENTS AS VECTORS
BUILDING THE MATRIX

- Binary
  - the elements of the vectors are either 1 or 0, where 1 indicates that the word has occurred in the document, and 0 that it has not
- Raw count
  - the elements are the raw frequency of occurrence of the word in the document

	Document 1	Document 2	Document 3
bank	0	0	4
bass	2	4	0
commercial	0	2	2
cream	2	0	0
guitar	1	0	0
fishermen	0	3	0
money	0	1	2

DOCUMENTS AS VECTORS
BUILDING THE MATRIX

- TF-IDF (the famous TF-IDF!!!!!) combines the term frequency (TF) with the inverse document frequency (IDF):
  - TF-IDF(t, d, D) = TF(t,d) × IDF(t, D)
  - in which:
    - t term,
    - d document,
    - D collection of documents

DOCUMENTS AS VECTORS
BUILDING THE MATRIX

- TF (Term Frequency):
  - Tf(t, d) = freq(t, d) = term t frequency in document d (it can be just the count or the relative frequency in the document; next we will use the second possibility)
- IDF (Inverse Document Frequency):
  - IDF(t, D) = log (|D| / |{d ∈ D: t ∈ d}|), in which:
    - |D| is the number of documents
    - |{d ∈ D: t ∈ d}| is the number of documents that contain the term t
    - log is in base e
    - (notice that there are other possible formulas)

ACTIVE LEARNING MOMENT: THINK-PAIR-SHARE

EXERCISE

- Let:
  - t = gato (cat)
  - d129 = document with 100 words, in which "gato" occurs 3 times
  - |D| = collection of 10.000.000 documents; "gato" appears in 1000 documents
- Considering:
  - Tf(t, d) = freq(t, d) = term t relative frequency in document d
  - IDF(t, D) = log (|D| / |{d ∈ D: t ∈ d}|)
- Then:
  - TF-IDF(gato, d129, D) = TF(gato, d129) x IDF(gato, D) =
  - = 0,03 x log (|D| /1000) = 0,03 x 9,2

DOCUMENTS AS VECTORS
BUILDING THE MATRIX

- Raw counts:

	Document 1	Document 2	Document 3
bank	0	0	4
bass	2	4	0
commercial	0	2	2
cream	2	0	0
guitar	1	0	0
fishermen	0	3	0
money	0	1	2

- Tf-idf

	Document 1	Document 2	Document 3
bank	0	0	0.76
bass	0.23	0.28	0
commercial	0	0.23	0.23
cream	0.62	0	0
guitar	0.48	0	0
fisherman	0	0.70	0
money	0	0.18	0.23

