U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

LIACC

MIEIC
EIC0029 – Artificial Intelligence

Natural Language Processing

Henrique Lopes Cardoso

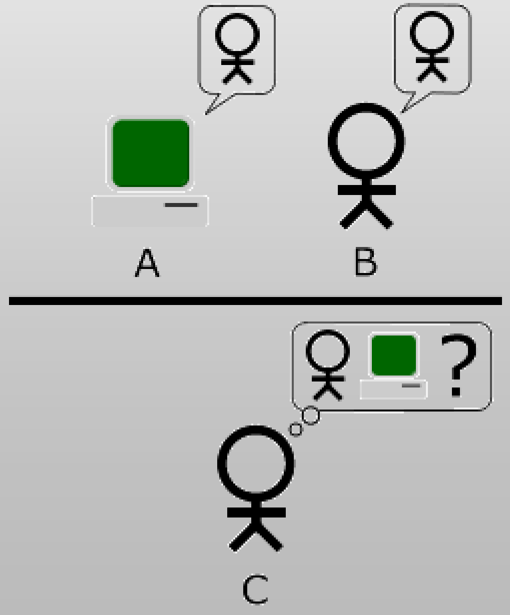
hlc@fe.up.pt

April 2020

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		Responses: *Retrieval-Based*	Responses: *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

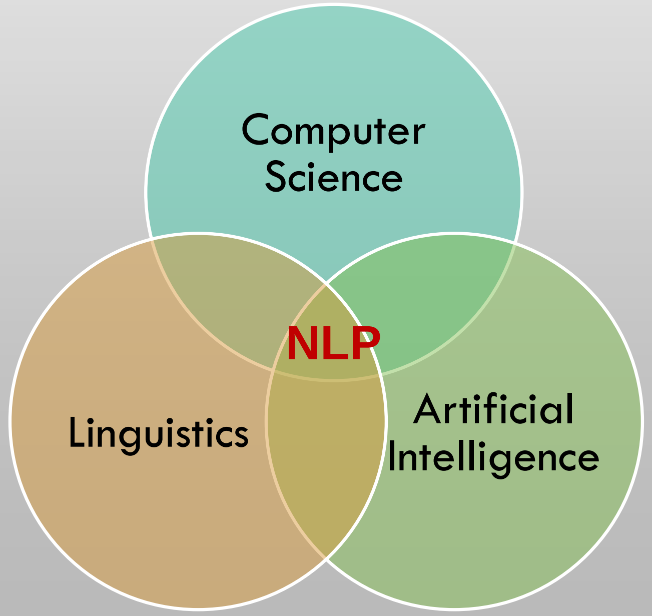
U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

