

Introduction to Natural Language Processing



Master 2 Droit du Numérique IA

Chloé Braud, CNRS - IRIT, chloe.braud@irit.fr

25 avril 2024

NLP is used for many applications to mine text

- spam filtering
- spell checking
- automatic translation
- finding information in text / databases
- finding entities and linking them
- classifying text, e.g. topic, fake news ...
- analysing opinion / sentiment in text
- summarizing text
- simplifying text
- profiling users
- detecting mental illness
- ...



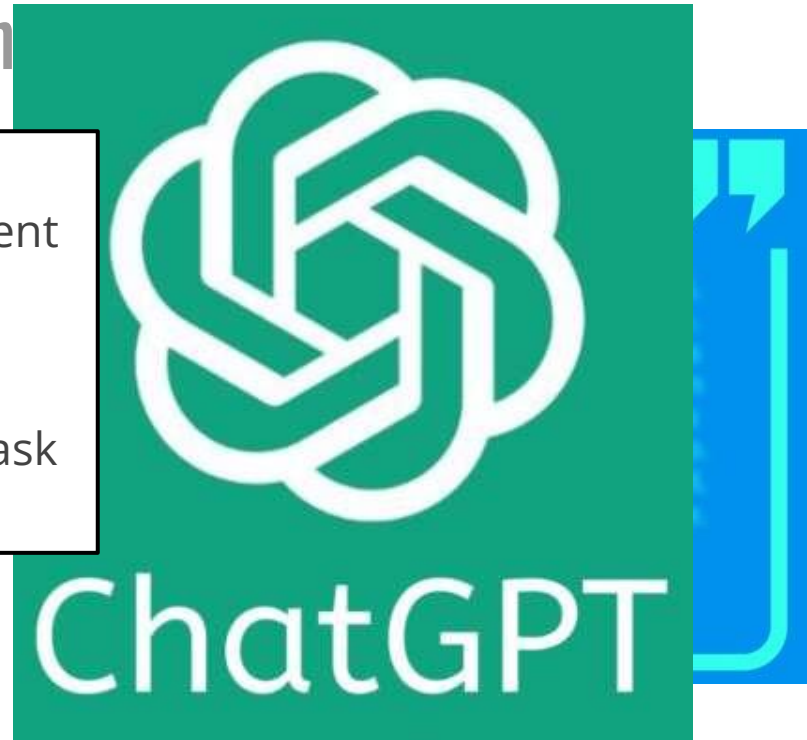
More and more companies and other academic fields are interested in NLP techniques

NLP is used for many applications

Released of ChatGPT:

- seems able to solve any problems = fluent generation, able to summarize, answer questions, multilingual etc
 - prompt engineering: find the good questions to ask in order to solve any task
- is NLP solved?

- profiling users
- detecting mental illness
- ...



More and more companies and other academic fields are interested in NLP techniques

Is it still interesting to take a Natural Language Processing course?

Let's ask Gemini (in English and French):

<https://gemini.google.com/app/b9ec5350707b545e> (en)

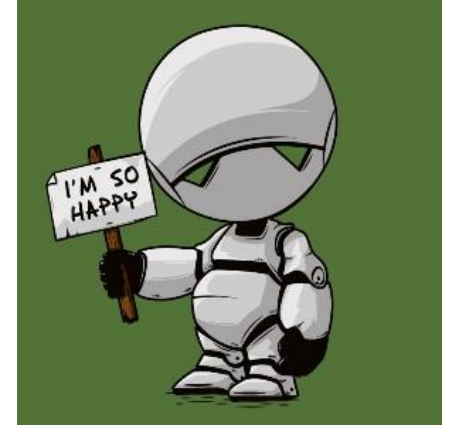
<https://gemini.google.com/app/6434bd5f7295aa22> (fr)

→ Still many challenges, according to Gemini:

<https://gemini.google.com/app/1023a063edb036d0> (en)

<https://gemini.google.com/app/b9ec5350707b545e> (fr)

- Generative AI excels at creating human-like text
- NLP provides the foundation for generative AI to understand language
- They work together
- Challenges: more natural conversations, word knowledge, biases, evaluation...



Is it still interesting to take a Natural Language Processing course?