Note: just an example, doesn't mean that these counts are accurate

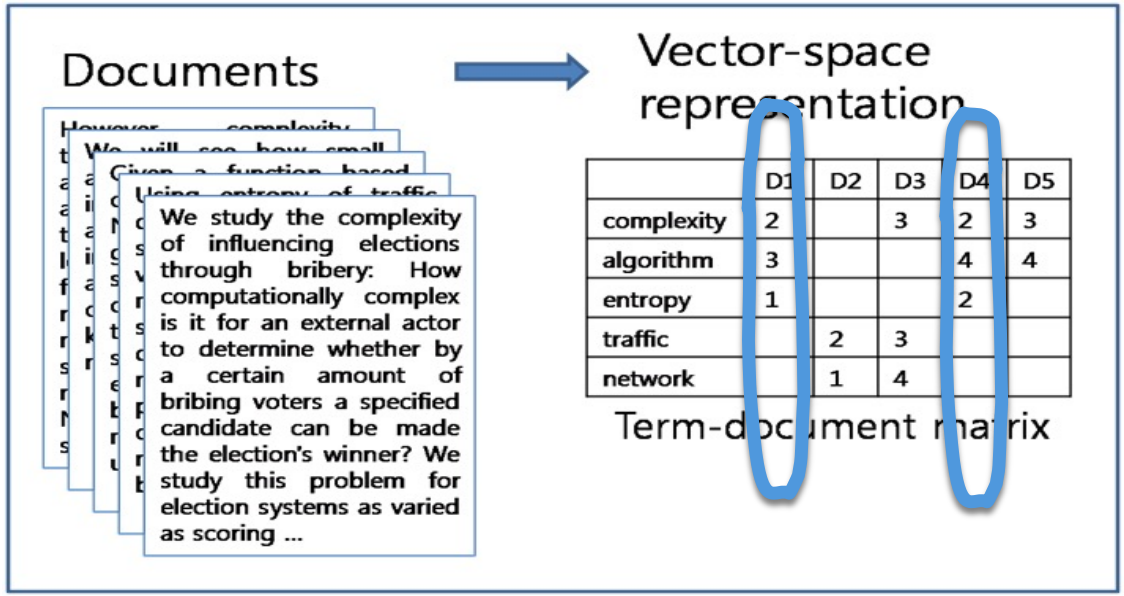
OVERVIEW

- Learning objectives
- Topics
  - Language Representation:
    - Language as vectors
    - Documents as vectors
    - Words as vectors
    - Sentences as vectors
  - Feature Engineering Machine Learning Approach
    - Concept and examples of interesting features
    - Some widely used algorithms
    - End-to-end example
- Key takeaways
- Suggested readings

WORDS AS VECTORS

- A word can have a dual role in word space:
  - Each word can be a dimension, an axis of the space (=> documents as vectors)
  - Each word can be a vector in the space in which documents are the axis (=> words as vectors)

PREVIOUSLY:

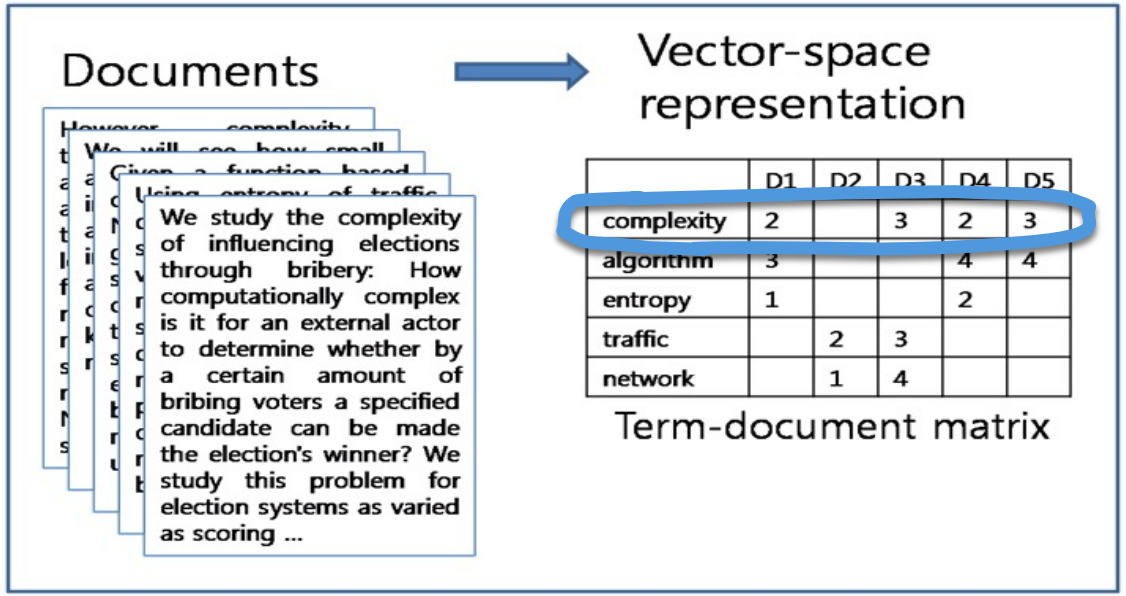


	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

WORDS AS VECTORS

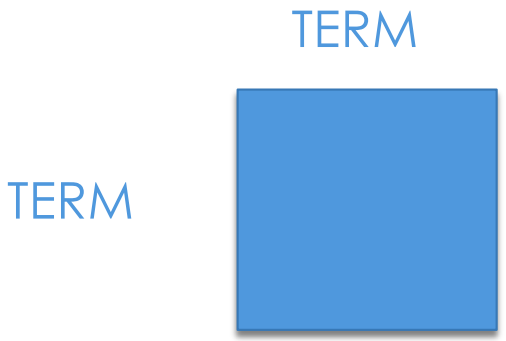
- A word can have a dual role in word space:
  - Each word can be a dimension, an axis of the space (=> documents as vectors)
  - Each word can be a vector in the space in which documents are the axis (=> words as vectors)