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

NLP Tasks

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

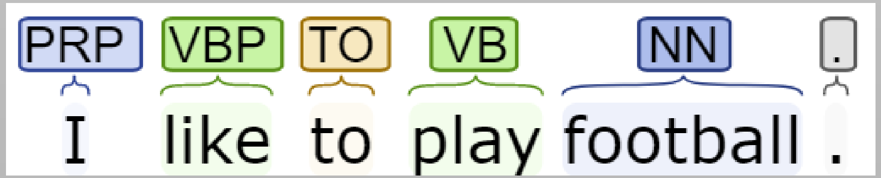
- A few of the most relevant tasks:
  - Tokenization
    - Split a sentence into tokens (words)

```
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

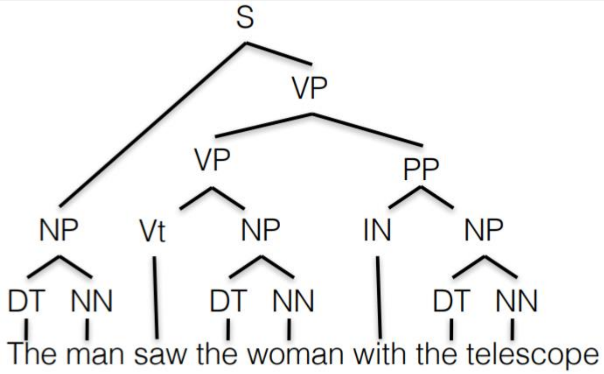
```
Hello. Are you Mr. Smith? I've finished my M.Sc. on Informatics!
['Hello.',
 'Are you Mr. Smith?',
 'I've finished my M.Sc. on Informatics!']
```

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



NLP Tasks

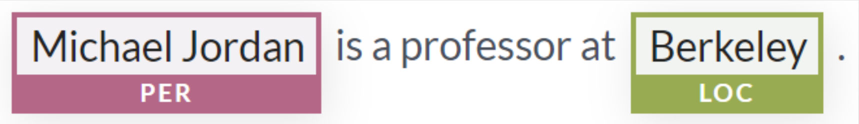
- Syntax parsing
  - Determine the parse tree (grammatical analysis) of a sentence



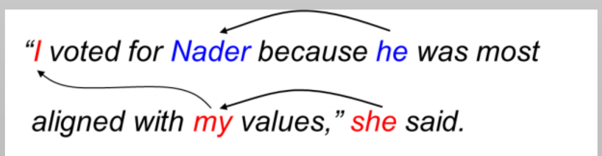
- Word sense disambiguation
  - Select the meaning of words in a context

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, …
  - 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
- …

NLP Applications

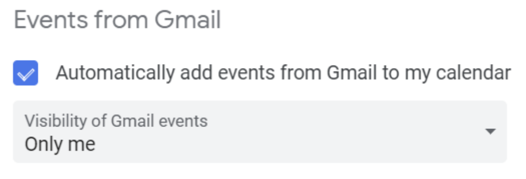
- Machine Translation
  - Based on multilingual textual corpora
  - Text translation and multilingual real-time conversations

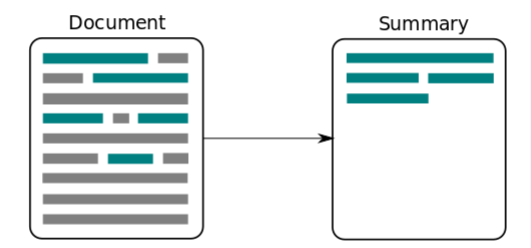
