U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

MIEIC
EIC0029 – Artificial Intelligence

Introduction to Natural Language Processing

Henrique Lopes Cardoso

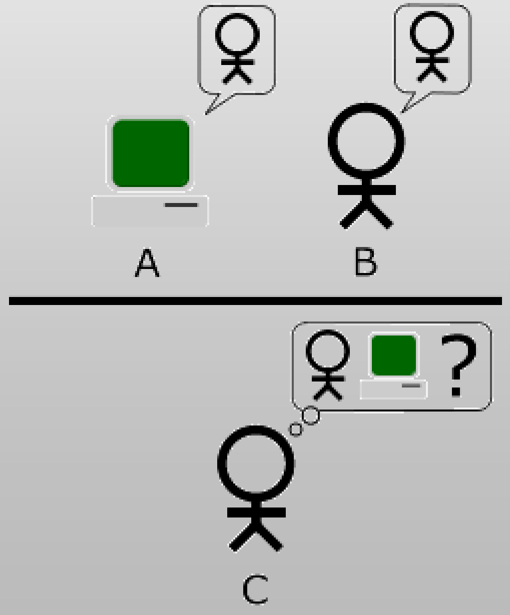
hlc@fe.up.pt

May 2021

The Turing Test

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human."

[Alan Turing, 1950]



Capabilities:

- natural language processing
- knowledge representation
- automated reasoning
- machine learning

Chatbot Hype

Chatbot Conversations

Natural Language Processing and Understanding

Conversations	Retrieval-Based	Generative-Based
Open Domain	Impossible	General AI (Hardest)
Closed Domain	Rules-Based (Easiest)	Smart Bot (Hard)

Responses

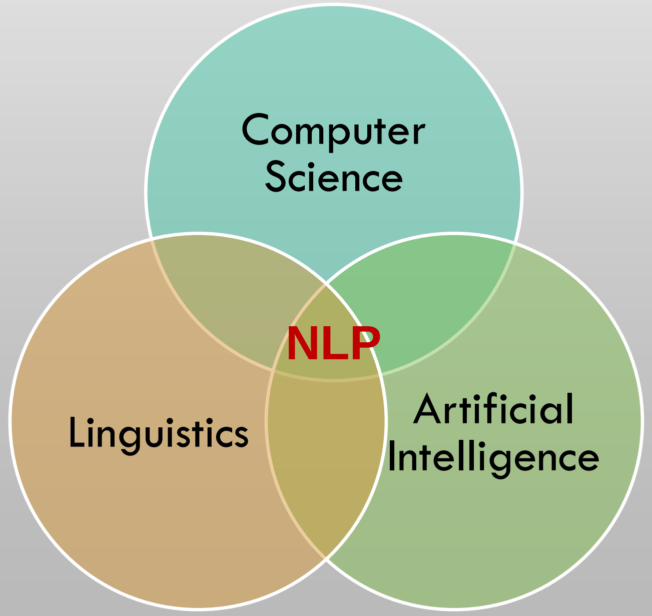
Natural Language Processing (NLP)

definitions, tasks and applications

Natural Language Processing

Natural language processing (**NLP**) is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between computers and human (natural) languages, and, in particular, concerned with programming computers to fruitfully process large natural language corpora.

[Wikipedia]



Challenges:

- natural language understanding
- natural language generation
- language and machine perception
- dialog systems

Some NLP Applications

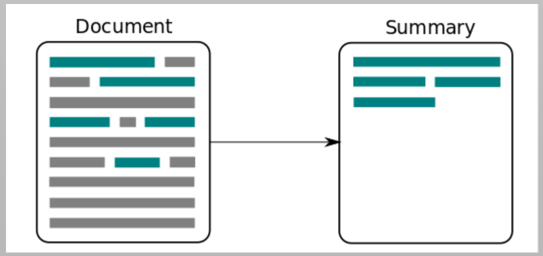
- Machine Translation
  - Based on multilingual textual corpora
  - Text translation and multilingual real-time conversations

- Speech-to-Text/Text-to-Speech
  - Convert spoken language to written text and vice versa
  - Chatbots, voice control, domotics, readers, …

Some NLP Applications

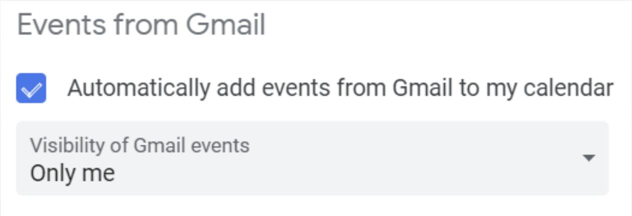
- Sentiment Analysis and Opinion Mining
  - Determine polarity about specific topics
  - Identify trends of public opinion in social media
  - Analyze product reviews

- Text Summarization
  - Build a summary out of a long text



Some NLP Applications

- Information Extraction
  - Extract relevant entities from text
  - Event identification, “add to calendar” features



Subject: extra NLP class
Date: September 25, 2020
To: Henrique Lopes Cardoso

Dear Henrique, we’re having an extra NLP class tomorrow, from 10:00-11:30, via Zoom.
-HLC

Event: extra NLP class
Date: Sept-26-2020
Start: 10:00am
End: 11:30am
Where: Zoom

- Question Answering
  - Automatically answer questions posed in natural language
  - IBM Watson won *Jeopardy!* on 2011

Some NLP Applications

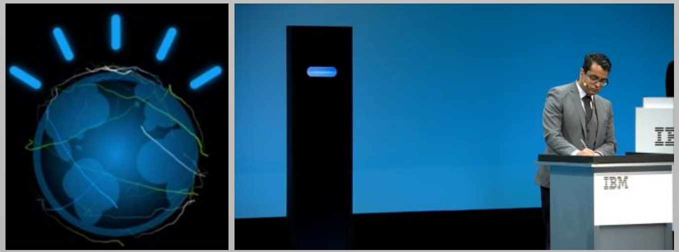
- Automated Fact Checking and Fake News Detection
  - Given a claim, collect evidence to check if it is true
  - Given a news article, check whether it is accurate

FEVER

- Argument Mining and Debate Portals
  - Extract arguments that expose a certain position
  - Aggregate pros and cons for a debatable topic
  - Debate on a given topic

PRO DEBATEPEDIA CON

BRITANNICA PROCON.ORG Reliable. Nonpartisan. Empowering.