PREVIOUSLY:



WORDS AS VECTORS

- But we can take this idea a little further:
  - Instead of a term-document matrix we can have term-term matrix (also called word-word matrix or term-context matrix) by considering the words' context



WORDS AS VECTORS
BUILDING THE MATRIX

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

Window (context) = 1

The resulting counts matrix will then be:

"Like" twice in the context of "I"

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Figure from: https://towardsdatascience.com/word-to-vectors-natural-language-processing-b253dd0b0817

WORDS AS VECTORS
BUILDING THE MATRIX

: Center Word

: Context Word

c=1 The cute cat jumps over the lazy dog.

c=2 The cute cat jumps over the lazy dog.

https://docs.chainer.org/en/stable/examples/word2vec.html

WORDS AS VECTORS
BUILDING THE MATRIX

- |V| = the size of the vocabulary
  - |V| usually between 10.000 and 50.000
    - most frequent words are used; more than that is not helpful
  - size of the window between 1 and 8
    - => total context between 3 and 17

WORDS AS VECTORS
BUILDING THE MATRIX

- But… simple frequency is still not the best measure
  - words such as “the”, “it”, “they” are not discriminative
  - the best weighting or measure of association between two words should tell us more than a chance of co-ocurrence
  - Use:
    - pointwise mutual information (Tf-idf is usually not used for word/word similarity)
    - …

WORDS AS VECTORS
BUILDING THE MATRIX

- Up to now: very high-dimensional space
  - Words represented as sparse vectors
    - Long vectors: length = |V|, 10.000-50.000
    - Sparse vectors: most entries = 0
      - => need for dimensionality reduction
- Next week we will see that words can be represented as dense vectors
    - Short vectors: 50-1.000 dimensions
    - Dense vectors: most entries ≠ 0

SIMILARITY/DISTANCE METRICS AGAIN

- How to measure similarity between two words or two documents when they are represented as vectors?
  - Remember that distances can be transformed into similarity measures (low distances => similar words/documents)

	Document 1	Document 2	Document 3
bank	0	0	0.76
bass	0.23	0.28	0
commercial	0	0.23	0.23
cream	0.62	0	0
guitar	0.48	0	0
fisherman	0	0.70	0
money	0	0.18	0.23

SIMILARITY/DISTANCE METRICS AGAIN

- If documents are vectors, distances can also be easily calculated:

SIMILARITY/DISTANCE METRICS AGAIN

- If documents are vectors, distances can also be easily calculated:
  - Use the angle!!
    - small angles mean similar documents
    - maximal similarity when the angle is zero

SIMILARITY/DISTANCE METRICS AGAIN

- Cosine similarity (cosine is 1 when the angle between two vectors is 0 and smaller for every other angle)

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

SIMILARITY/DISTANCE METRICS AGAIN

	abandon	⋯	plane	⋯	kill	⋯	survive	⋯	people
$\overrightarrow{\text{DOC 1}} =$	(0.11	⋯	1.00	⋯	0.03	⋯	0.23	⋯	0.65)
⋮									
$\overrightarrow{\text{DOC 2}} =$	(0.00	⋯	0.01	⋯	1.00	⋯	0.11	⋯	0.09)

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

$$sim(\text{DOC1 \& DOC2}) = \frac{0.11 * 0.00 + \cdots + 0.65 * 0.09}{\sqrt{0.11^2 + \cdots + 0.65^2} * \sqrt{0.00^2 + \cdots + 0.09^2}} \in [0,1]$$

OVERVIEW

- Learning objectives
- Topics
  - Language Representation:
    - Language as vectors
    - Documents as vectors
    - Words as vectors
    - Sentences as vectors
  - Feature Engineering Machine Learning Approach
    - Concept and examples of interesting features
    - Some widely used algorithms
    - End-to-end example
- Key takeaways
- Suggested readings

CASE STUDY

- With these metrics applied to sentences you can build a retrieval-based chatbot!!!!