- Sentiment Analysis and Opinion Mining
  - Determine polarity about specific topics
  - Identify trends of public opinion in social media
  - Analyze product reviews

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

NLP Applications

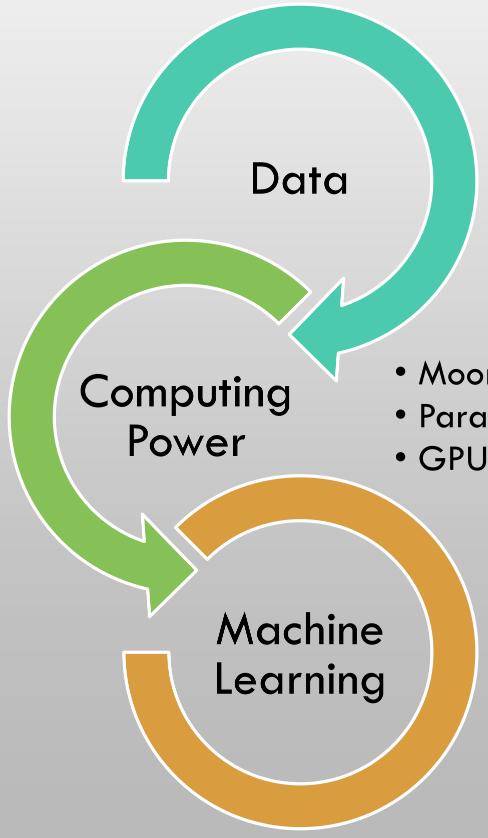
- Information Extraction
  - Extract relevant entities from text
  - Event identification, “add to calendar” features
- Question Answering
  - IBM Watson won *Jeopardy!* on 2011
- Text Summarization
  - Build a summary of a long text
- Argument Mining and Debate Portals
  - Extract arguments that expose a certain position
  - Aggregate pros and cons for a debatable topic





Statistical NLP

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



- Web data
- Corpora
- Crowdsourcing

- Moore's law
- Parallelism
- GPUs

- Supervised
- Unsupervised
- Deep-learning

Machine Learning in NLP

Data → Feature Engineering → ML Algorithm → Model

Data
- Preparation
- Learning instances

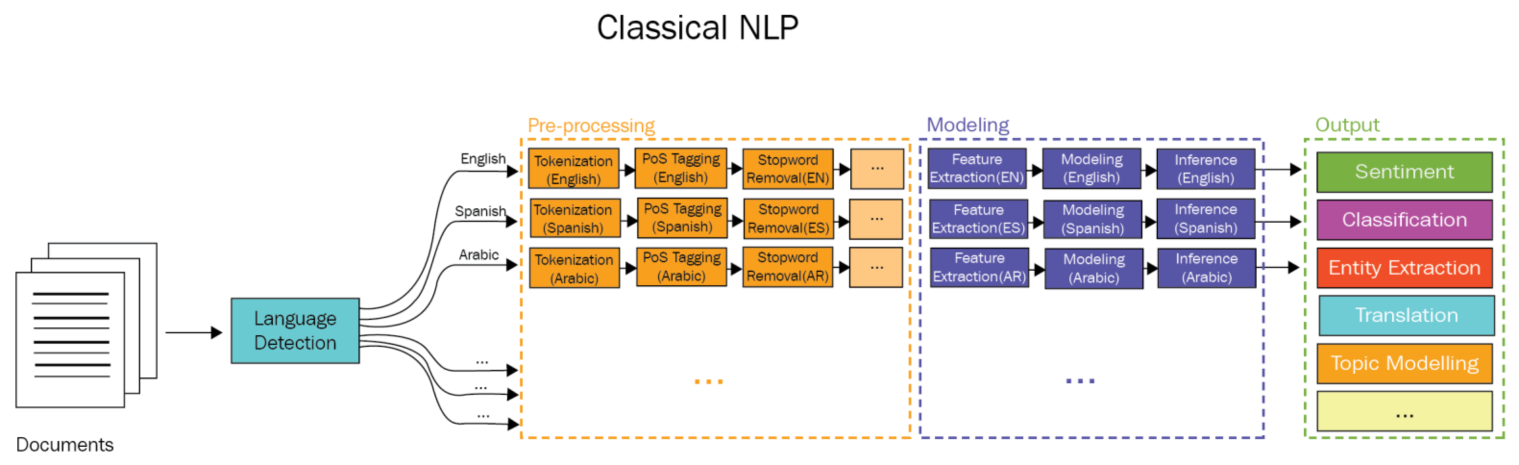
Feature Engineering
- Definition
- Extraction
- Selection

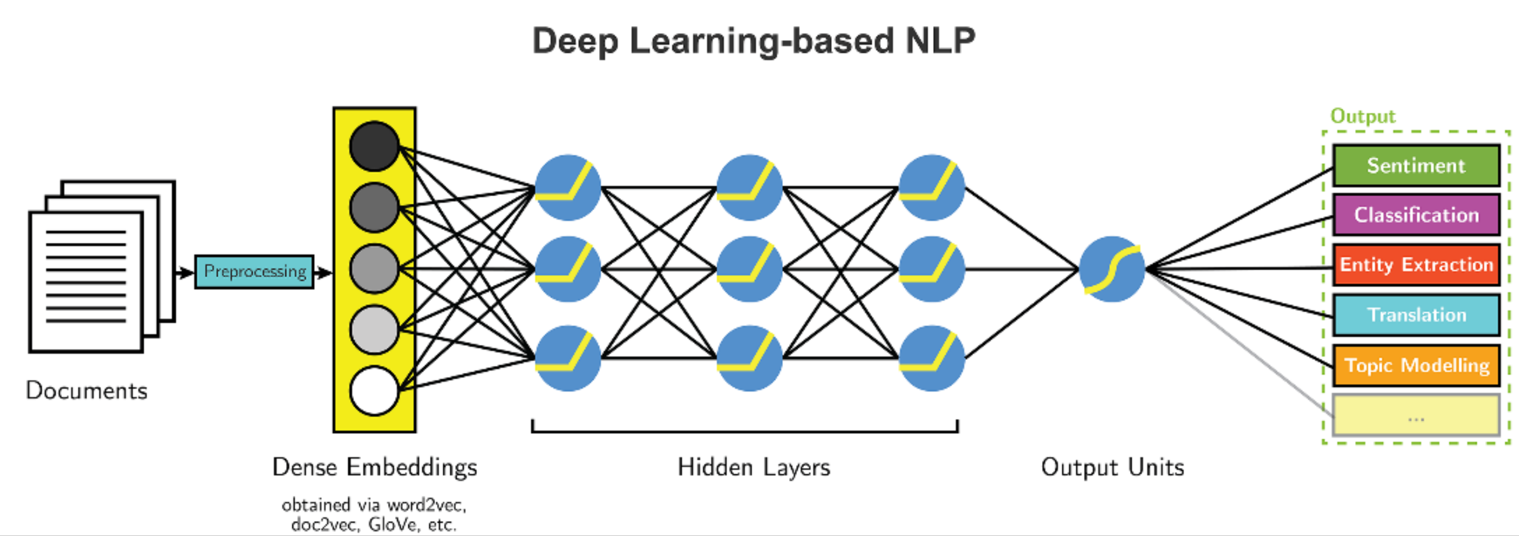
ML Algorithm
- NB, SVM