NLP Tasks

- Most NLP tasks aim at making it easier for machines to understand natural language

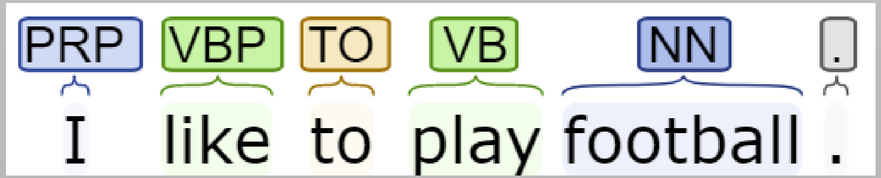
- A few of the most relevant tasks:
  - Tokenization
    - Split a sentence into tokens (words)

```
That U.S.A. poster-print costs $12.40...
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

  - Sentence breaking
    - Split a text into sentences

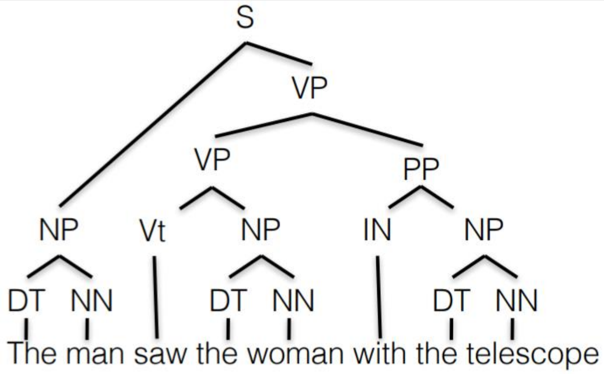
```
Hello. Are you Mr. Smith? I've finished my M.Sc. on Informatics!
['Hello.',
 'Are you Mr. Smith?',
 'I've finished my M.Sc. on Informatics!']
```

  - Part-of-Speech (POS) tagging
    - Determine the role category for each word in a sentence



NLP Tasks

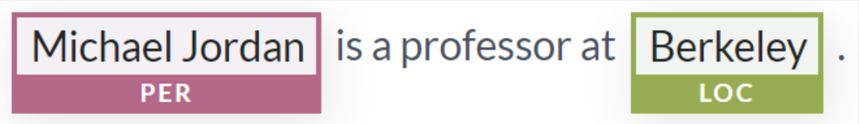
- Syntax parsing
  - Determine the parse tree (grammatical analysis) of a sentence



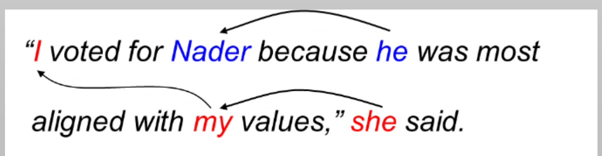
- Word sense disambiguation
  - Select the meaning of words in a context

A mouse is a mammal.
My mouse is broken.

- Named-entity recognition (NER)
  - Determine which items in a text map to entities (people, institutions, places, dates, …)



- Co-reference resolution
  - Determine which words ("mentions") refer to the same objects ("entities")



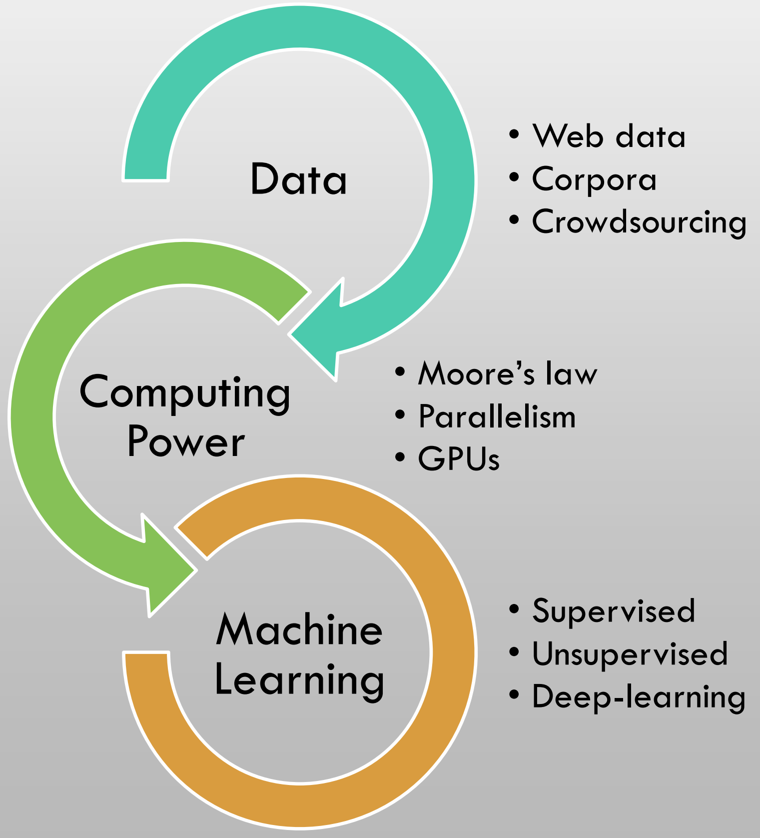
- …

Language Resources

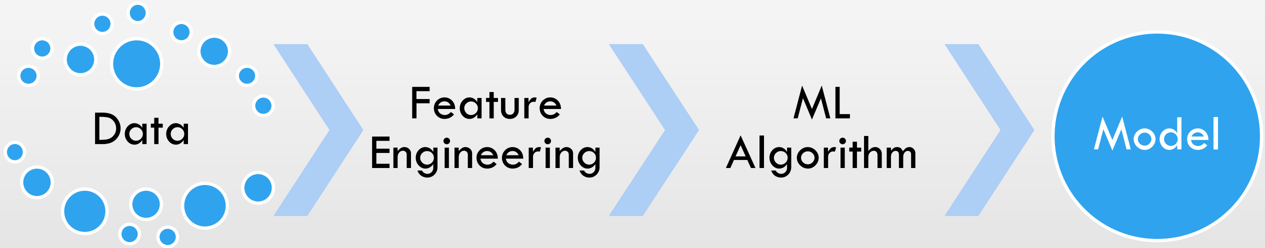
- Lexical databases: WordNet, CONTO.PT, WordNet.pt, …
  - Synsets, word-sense pairs
  - Semantic relations: hypernym/hyponym, meronym/holonym, troponym, entailment, …
- TreeBanks: PDTB, CSTNews, …
  - Text corpora annotated with discourse or semantic sentence structures
- Knowledge graphs: Google, DBpedia, …
  - Entity-predicate relations
- Lexicons: SentiWordNet, SocialSent, SentiLex, …
  - Words connoted with specific classes (+/-, …)
- Word embeddings: word2vec, GloVe, fastText, …
  - Distributed representations of words
- Language Models: ELMo, BERT, …
- Annotated datasets for several NLP tasks
  - Usually released under "shared-tasks", such as those at SemEval
- …

Statistical NLP

- Knowledge (grammar) based methods are overtaken by data-driven statistical techniques



Machine Learning in NLP



- Preparation
- Learning instances

- Definition
- Extraction
- Selection

- NB, SVM
- MLP
- CRF, HMM

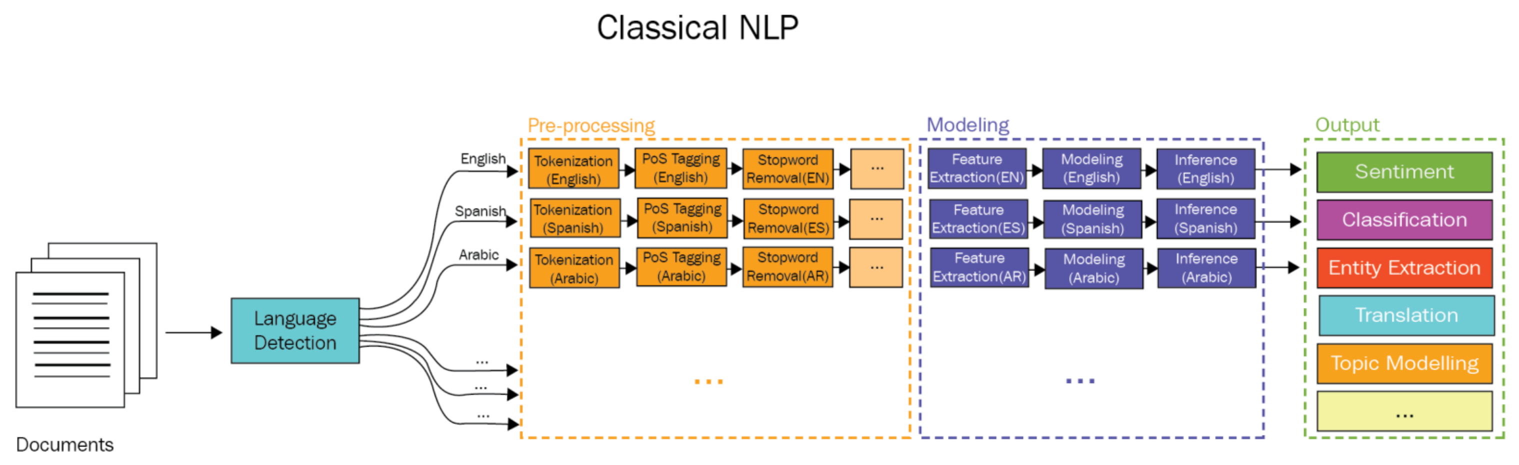
- Validation

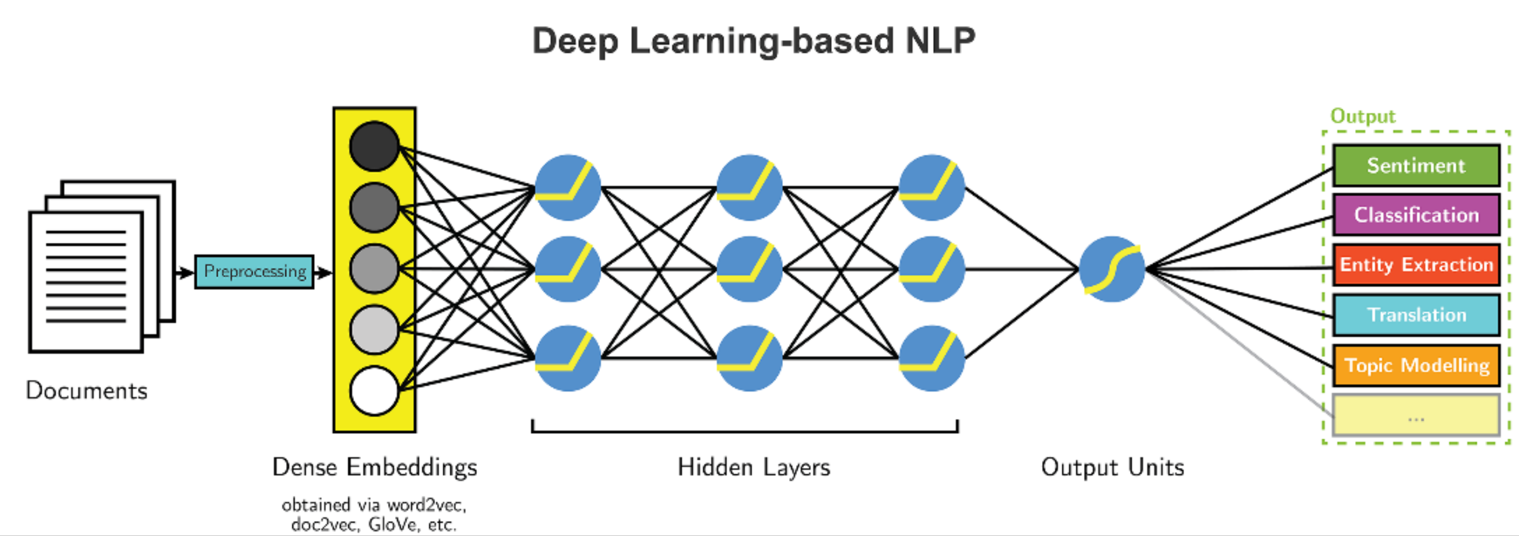
- Common linguistic features used in NLP
  - *Lexical*: BoW, TF-IDF, n-grams, word stems, …
  - *Syntactic*: part-of-speech (POS) tagging, parsing, …
  - *Grammatical*: verb tenses, number, gender, …
  - *Semantic*: word similarities, relations, embeddings, …
  - *Structural*: paragraphs, sentence length, document sections, distance metrics, …

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

Classical vs Deep Learning NLP





Natural Language Understanding

- Can computers *understand* natural language?

- 2011: IBM Watson, a question answering computer system, won *Jeopardy!*

  - Q&A technology takes a question expressed in natural language and returns a precise answer
  - Does Watson *understand* the questions?

Natural Language Understanding

- Why is NLU difficult?
  - Ambiguity (“Red tape holds up new bridges”, “Hospitals sued by 7 Foot Doctors”)
  - Non-standard language use (e.g. in Twitter)