Let's ask Gemini (in English and French):

<https://gemini.google.com/app/b9ec5350707b545e> (en)

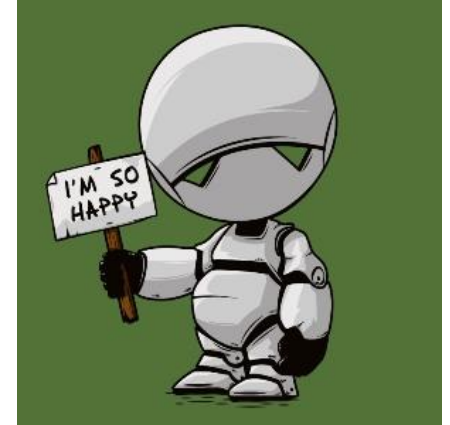
<https://gemini.google.com/app/6434bd5f7295aa22> (fr)

→ Still many challenges, according to Gemini:

<https://gemini.google.com/app/1023a063edb036d0> (en)

<https://gemini.google.com/app/b9ec5350707b545e> (fr)

- Generative AI excels at creating human-like text
- NLP provides the foundation for generative AI to **understand language**
- They work together
- Challenges: more natural conversations, word knowledge, biases, evaluation...



Is it still interesting to take a Natural Language Processing course?

Let's ask Gemini (in English and French):

<https://gemini.google.com/app/b9ec5350707b545e> (en)

<https://gemini.google.com/app/6434bd5f7295aa22> (fr)

→ Still many challenges, according to Gemini:

<https://gemini.google.com/app/1023a063edb036d0> (en)

<https://gemini.google.com/app/b9ec5350707b545e> (fr)

- **Generative AI** excels at creating human-like text
- **NLP** provides the foundation for generative AI to **understand language**
- They work together
- Challenges: more natural conversations, word knowledge, biases, evaluation...



Is it still interesting to take a Natural Language Processing course?

Let's ask Gemini (in English and French):

<https://gemini.google.com/app/b9ec5350707b545e> (en)

<https://gemini.google.com/app/6434bd5f7295aa22> (fr)

→ Still many challenges, according to Gemini:

<https://gemini.google.com/app/1023a063edb036d0> (en)

<https://gemini.google.com/app/b9ec5350707b545e> (fr)

- **Generative AI** excels at creating human-like text
- **NLP** provides the foundation for generative AI to **understand language**
- They work together
- **Challenges:** more natural conversations, word knowledge, biases, evaluation...



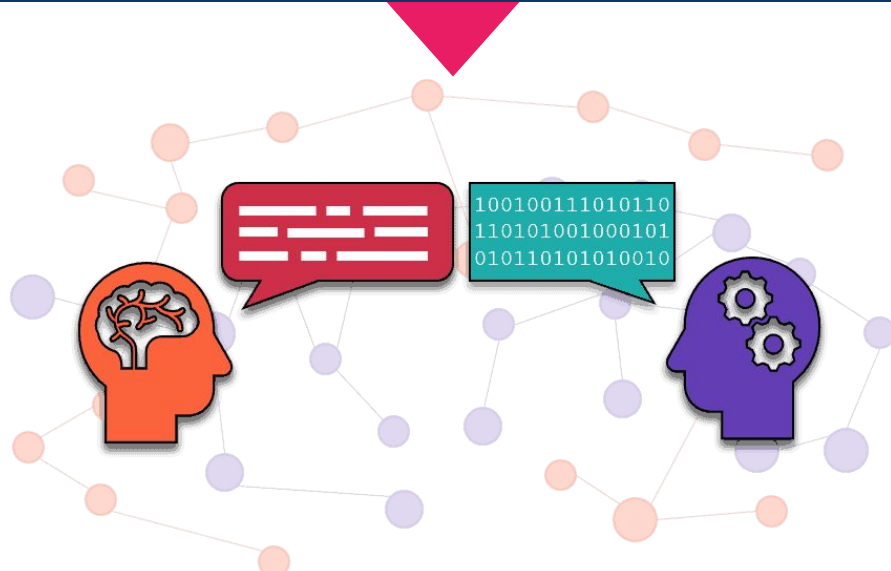
Content of this course: NLP basics

1. What is Natural Language Processing?
2. (Computational) Linguistics basics
3. Learning from textual data
4. Main applications of NLP
5. Practical NLP: finding data, machine learning libraries, learning settings
6. Limits and current challenges of NLP

→ Practical session:

- pre-processing text data ;
- exploring language models: bias, classification, explainability

What is Natural Language Processing?



What is Natural Language Processing?