- MLP
- CRF, HMM

Model
- 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, …

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 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

Basic Text Processing

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

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

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

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 missis 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

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”
  - Morphology: splitting a word into 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
    - An 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 $C = \{c_1, c_2, \dots, c_m\}$
  - 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 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

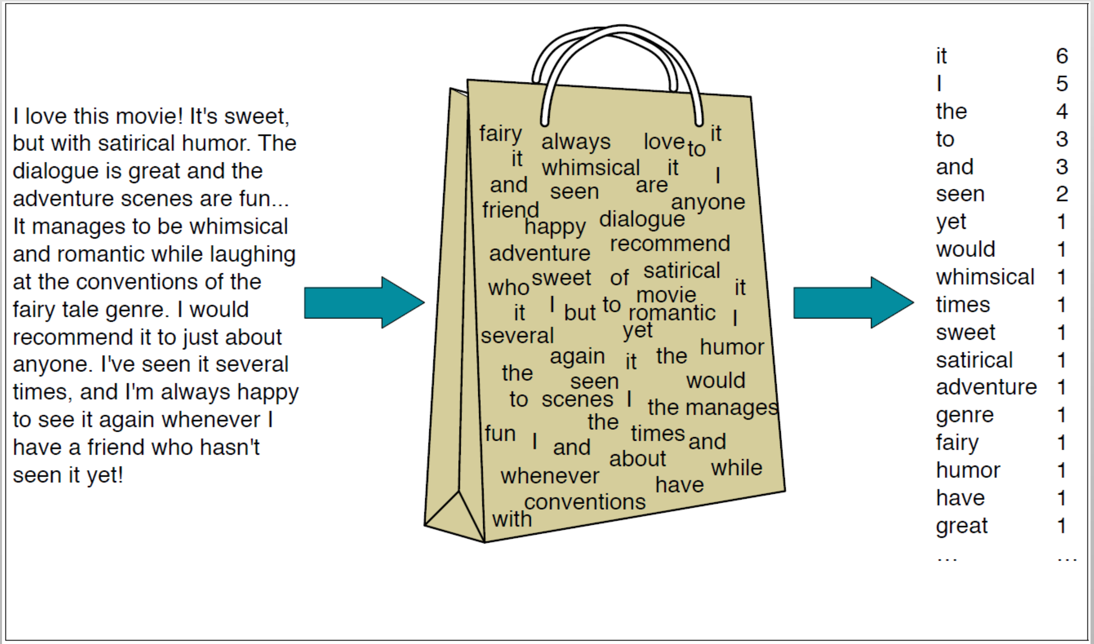
- Probabilistic classifier: more than predicting a class, outputs the probability of the observed document belonging to each of the classes

- 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

- Bays rule:

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