  - Segmentation issues (“the New York-New Haven Railroad”)
  - Idiomatic expressions (“throw in the towel”, “dark-horse candidate”, …)
  - Neologisms (“unfriend”, “retweet”, …)
  - World knowledge (“Mary and Sue are sisters” / “Mary and Sue are mothers”)
  - Tricky entity names (“Where is *A Bug’s Life* playing?”, “*Let it Be* was recorded …”)
  - …
- Need knowledge about language and knowledge about the world!

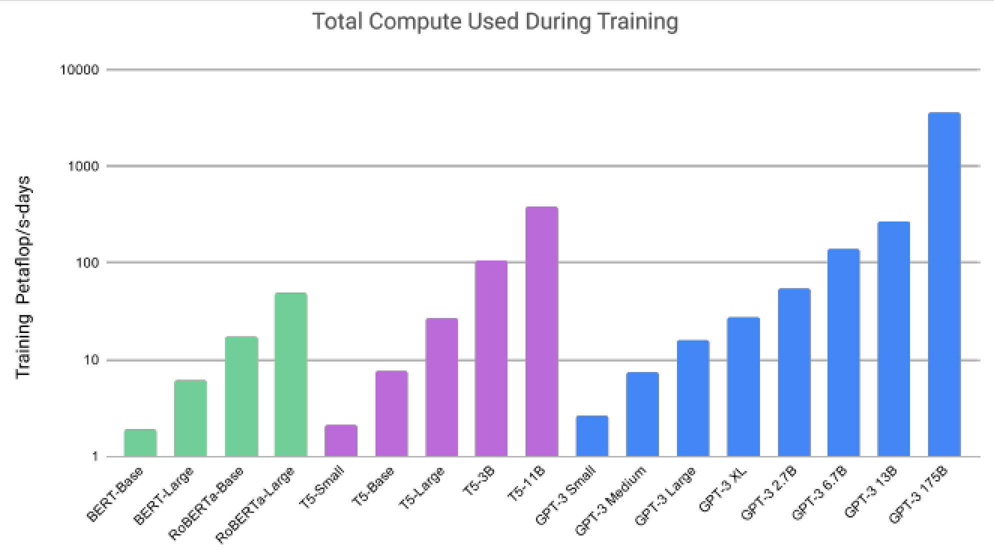
Natural Language Generation

- The process of transforming structured data into natural language
  - *Data-to-text*: generate textual summaries of databases and data sets (weather, finance, business, …)
  - Integrated into business intelligence and analytics platforms
- Other application areas: automated journalism, chatbots, question-generation, …
  - … and fake news?

- 2019: OpenAI announces GPT-2
  - A large language model with 1.5 billion parameters, trained on 8 million web pages
  - Generates "convincing" news articles and product reviews (but it cannot write "true" articles)
  - Doesn't understand what it generates

Natural Language Generation

- 2020: OpenAI announces GPT-3
  - A larger language model with 175 billion parameters
  - Evaluated on over two dozen NLP datasets, as well as several novel tasks



Natural Language Generation

- GPT-3

Plain language: I don't know who the 10 people really are yet but they are all to blame.
Legal language: Defendants Does 1 through 10 are sued herein under fictitious names. Their true names and capacities are unknown to Plaintiff. When their true names and capacities are ascertained, Plaintiff will amend this complaint by inserting their true names and capacities herein. Plaintiff is informed and believes and thereon alleges that each of the fictitiously named Defendants is responsible in some manner for the occurrences herein alleged, and that Plaintiff's damages as herein alleged were proximately caused by those Defendants.
Plain language: Everyone I'm suing worked together.
Legal language: Plaintiff alleges on information and belief that Defendants, and each of them, are and at all times herein mentioned were the agents and servants of each other and in doing the things hereinafter alleged were acting within the scope of such agency and service.
Plain language: The people I'm suing are my landlords.
Legal language: The Defendants are, in some manner, the owners and lessors of the real property where Plaintiff resides.
Plain language: My landlord entered my apartment without my permission.
Legal language: Defendant entered Plaintiff's dwelling without Plaintiff's consent.
Plain language: My apartment had mold and it made me sick.
Legal language: Plaintiff's dwelling was infested with toxic and allergenic mold spores, and Plaintiff was rendered physically incapable of pursuing his or her usual and customary vocation, occupation, and/or recreation.
Plain language: My landlord told me he didn't want Black people living in the complex.
Legal language: Plaintiff is informed and believes and thereon alleges that the Defendant, by and through his agents and servants, has engaged in a pattern and practice of refusing to rent, and otherwise made unavailable, dwelling units to Black persons because of their race or color.