At the **interface between:**

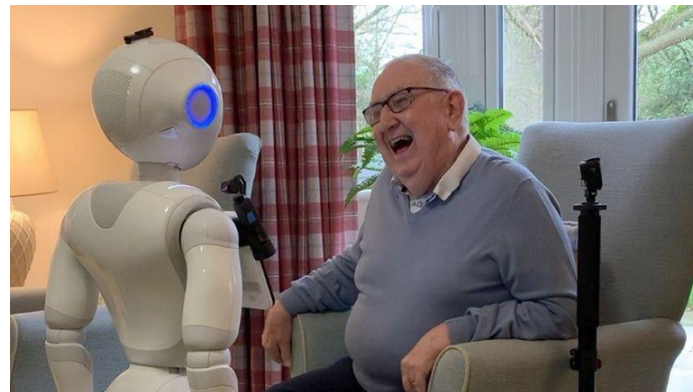
- Linguistics: dealing with natural languages
- Computational Science: algorithms
- Mathematics: statistics, probabilities
- Physics: speech processing

Subdomain of Artificial Intelligence:

- human machine interaction
- human human interaction

= access to information / extract information

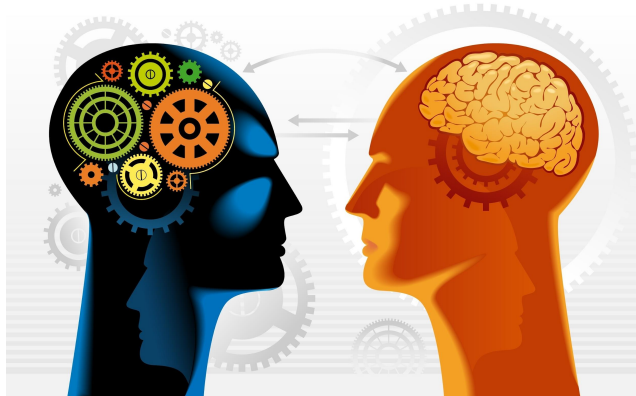
= understanding intelligence through communication



What is Natural Language Processing?

Computational techniques to process data in natural language

- **Practical:** Creating tools to automatically process data in natural language, i.e. tools to analyse, model, “understand”, generate, save language
- **Theoretical:** Using empirical methods to better understand what is language: how does it work? how people understand each other?

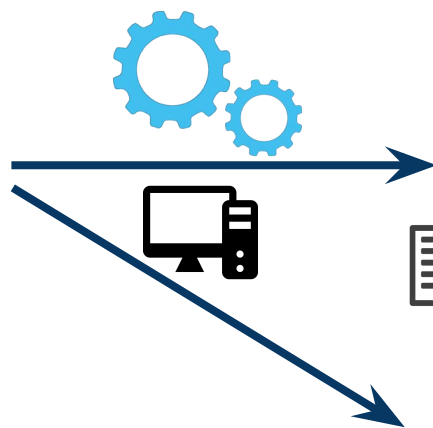


What is Natural Language Processing?

Take data

Process

Develop applications



Understand language

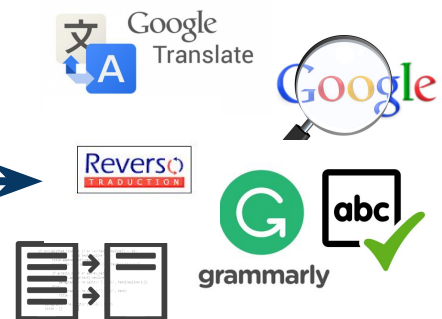
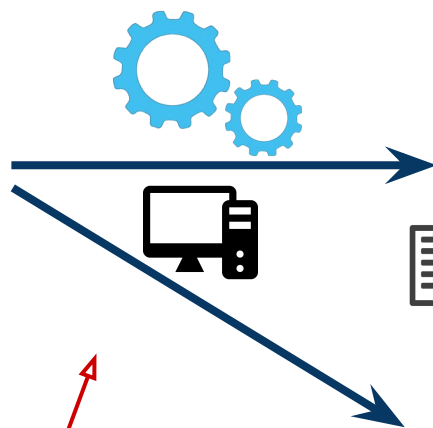


What is Natural Language Processing?