- Most likely class:

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

- 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 is available
- 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 +	NB Counts −	Binary 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

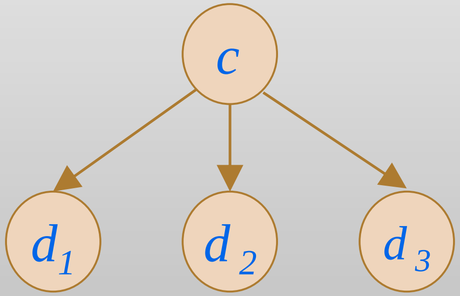
- Lexicons provide external knowledge that can be very useful for the task!
- Sentiment lexicons
  - Lists of words that are pre-annotated with positive or negative polarity
  - Example: MPQA Subjectivity Lexicon
    - 6885 words, 2718 positive and 4912 negative, strongly or weakly biased
    - + : admirable, 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

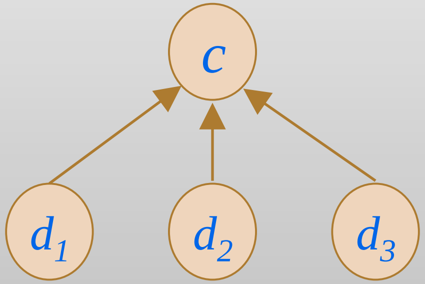
- A generative model makes use of this likelihood term: how to generate the features of a document if we knew it was of class c?



- Example: Naïve Bayes

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

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



- Example: Logistic Regression

Generative vs Discriminative Classifiers

- NB 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