- Knowledge base
  - Q_1:A_1
  - Q_2:A_2
  - Q_3:A_3
  - …

- User: Q

- How: find the distance between Q and Q_i. Return the A_j associated with the Q_j that has the highest similarity (or smaller distance)

CASE STUDY

- Example:
  - "TRAINING" CORPUS (knowledge base of FAQs):
    - Q: "Writer of LOTR" A: Tolkien
  - "TEST": Who wrote LOTR?
- Simple solution with Jaccard(s, t) = $|s \cap t| / |s \cup t|$
  - and some pre-processing that might help… or not:
    - Direct: Jaccard(Writer of LOTR, Who wrote LOTR?)
    - Lowercasing: Jaccard(writer of lotr, who wrote lotr?)
    - Stop-words + Punctuation removal: Jaccard(writer lotr, who wrote lotr)
    - Lemmatization: Jaccard(write lotr, who write lotr)
- Now you can use the cos-similarity instead of Jaccard
  - Which vectors? Should we return Tolkien?

CASE STUDY

- This was in 2017
  - and due to a chatbot based in similarity measures!!!!

Generated by DALL-E

OVERVIEW

- Learning objectives
- Topics
  - Language Representation:
    - Language as vectors
    - Documents as vectors
    - Words as vectors
    - Sentences as vectors
  - Feature Engineering Machine Learning Approach
    - Concept and examples of interesting features
    - Some widely used algorithms
    - End-to-end example
- Key takeaways
- Suggested readings

FEATURE ENGINEERING

- Feature Engineering is a machine learning approach in which domain knowledge is used to extract features from raw data. These features are used to improve the performance of machine learning algorithms
  - Some features make sense, others do not!

ACTIVE LEARNING MOMENT: THINK-PAIR-SHARE

EXERCISE

- Consider the task of classifying a proper noun, in Portuguese, according with its gender
  - That is, being given a proper name, decide if it is feminine or masculine
    - Example:
      - Luísa, Manuel, João, Inês, …
- Question: which are the features that will help you to decide (if a given noun is feminine or masculine)?

EXERCISE

- Two classes:
  - fem (feminine) and masc (masculine)
- Build a corpus:
  - The X most frequent masculine (masc) and feminine (fem) proper names
- Example:
  - my_data = [('Maria', 'fem'), ('Matilde ', 'fem'), ('Carolina', 'fem'), ('Beatriz', 'fem'), ('Alícia', 'fem'), ('Nicole', 'fem'), ('João ', 'masc'), ('Rodrigo ', 'masc'),('Martim', 'masc'), ('Francisco', 'masc'), ...

EXERCISE

- Some features make sense:
  - The last character might give you an important clue
    - Example:
      - Maria, Luísa, Joana, Ana vs. Miguel, João, Pedro
- Some features don't:
  - In this scenario, the size of the proper noun is irrelevant
    - Example:
      - Ana (size 3) vs. Rui (size 3)
      - Inês (size 4) vs. João (size 4)
      - Antonieta (size 9) vs. Godofredo (size 9)
- Anyway, you will fail! 😟
  - Examples:
    - Luca, Garcia, Quaresma …

EXERCISE

	First char	Last char	Num a's	…	Num z's	Has a's?	…	Has z's?	Gender
Maria	m	a	2		0	1		0	fem
Pedro	p	o	0		0	0		0	mes
Inês	i	s	0		0	0		0	fem
Miguel	m	l	0		0	0		0	mas
Jorge	j	e	0		0	0		0	mas
Fernando	f	o	1		0	1		0	mas

ACTIVE LEARNING MOMENT: THINK-PAIR-SHARE

EXERCISE

- Detect interactional style (paper from JURAFSKY)
  - Given speech and text from a conversation can we tell if a speaker is
    - Awkward?
    - Flirtatious?
    - Friendly?
  - Dataset
    - 1000 4-minute “speed-dates”
    - Each subject rated their partner for these styles
    - Example:

EXERCISE

- Which features should we consider?