Take data

Process

Develop applications



Understand language



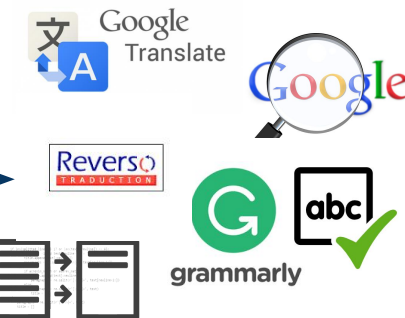
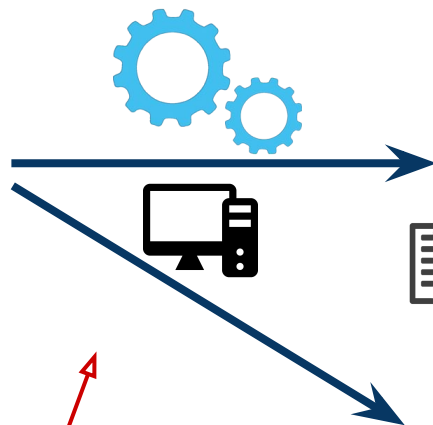
What does it mean?
How do we do that?

What is Natural Language Processing?

Take data

Process

Develop applications



Understand language



Many challenges!

What does it mean?
How do we do that?

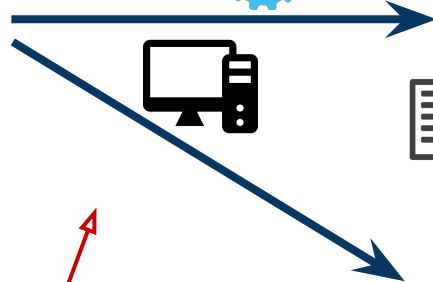
What is Natural Language Processing?

Take data

Process

Develop applications

Which ones?



Understand language



Many challenges!

What does it mean?
How do we do that?

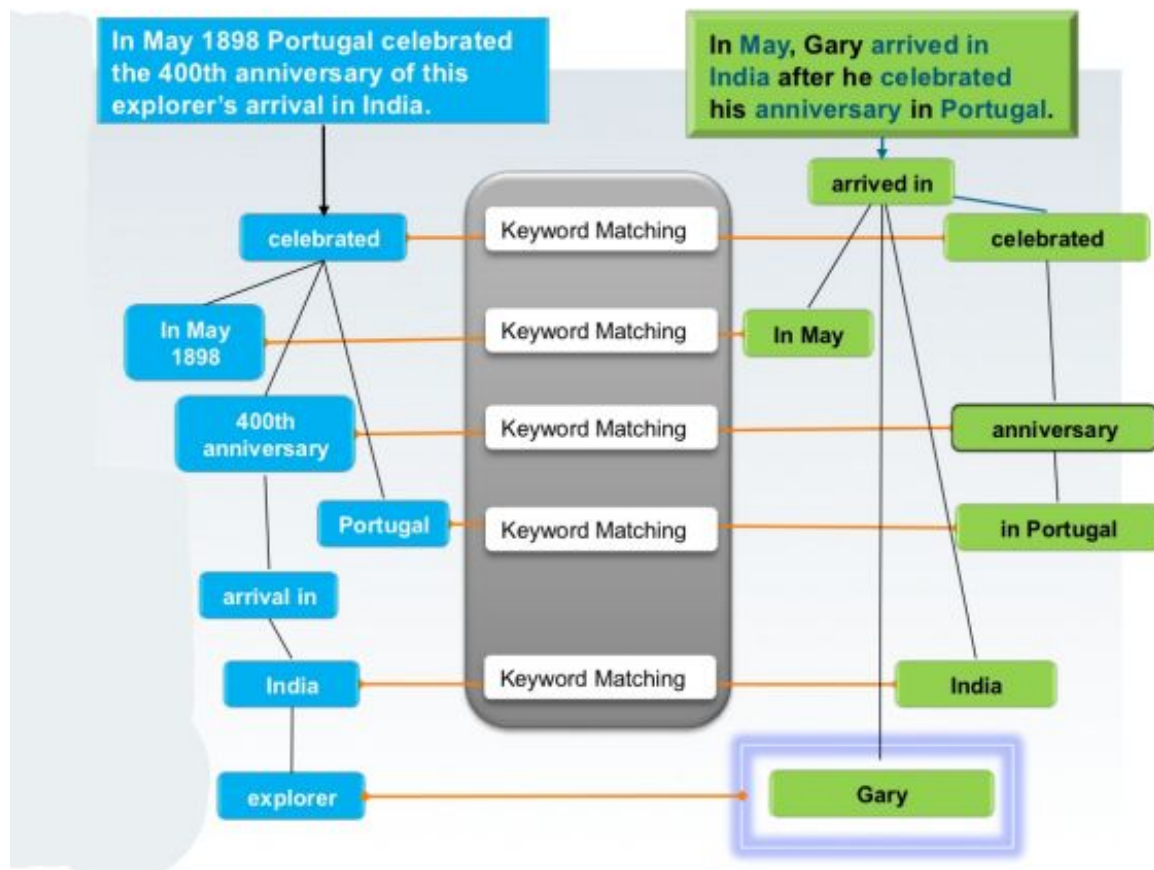
Why NLP is hard?