- Logistic Regression generally works better on larger documents or datasets

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

Modern NLP

word embeddings, deep learning

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

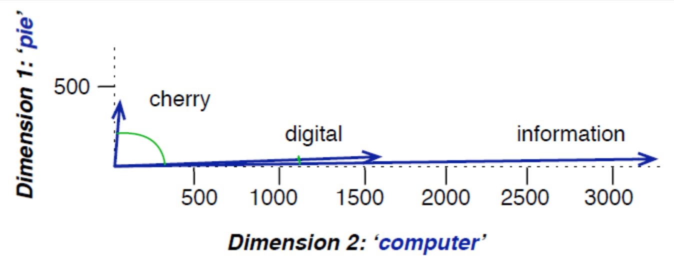
- We can represent the meaning of a word based on the contexts in which it occurs
  - Unsupervised approach: observe word usage on large (non-annotated) corpora
  - Sparse vectors (|V| = 20k? 50k?): word counts, or TF-IDF
  - Dense vectors: short vectors (50-1000 real numbers), most values non-zero
    - Trained using algorithms such as skipgram (word2vec)

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

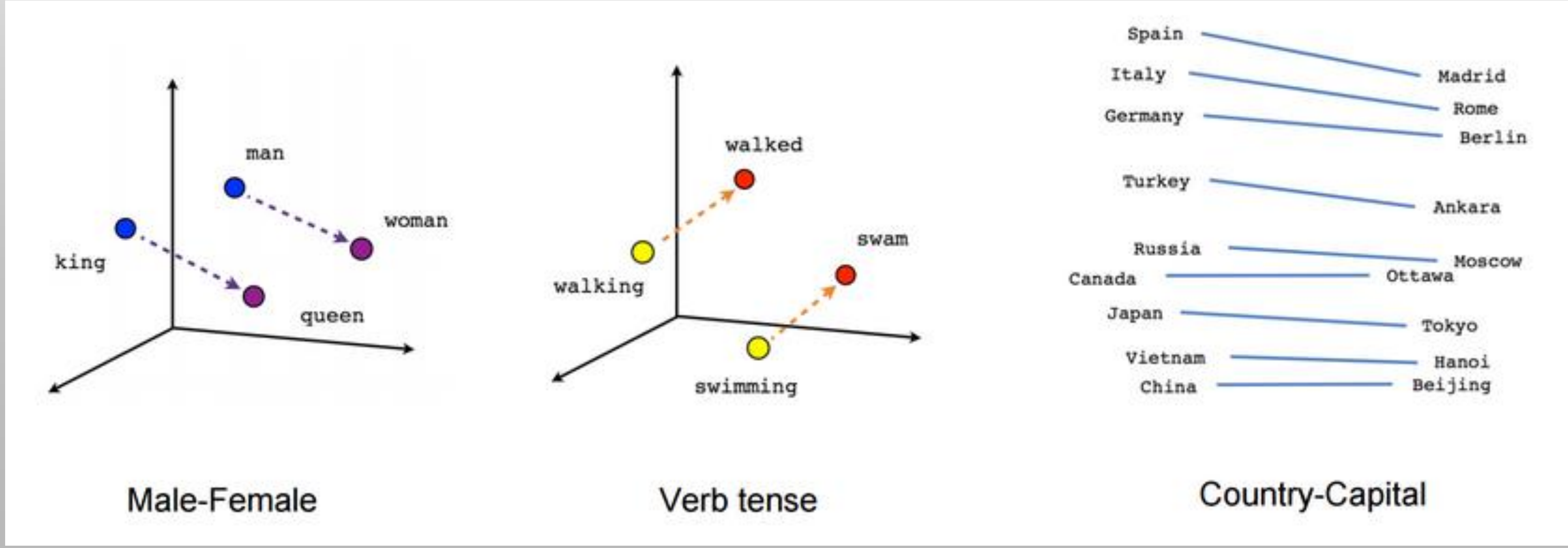
Word Embeddings

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

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

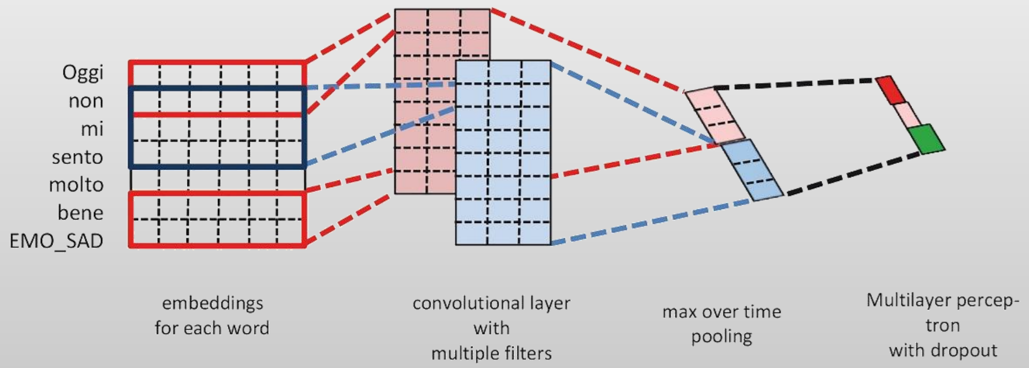


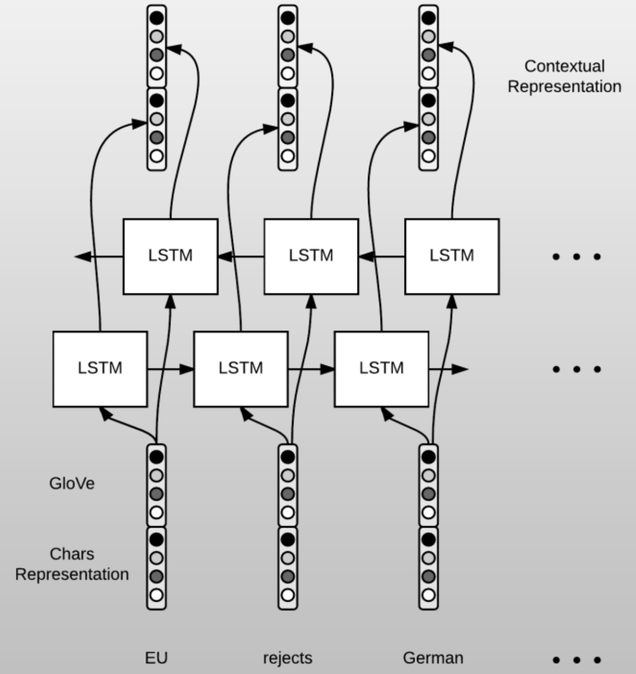
- Relational properties in the vector space



Deep Learning in NLP

- Convolutional Neural Networks





- Recurrent Neural Networks

- … and many other advanced architectures, using language models, attention, transformers, …