DETECT INTERACTIONAL STYLE

- Prosodic and Lexical features:
  - Pitch min, max, mean, range, …
  - Duration of turn
  - Number of words
  - Use of past tense
  - Use of “you”
  - Use of “we”
  - …

DETECT INTERACTIONAL STYLE

- Discourse Features
  - # of Backchannels
    - Examples:
      - Uh-huh. Yeah. Right. Oh, Okay
  - # of Appreciations
    - Examples:
      - Wow. That's true. Oh, great!
  - # of Questions
  - Amount of Laughter
  - Total number of turns
  - # of disfluencies
  - Amount of overlapped speech
  - …

BY THE WAY…

- Disfluencies:
  - "false starts"
    - words and sentences that are cut off
    - phrases that are restarted or repeated
    - repeated syllables
  - "fillers":
    - non-lexical utterances (ex: huh, uh, erm, um, and hmm)
    - semiarticulate utterances (ex: well, so, I mean, and like)
    - "repaired" utterances: correcting mispronunciations (for instance)

Adapted from Wikipedia

DETECT INTERACTIONAL STYLE

- Results presented by Jurafsky:
  - Good predictors, across both genders:
    - Awkward speaker: slow, lower pitched, stilted talk
    - Flirtatious speaker: greater laughter, more questions, and referring to the past
    - Friendly speaker: greater laughter
  - Gender differences
    - Flirtation:
      - Women raise pitch,
      - Men drop pitch
      - …

OVERVIEW

- Learning objectives
- Topics
  - Language Representation:
    - Language as vectors
    - Documents as vectors
    - Words as vectors
    - Sentences as vectors
  - Feature Engineering Machine Learning Approach
    - Concept and examples of interesting features
    - Some widely used algorithms
    - End-to-end example
- Key takeaways
- Suggested readings

SOME WELL KNOWN ALGORITHMS

- Classification tasks:
  - Decision trees
  - Support Vector Machines
  - Naïve Bayes
  - ...
- Sequence Prediction tasks:
  - CRFs
  - ...

NAÏVE BAYES (NB) CLASSIFIER

- Family of algorithms that, to predict the category of a given sample:
  - apply Bayes theorem, and
  - assume that every feature is independent of the others
  - Being
    - $C = \{c_1, \ldots, c_n\}$ // categories
    - $x = (E_1, \ldots, E_m)$. // evidences/features
  - Then
    - argmax c_i in C $P(c_i \mid x) \cong$
      - argmax c_i in C $P(E_1 \mid c_i) * \ldots * P(E_m \mid c_i) * P(c_i)$
  - So, find
    - **$P(c_i \mid x) = P(E_1 \mid c_i) * \ldots * P(E_m \mid c_i) * P(c_i)$** and check the highest value (remember that this is not a probability)

NB DRAWBACKS

- Independence assumption
  - But many features are not independent
- "zero-frequencies" are a problem
  - Possible solution?
    - Smoothing techniques (as add-one)

OVERVIEW

- Learning objectives
- Topics
  - Language Representation:
    - Language as vectors
    - Documents as vectors
    - Words as vectors
    - Sentences as vectors
  - Feature Engineering Machine Learning Approach
    - Concept and examples of interesting features
    - Some widely used algorithms
    - End-to-end example
- Key takeaways
- Suggested readings

FORENSIC LINGUISTICS

- Application of linguistic knowledge, methods, and insights to the forensic context of law, language, crime investigation, trial, and judicial procedure
- Nice stories:
  - my/myself vs. me/meself
  - on the left vs. to the left vs. on the left side
  - I am going vs. I don't want. Also: "She tried to break it off. So, I broke her neck."

Forensic Linguistics Is also an NLP TASK!