Example: Question-Answering with Watson



Why NLP is hard?

Keyword matching can provide good clues
→ but here, misleading

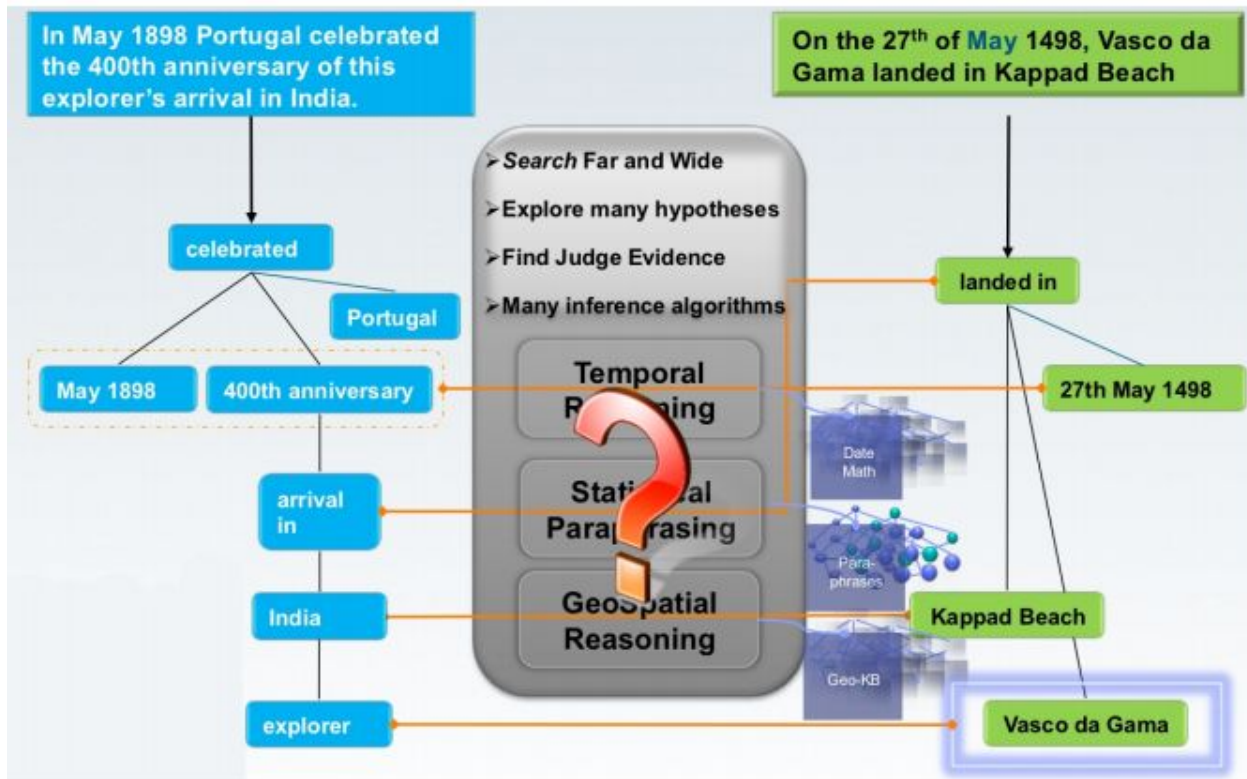


Why NLP is hard?