Plain language: My landlord didn't maintain the property.
Legal language: The Defendants have permitted the real property to fall into disrepair and have failed to comply with state and local health and safety codes and regulations.
Plain language: The apartment was covered by San Francisco rent control.
Legal language: Plaintiff's dwelling was a rent-controlled unit within the meaning of Section 37.2(b) of the San Francisco Rent Ordinance.
Plain language: My landlord didn't return my security deposit in the time allowed by California law.
Legal language: Defendant has failed to refund to Plaintiff all sums of money paid to Defendant as security deposit, within the time periods specified in California Civil Code section 1950.5.

Question: A 10 year old boy presents with recurrent episodes of dyspnoea, cough and wheeze triggered by cold air and allergens. After performing tests he is diagnosed with an obstructive disease and given medication. Which receptor is the medication most likely to work on:

A) muscarinic receptor
B) G-protein coupled receptor
C) Era
D) Erb
E) a-1 receptor

Correct answer is **B**

Explanation: The patient is suffering from asthma, a disease of the airways. The drugs used to treat asthma are bronchodilators. The bronchodilators act on the beta-2 receptors. The beta-2 receptors are G-protein coupled receptors

Some selected flaws:

Q: Which is heavier, a toaster or a pencil?
A: A pencil is heavier than a toaster.

Q: How many eyes does my foot have?
A: Your foot has two eyes.

Q: How do you sporgle a morgle?
A: You sporgle a morgle by using a sporgle.

Q: How many rainbows does it take to jump from Hawaii to seventeen?
A: It takes two rainbows to jump from Hawaii to seventeen.

Q: Do you understand these questions?
A: I understand these questions.

Q: When counting, what number comes before 10,000?
A: Nine thousand and ninety-nine comes before 10,000.

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

Basic Text Processing

regular expressions, tokenization, normalization, lemmatization, stemming, segmentation

Regular Expressions

- A regular expression is a sequence of characters that define a search pattern
  - Makes use of meta-characters, such as {}[]()^$.|*+?\
    - [A-Z] uppercase letter
    - [a-z] lowercase letter
    - [0-9] digit
    - ^ negation
    - | disjunction
    - ? optional
    - * zero or more
    - + one or more
    - . Any
    - …

Example: find all instances of the word "**the**" in a text

- the — misses capitalized letters
- [tT]he — returns "other" or "theology"
- [^a-zA-Z][tT]he[^a-zA-Z]

Text Normalization

- Converting text to a more convenient form
- Tokenization: segmenting words in a text
- Word normalization
    - Case folding
    - Lemmatization
    - Stemming
- Sentence segmentation

Word Tokenization

- Initial approach: look for spaces, punctuation and other special characters
- What about:
  - Ph.D., AT&T, can't, we're, state-of-the-art, guarda-chuva
  - $45.55, 123,456.78, 123.456,78
  - 07/04/2020, April 4, 2020
  - http://www.fe.up.pt, hlc@fe.up.pt, #iart
  - New York, Vila Nova de Gaia
- Certain languages do not have space splitting!
  - German, Chinese, Japanese, …