ACTIVE LEARNING MOMENT

EXERCISE

- A cruel criminal (from now on CC) is stealing TV's remote controls from innocents' homes, and he/she always leaves a letter in the crime scene
- There was another crime, and a suspect was detained. Inspector Morcela wants to know if the suspect is the real criminal or an imitator

Inspector Morcela, the ONE

EXERCISE

- Due to your expertise in NLP, Inspector Morcela calls you:
  - he wants you to analyse the letters and tell him which are the chances of being the same criminal or an imitator
- You decide to approach this challenge as a classification problem:
  - You add to the given letters (positive examples) other letters written by other criminals (negative examples).
  - You decide to consider the following features:
    - 1 – the letter starts with the word "I" or not;
    - 2 – the letter has spelling errors or not;
    - 3 – the letter has few/average/high number of adjectives;
    - 4 – the letter is extremely polite/normal/rude.

Based on this, you build the following table:

Starts with "I"	Spelling errors	Adjectives	Style	Written by CC?
NO	YES	FEW	EXTR. POLITE	YES
NO	NO	FEW	EXTR. POLITE	YES
NO	YES	AVERAGE	EXTR. POLITE	NO
NO	YES	HIGH	NORMAL	NO
YES	YES	HIGH	RUDE	NO
NO	NO	HIGH	NORMAL	YES
YES	NO	AVERAGE	RUDE	NO
NO	YES	FEW	NORMAL	YES
YES	YES	FEW	RUDE	NO
YES	YES	HIGH	NORMAL	NO
YES	NO	FEW	NORMAL	NO
NO	NO	AVERAGE	NORMAL	NO
YES	YES	AVERAGE	EXTR. POLITE	NO
YES	NO	HIGH	NORMAL	YES

Q1: By using Naïve Bayes (no smoothing), find the probability of the new letter being written by CC.

Now you look at the new letter:

1 – It starts with "I";
2 – There are no spelling mistakes;
3 – It has few adjectives;
4 – The style is extremely polite.

P('Written_by_CC' = YES | 'starts_with_I' = YES, 'Spelling errors' = NO, Adjectives = FEW, Style = 'EXTR. POLITE')

Starts with "I"	Spelling errors	Adjectives	Style	Written by CC?
NO	YES	FEW	EXTR. POLITE	YES
NO	NO	FEW	EXTR. POLITE	YES
NO	YES	AVERAGE	EXTR. POLITE	NO
NO	YES	HIGH	NORMAL	NO
YES	YES	HIGH	RUDE	NO
NO	NO	HIGH	NORMAL	YES
YES	NO	AVERAGE	RUDE	NO
NO	YES	FEW	NORMAL	YES
YES	YES	FEW	RUDE	NO
YES	YES	HIGH	NORMAL	NO
YES	NO	FEW	NORMAL	NO
NO	NO	AVERAGE	NORMAL	NO
YES	YES	AVERAGE	EXTR. POLITE	NO
YES	NO	HIGH	NORMAL	YES

P('Written_by_CC' = YES | 'starts_with_I' = YES, 'Spelling errors' = NO, Adjectives = FEW, Style = 'EXTR. POLITE') =

P('starts_with_I' = YES | 'Written_by_CC' = YES)* P('Spelling errors' = NO | 'Written_by_CC' = YES) * P(Adjectives = FEW | 'Written_by_CC' = YES)* P(Style = 'EXTR. POLITE | 'Written_by_CC' = YES) * P('Written_by_CC' = YES)

Starts with "I"	Spelling errors	Adjectives	Style	Written by CC?
NO	YES	FEW	EXTR. POLITE	YES
NO	NO	FEW	EXTR. POLITE	YES
NO	YES	AVERAGE	EXTR. POLITE	NO
NO	YES	HIGH	NORMAL	NO
YES	YES	HIGH	RUDE	NO
NO	NO	HIGH	NORMAL	YES
YES	NO	AVERAGE	RUDE	NO
NO	YES	FEW	NORMAL	YES
YES	YES	FEW	RUDE	NO
YES	YES	HIGH	NORMAL	NO
YES	NO	FEW	NORMAL	NO
NO	NO	AVERAGE	NORMAL	NO
YES	YES	AVERAGE	EXTR. POLITE	NO
YES	NO	HIGH	NORMAL	YES