It won't be enough most of the time:

→ stronger evidence are much harder to find, relying on:

- fine-grained analysis (e.g. temporal reasoning)
- complex understanding (e.g. paraphrasing)

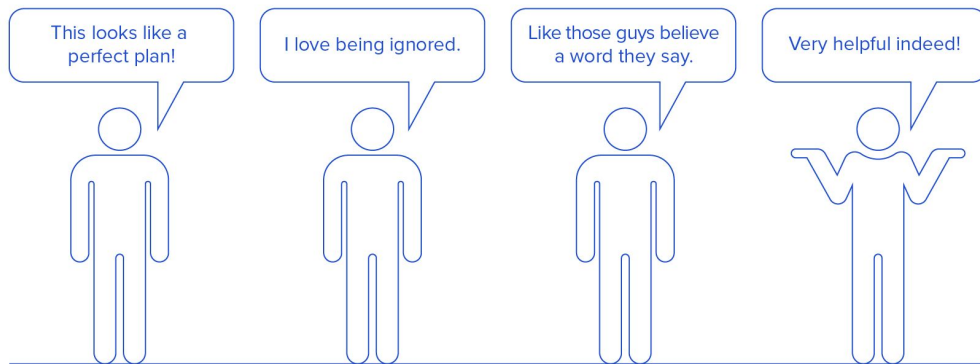


Why NLP is hard?

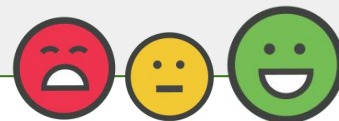
Many issues:

- ambiguity
- coreferent entities
- equivalent forms
- sarcasm
- negation, contrast
- ...

Coronavirus quickly spread worldwide in 2020. The virus mostly affects elderly people. They can easily catch it.



This mouse **is not good** looking, but it works **perfect** and I **like** it.



Why NLP is hard?

Other issues:

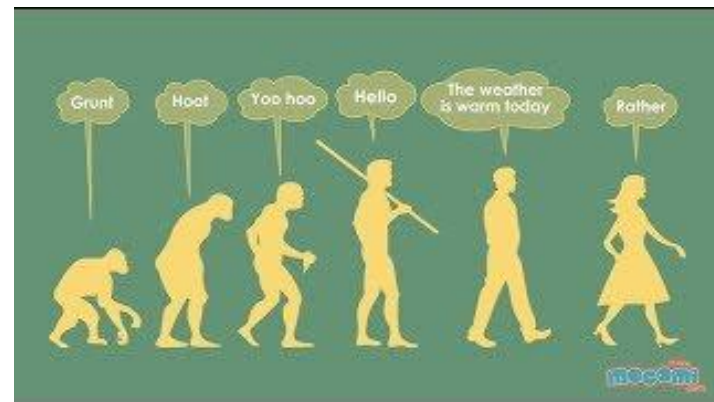
- language is evolving
- speakers do not always master their own language...

- speakers are creative:

- e.g. lol, mdr, yolo, omg ...
- e.g. cheeeeeerrrsss
- e.g. love u
- e.g. <3 :) :D

+ of course: many languages with variations across domains, socio-economic groups ...

→ Makes the study of language fascinating but its automatic processing harder (and funnier ;)



DaBaddest BitchAroun
@ATL_PRINCESS



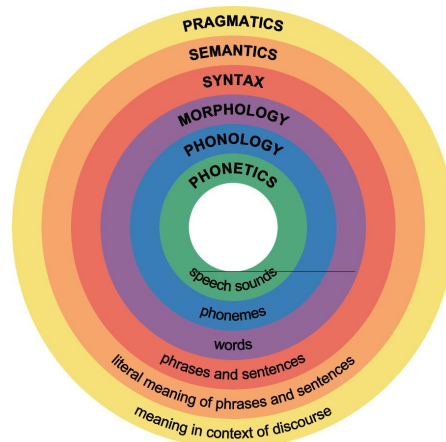
Sometimes i snap at ppl on twitter bcuz im insecure. Its a defense magnesium