```
import nltk
from nltk import word_tokenize

text = 'That U.S.A. poster-print costs $12.40...'
tokens = word_tokenize(text)
print(len(tokens))
print(tokens)
```

```
7
['That', 'U.S.A.', 'poster-print', 'costs', '$', '12.40', '...']
```

Sub-word Tokens

- What if we tokenize by word pieces?
- Advantages:
  - Dealing with unknown words (particularly relevant for Machine Learning)
    - E.g. training corpus containing “low” and “lowest”, but not “lower”, which appears in the test corpus
  - Robustness to misspellings
  - Dealing with multi-lingual data
- Wordpieces (used, for instance, in BERT)
  - Given the token “intention” and the dictionary [“in”, “tent”, “intent”, “##tent”, “##tention”, “##tion”, “#ion”], obtains the tokens [“intent”, “##ion”]

Word Normalization

- Putting words/tokens in a standard format
  - Reduces the vocabulary size
  - Helps Machine Learning models to generalize

- Case folding
  - Putting every word in lower case
  - Not always helpful, and thus not always performed
    - Sentiment analysis: uppercase might denote anger, …
    - Named-entity recognition: US/us, Mike Pence/mike pence, …

Word Normalization

- Lemmatization
  - Determining the root of the word: many words have the same root!
    - "am", "are", "is" → "be"
    - "He is reading detective stories" → "He be read detective story"
  - Morphological parsing: words are built from morphemes
    - Stems: the central morpheme of a word, supplying the main meaning
    - Affixes: adding additional meaning
- Stemming
  - A simpler and cruder method that simply cuts off word final affixes

Sentence Segmentation

- Splitting a text into sentences
  - Typically based on punctuation marks
  - But the period '.' is particularly ambiguous (e.g. Mr., Ph.D., Inc., Sr., …)
  - Decide (learn) whether a period is part of the word or is a sentence-boundary marker
    - Abbreviation dictionary can help determine whether the period is part of a commonly used abbreviation