P('Written_by_CC' = YES | 'starts_with_I' = YES, 'Spelling errors' = NO, Adjectives = FEW, Style = 'EXTR. POLITE') =

P('starts_with_I' = YES | 'Written_by_CC' = YES)*
P('Spelling errors' = NO | 'Written_by_CC' = YES) *
P(Adjectives = FEW | 'Written_by_CC' = YES)*
P(Style = 'EXTR. POLITE | 'Written_by_CC' = YES) *
P('Written_by_CC' = YES) = 1/5 * 3/5 * 3/5 * 2/5 * 5/14 = 0,01

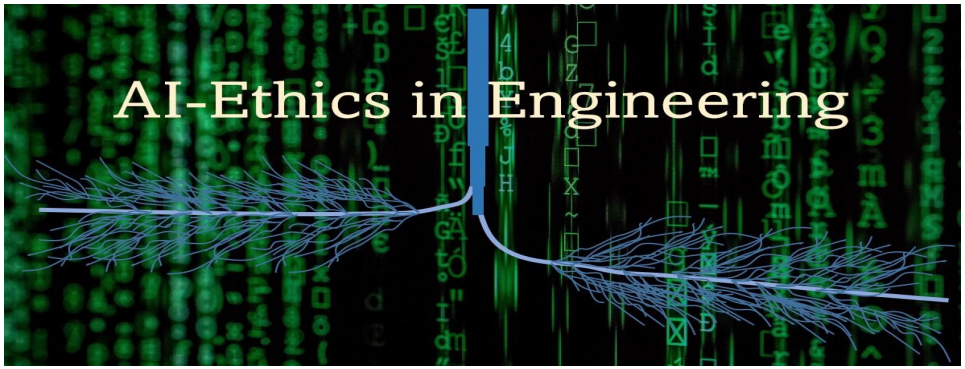
Do not forget that this is not a probability!

Starts with "I"	Spelling errors	Adjectives	Style	Written by CC?
NO	YES	FEW	EXTR. POLITE	YES
NO	NO	FEW	EXTR. POLITE	YES
NO	YES	AVERAGE	EXTR. POLITE	NO
NO	YES	HIGH	NORMAL	NO
YES	YES	HIGH	RUDE	NO
NO	NO	HIGH	NORMAL	YES
YES	NO	AVERAGE	RUDE	NO
NO	YES	FEW	NORMAL	YES
YES	YES	FEW	RUDE	NO
YES	YES	HIGH	NORMAL	NO
YES	NO	FEW	NORMAL	NO
NO	NO	AVERAGE	NORMAL	NO
YES	YES	AVERAGE	EXTR. POLITE	NO
YES	NO	HIGH	NORMAL	YES

Q2: What can you say to Inspector Morcela considering that your verdict can take an innocent to prison or lead to the release of a criminal? (this is serious)

THE VEREDICT

- After finding P('Written by CC' = NO | bla-bla), you would have a clue about the letter being written or not by CC. However, even so, we have very little data, thus, we could never tell Morcela for sure that the suspect should be released or go to prison. Besides, we are using Naïve Bayes (naïve assumption). Besides (again)… an innocent could go to prison due to you. That is very disturbing.



https://towardsdatascience.com/ai-ethics-in-engineering-437ec07046a6

KEY TAKEAWAYS

KEY TAKEAWAYS

- Words, sentences and documents can be represented as vectors (sparse for now)
- Being able to represent language as vectors allow us to apply mathematic stuff to language (for instance, similarity metrics and machine learning algorithms)
- In feature-based machine learning the specific characteristics of the data in hands is carefully analysed and tailored features are design to be given as algorithms input
- We should always be very carefully in taking blind decisions, based on a score produced by an algorithm

SUGGESTED READINGS

READINGS

- Jurafsky: 6.3, 6.4, 6.5