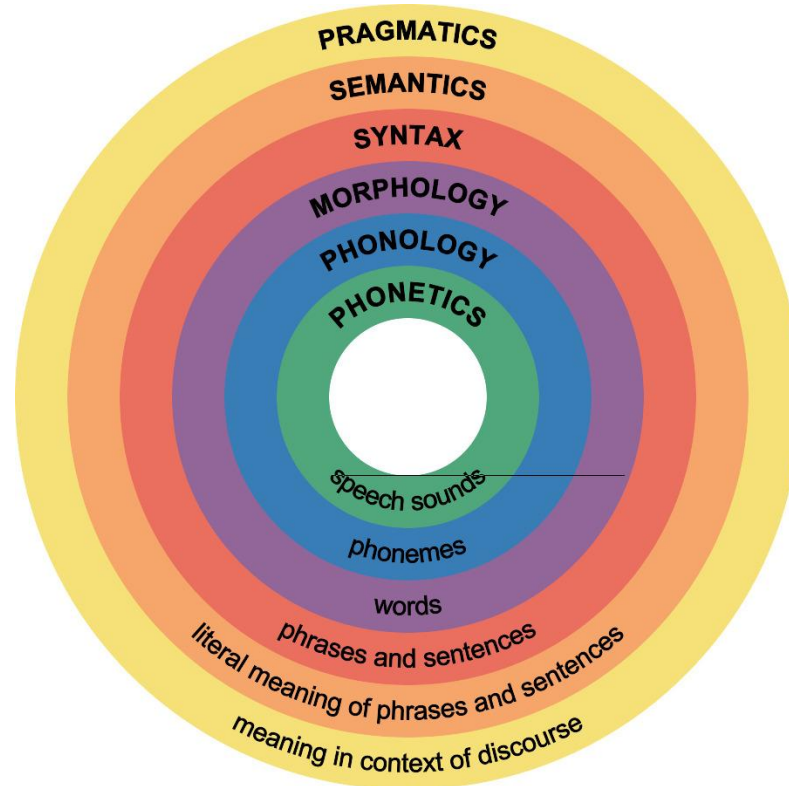
Donald J. Trump
@realDonaldTrump

After ^{ing} having ^g written many best-selling books, ^g and somewhat priding myself on my ability to write, it should be noted that the Fake News constantly likes to ^g pour ^g over my tweets looking for a ^g mistake. I capitalize certain words ^g only for emphasis, not b/c they should be capitalized!

Computational Linguistics Basics



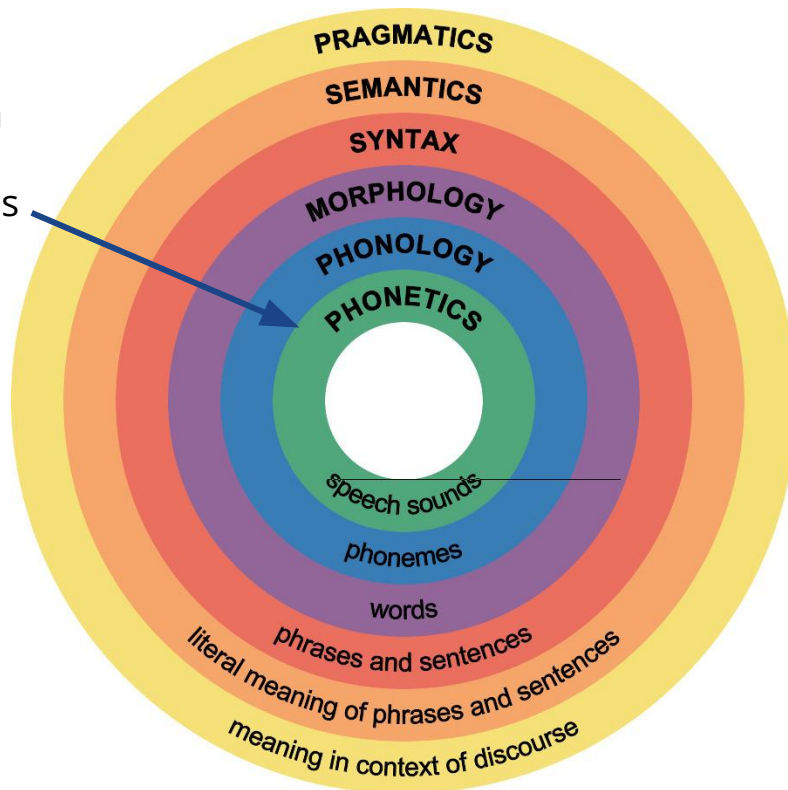
Main levels of linguistics structure



Main levels of linguistics structure

Phonetics-Phonology:

Speech processing is distinct from NLP = focus on text. But NLP researchers work on transcriptions from speech.