```
from nltk.tokenize import sent_tokenize
text = "Hello. Are you Mr. Smith? Just to let you know that I have finished my M.Sc. and Ph.D. on AI. I loved it!"
print(sent_tokenize(text))
```

```
['Hello.', 'Are you Mr. Smith?', 'Just to let you know that I have finished my M.Sc.', 'and Ph.D. on AI.', 'I loved it!']
```

Text Classification

bag-of-words, Naïve Bayes, features, generative and discriminative classifiers

Slides mostly based on Jurafsky and Martin (2019): “Speech and Language Processing”, 3rd ed. draft.

Text Classification Tasks

- Given a text, classify it according to a number of classes
  - Spam detection in emails: spam/not spam
  - Sentiment analysis in product reviews: positive/negative, -/0/+, --/-/0/+/++
  - Assign subject categories, topics, or genres
  - Authorship identification from a closed list
  - Age/gender identification
  - Language detection
  - …

- More formally:
  - Input: a document d and a fixed set of classes $C = \{c_1, c_2, \ldots, c_m\}$
  - Output: predicted class $c \in C$ for document d

Hand-coded Rules

- Rules based on combinations of words or other features
  - Spam detection: black-list of addresses and keyword detection
  - Sentiment analysis: ratio of word polarities appearing in a sentiment lexicon
  - …
- Accuracy can be high…
  - If rules are carefully refined by expert
- …but building and maintaining these rules is expensive

Supervised Machine Learning

- Making use of annotated datasets through Machine Learning algorithms
- Building a model
  - Input:
    - a fixed set of classes $C = \{c_1, c_2, \ldots, c_m\}$
    - a training set of *m* hand-labeled documents $\{(d_1,c_1), (d_2,c_2), \ldots, (d_n,c_n)\}$, where $d_i \in D$ and $c_i \in C$
  - Output: a classifier $\gamma : D \rightarrow C$
    - a mapping from documents to classes (or class probabilities)
- Classifying a document
  - Input
    - a document d
    - a classifier $\gamma : D \rightarrow C$
  - Output: predicted class $c \in C$ for document d

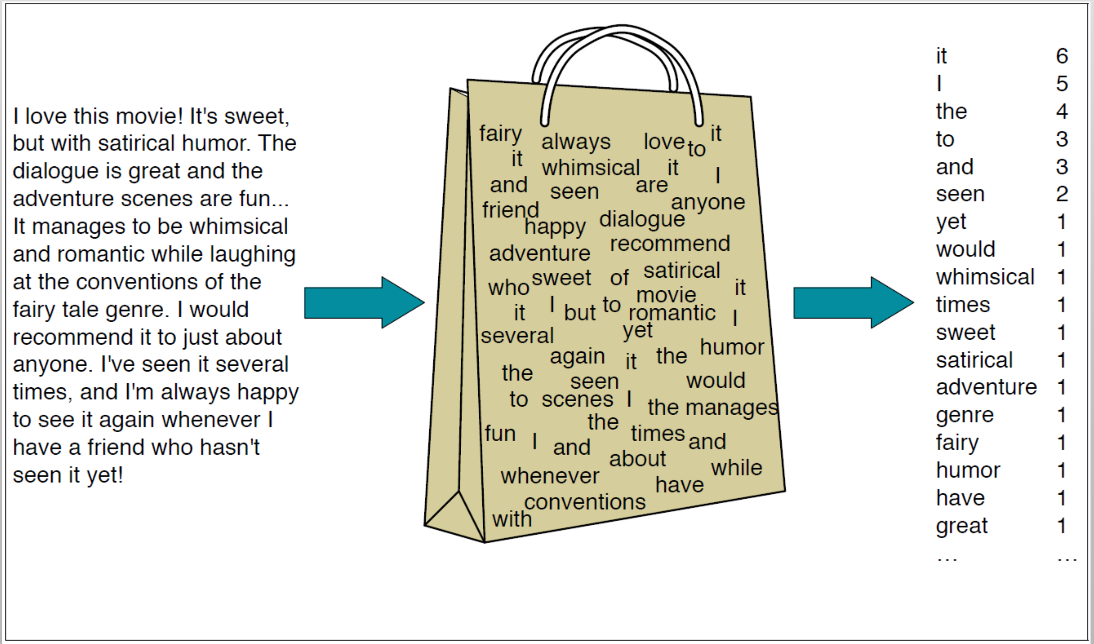
Classifiers

- Probabilistic classifier: more than predicting a class, outputs the probability of the observed document belonging to each of the classes

- Generative vs Discriminative classifiers
  - Generative classifiers build a model of how a class could generate some input data
    - Given an observation, return the class that has most likely produced the observation
    - Example: Naïve Bayes
  - Discriminative classifiers learn what features from the input are most useful to discriminate between the different possible classes
    - Examples: Decision Trees, Logistic Regression, Support Vector Machines

Bag of Words

- Machine Learning methods require that the data is represented as a set of features
- We thus need a way of going from a document *d* to a vector of features *X*
- The bag-of-words model
  - an unordered set of words, keeping only their frequency in the document
  - assume position does not matter



Naïve Bayes

- Naïve Bayes (NB) makes a simplifying (naïve) assumption about how the features interact

- Bayes rule:

$$P(c\,|\,d) = \frac{P(d\,|\,c)P(c)}{P(d)}$$

- Most likely class:

$$\begin{aligned} c_{MAP} &= \underset{c \in C}{\operatorname{argmax}}\, P(c\,|\,d) \\ &= \underset{c \in C}{\operatorname{argmax}} \frac{P(d\,|\,c)P(c)}{P(d)} \\ &= \underset{c \in C}{\operatorname{argmax}}\, P(d\,|\,c)P(c) \end{aligned}$$

- Representing a document with features:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, \ldots., f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

- Assuming conditional independence:

$$P(f_1, f_2, \ldots., f_n|c) \;=\; P(f_1|c) \cdot P(f_2|c) \cdot \ldots \cdot P(f_n|c)$$

- Naïve Bayes classifier:

$$c_{NB} \;=\; \underset{c \in C}{\operatorname{argmax}}\, P(c) \prod_{f \in F} P(f|c)$$

Naïve Bayes

- Applying NB to the text:

$$\text{positions} \leftarrow \text{all word positions in test document}$$
$$c_{NB} = \operatorname*{argmax}_{c \in C} P(c) \prod_{i \in positions} P(w_i|c)$$

- Going to log space:
  - avoid underflow and increase speed

$$c_{NB} = \operatorname*{argmax}_{c \in C} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

  - because $\log(xy) = \log(x) + \log(y)$
  - highest log probability class is still most probable

- Computing probabilities
  - Class priors: $\hat{P}(c) = \frac{N_c}{N_{doc}}$
  - Word probabilities per class:

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)}$$

  - Handling non-occurring words in a class
    - Add-one (Laplace) smoothing:

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)} = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Naïve Bayes Example

- A sentiment analysis (or polarity) task:

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

- Prior distributions:

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

- Word probabilities per class:

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$
$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$
$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

  - “with” doesn’t occur in training set: ignore it

- Class probabilities:

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$
$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

  - Chosen class: **negative (-)**

Naïve Bayes is Not So Naïve

- Very fast, low storage requirements
- Robust to irrelevant features: they cancel each other without affecting results
- Very good in domains with many equally important features
  - Decision Trees suffer from fragmentation in such cases – especially if little data
- Optimal if the assumed independence assumptions hold
- A good dependable baseline for text classification

Word Occurrence vs Word Frequency

- In how many documents of the class does the word occur?

Four original documents:

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- \+ and satire and great plot twists
- \+ great scenes great film

After per-document binarization:

- − it was pathetic the worst part boxing scenes
- − no plot twists or great scenes
- \+ and satire great plot twists
- \+ great scenes film

	NB Counts		Binary Counts	
	+	−	+	−
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

Dealing with Negation

- *I really like this movie* (positive)
- *I didn’t like this movie* (negative)

- Prepending `NOT_` to words affected by negation tokens (n’t, not, no, never, …)
  - `I did n't like this movie , but I`
  - `I did n't NOT_like NOT_this NOT_movie , but I`

- Using bigrams instead of single words
  - Sequences of two words: instead of “not” and “recommend”, “not recommend”

Making use of Lexicons

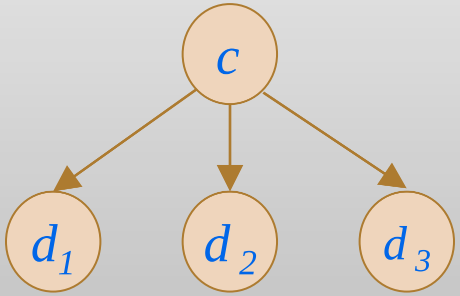
- Lexicons provide external knowledge that can be very useful for the task!
- Sentiment lexicons
  - Lists of words that are pre-annotated with positive or negative polarity
  - Example: MPQA Subjectivity Lexicon
    - 6885 words, 2718 positive and 4912 negative, strongly or weakly biased
    - + : admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great
    - – : awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate
- Features based on the occurrence of (positive or negative) sentiment-biased words
  - Useful when training data is sparse or vocabulary usage in test and training sets do not match
  - Dense lexicon features may generalize better than sparse individual-word features

Building other Features

- Predefine likely sets of words or phrases
  - Spam detection: “viagra”, “password will expire”, “Your mailbox has exceeded the storage limit”, “millions of dollars”, “click here”, “urgent reply”, …
- Paralinguistic and extra-linguistic features
  - Words in capital letters
  - HTML with low ratio of text-to-image, sender email address, …
- N-grams (character or word level)
  - Sequences of two (bigrams), three (trigrams) or even more words or characters
  - Can help alleviate the conditional independence assumption of NB
  - But typically generates a very sparse feature space (many bigrams will rarely occur)

Generative vs Discriminative Classifiers

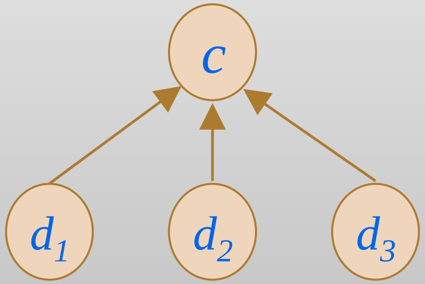
- A generative model makes use of a likelihood term: how to generate the features of a document if we knew it was of class c?



- Examples:
  - Naïve Bayes, Hidden Markov Models, …

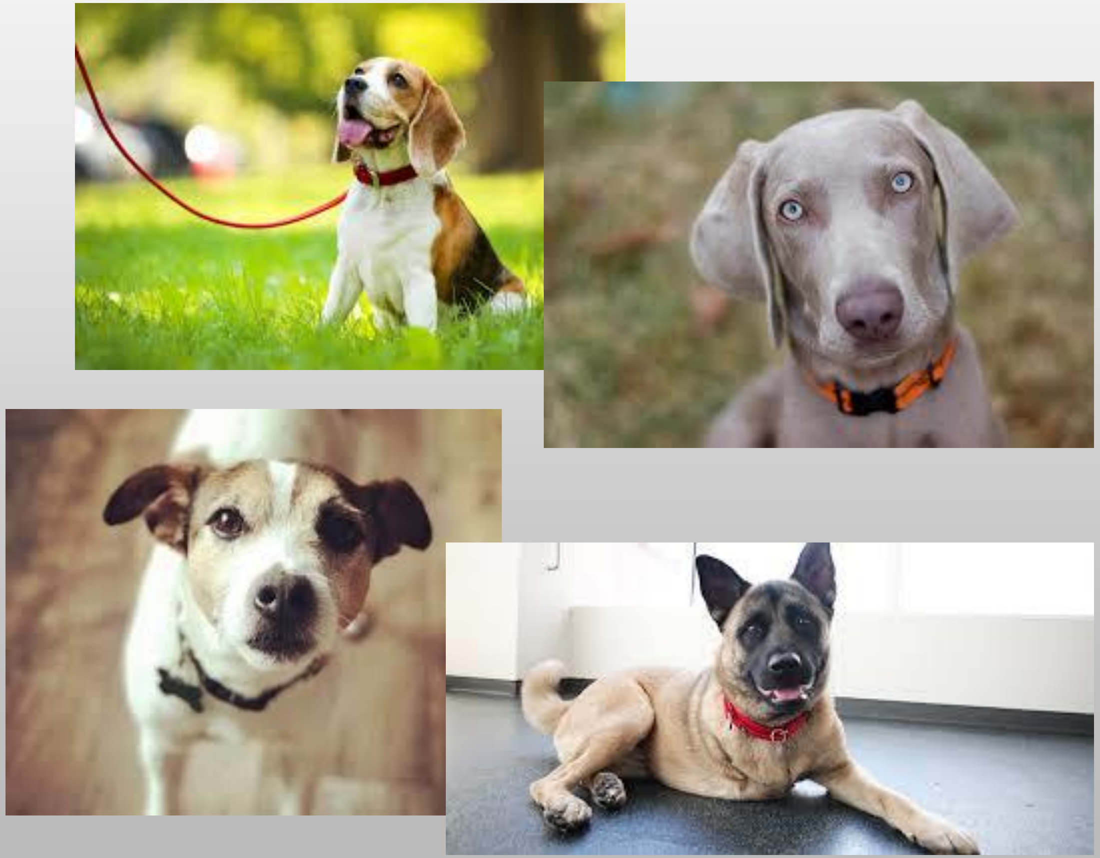
$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

- A discriminative model tries to learn to distinguish the classes, and attempts to directly compute $P(c|d)$



- Examples:
  - Logistic Regression, Decision Trees, Support Vector Machines, Neural Networks, Conditional Random Fields, …

Generative vs Discriminative Classifiers



Generative vs Discriminative Classifiers

- Naïve Bayes has overly strong conditional independence assumptions
  - Edge case: two strongly correlated features, e.g. using the same feature twice
    - NB treats both copies of the feature as if they were separate
  - If multiple features tell mostly the same thing, such evidence is overestimated
- Discriminative classifiers (e.g. Logistic Regression) assign more accurate probabilities when there are many correlated features
- Naïve Bayes is easy to implement and very fast to train (there is no optimization step)
- Logistic Regression generally works better on larger documents or datasets

UPORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

Modern NLP

word embeddings, deep learning, language models

Word Embeddings

- Representing a sentence based on a bag-of-words model obtains very sparse representations
  - Given a vocabulary of size |V|, a document is represented as a vector with many 0's and a few 1's

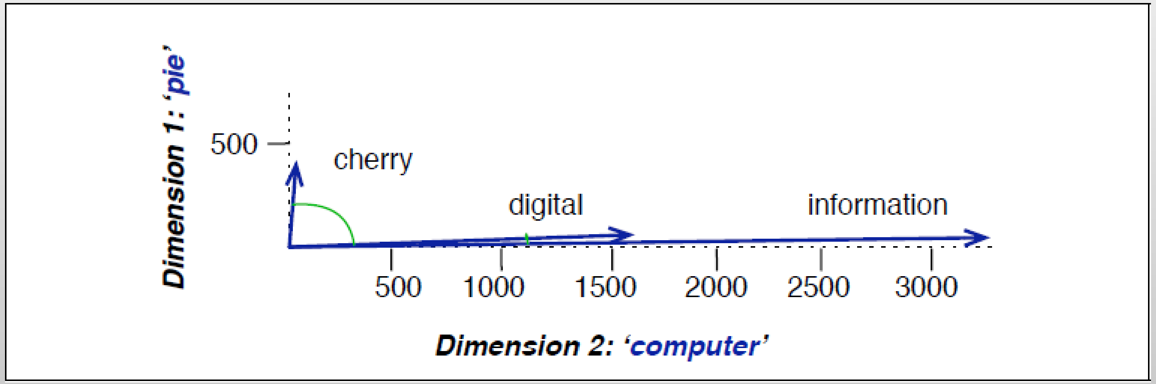
- We can represent the meaning of a word based on the contexts in which it occurs
  - Unsupervised approach: observe word usage on large (non-annotated) corpora
  - Sparse vectors (|V| = 20k? 50k?): words represented by a function of the counts of nearby words
  - Dense vectors: short vectors (50-1000 mostly non-zero real numbers), trained representations

- Representing sentences/documents: compute centroid of the word vectors

Word Embeddings: Comparing Words or Documents

- We can compute the semantic similarity between words using operators such as cosine similarity

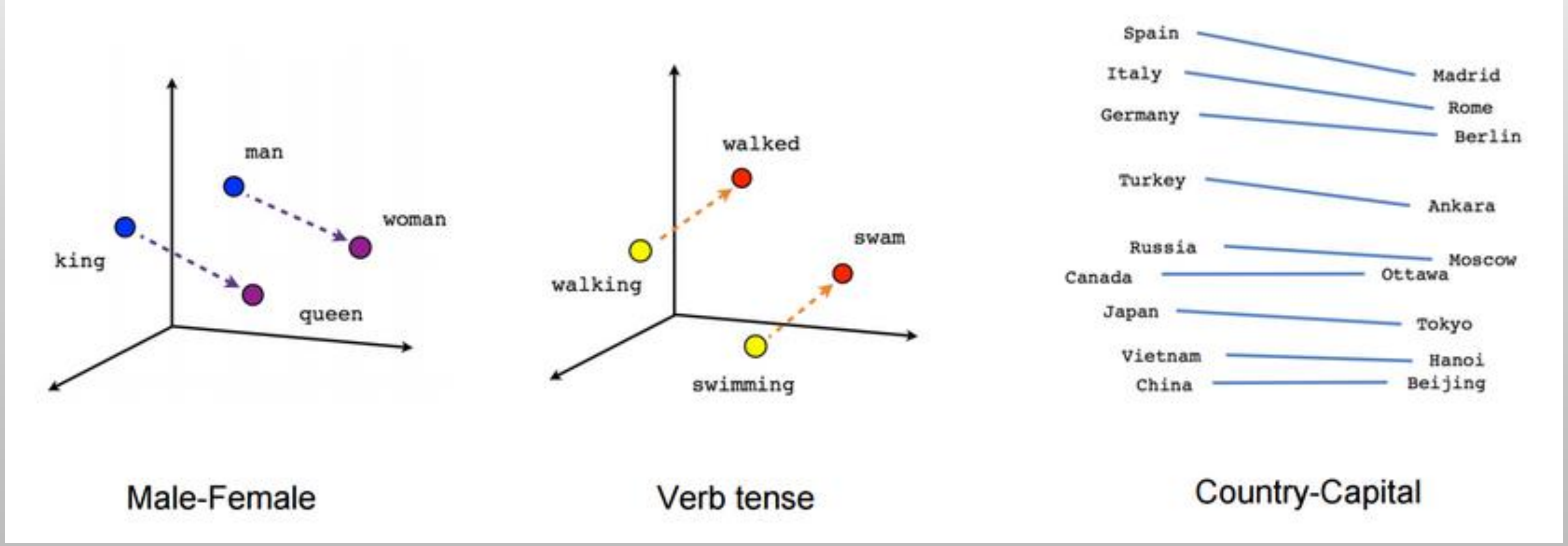
$$\text{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|}$$



- Comparing documents
  - Centroid of the vectors for all the words in the document: $d = \frac{w_1 + w_2 + \ldots + w_k}{k}$
  - Similarity between two documents: $\cos(d_1, d_2)$

Semantic Properties of Word Embeddings

- Analogy: ability to capture relational meanings

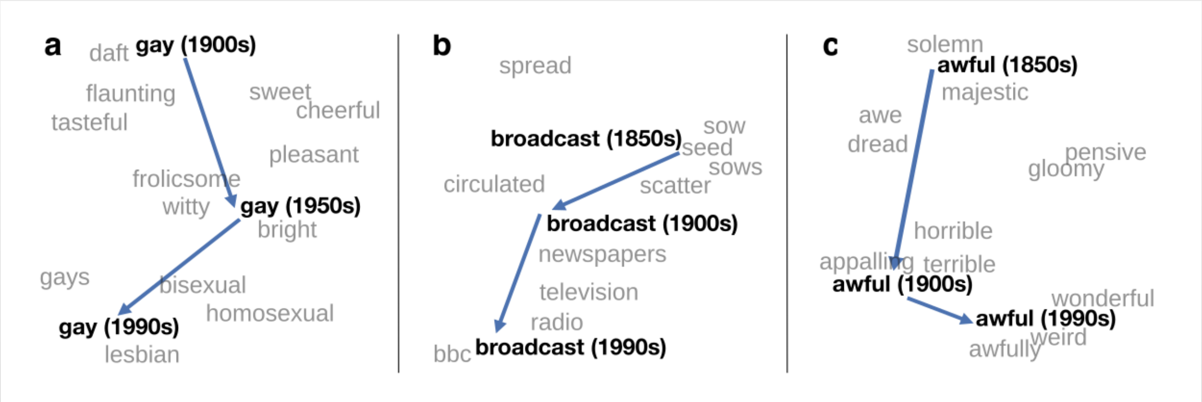


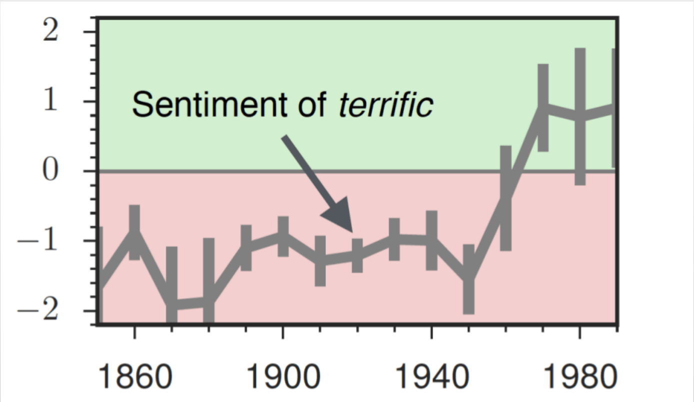
U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

Semantic Properties of Word Embeddings

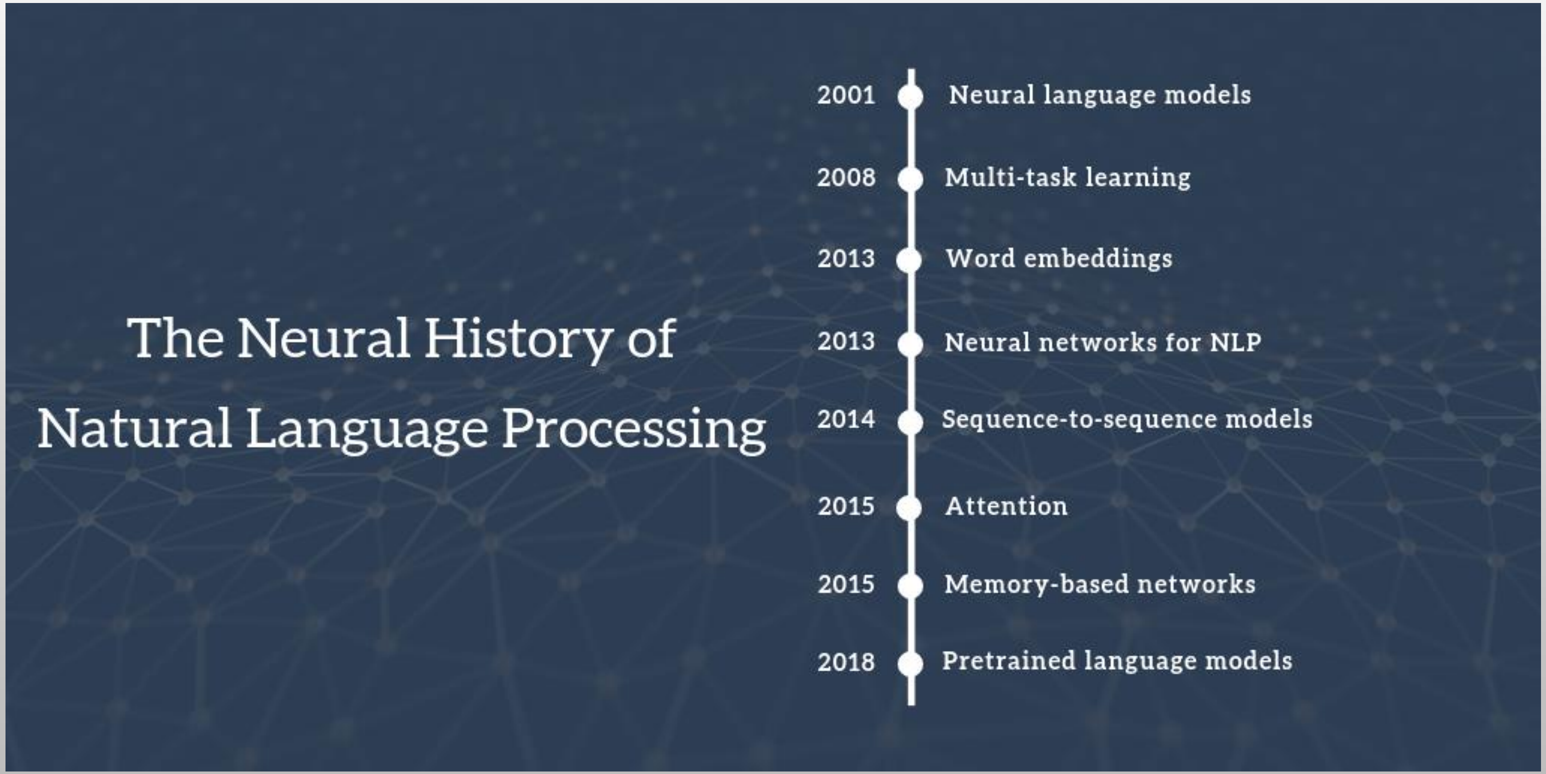
- Historical semantics: how word meaning changes over time [Hamilton et al., 2016]





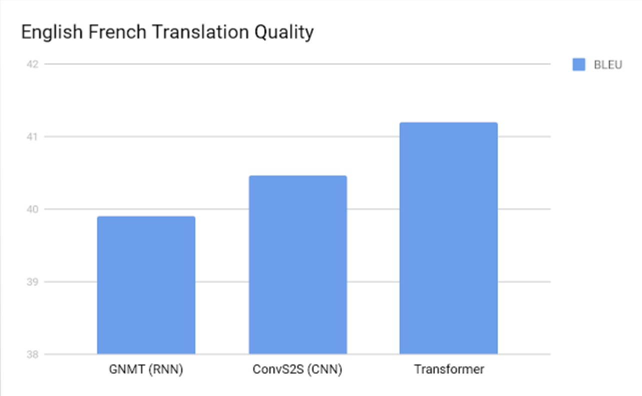
- Embeddings reproduce the implicit biases and stereotypes that are latent in the text [Bolukbasi et al., 2016]
    - ‘man’ is to ‘computer programmer’ as ‘woman’ is to ‘homemaker’
    - ‘father’ is to ‘doctor’ as ‘mother’ is to ‘nurse’

Deep Learning in NLP



https://ruder.io/a-review-of-the-recent-history-of-nlp/

Recent Trends



- Character-based representations
- From Recurrent Neural Networks, to Convolutional Neural Networks, to the Transformer [Vaswani et al., 2017]
- Pre-trained Language Models: BERT [Devlin et al, 2018]
- Transfer Learning: domain adaptation, cross-lingual learning, multi-task learning, …
- Adversarial Learning
- Reinforcement Learning