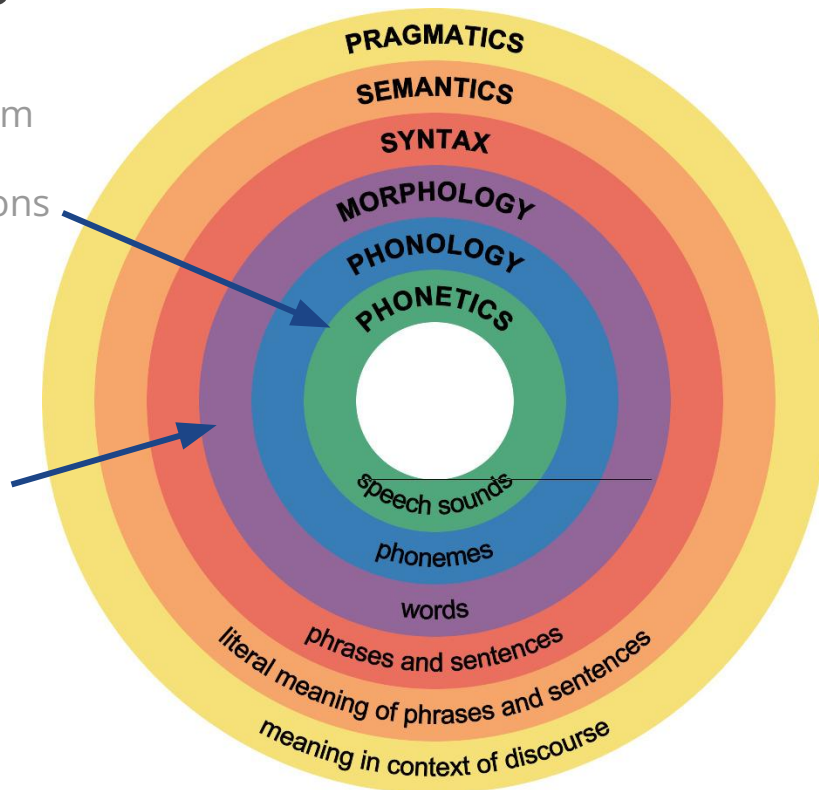
Main levels of linguistics structure

Phonetics-Phonology:

Speech processing is distinct from NLP = focus on text. But NLP researchers work on transcriptions from speech.

Morphology:

How words are built. Studies on new words creation: people are very creative!



Main levels of linguistics structure

Phonetics-Phonology:

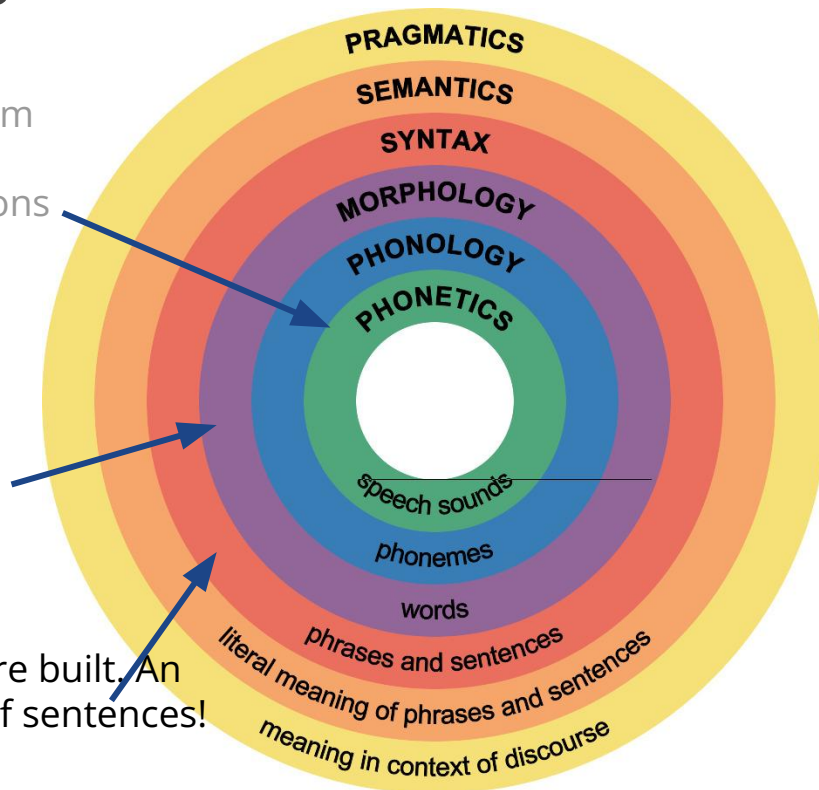
Speech processing is distinct from NLP = focus on text. But NLP researchers work on transcriptions from speech.

Morphology:

How words are built. Studies on new words creation: people are very creative!

Syntax:

How sentences are built. An infinite number of sentences!



Main levels of linguistics structure

Phonetics-Phonology:

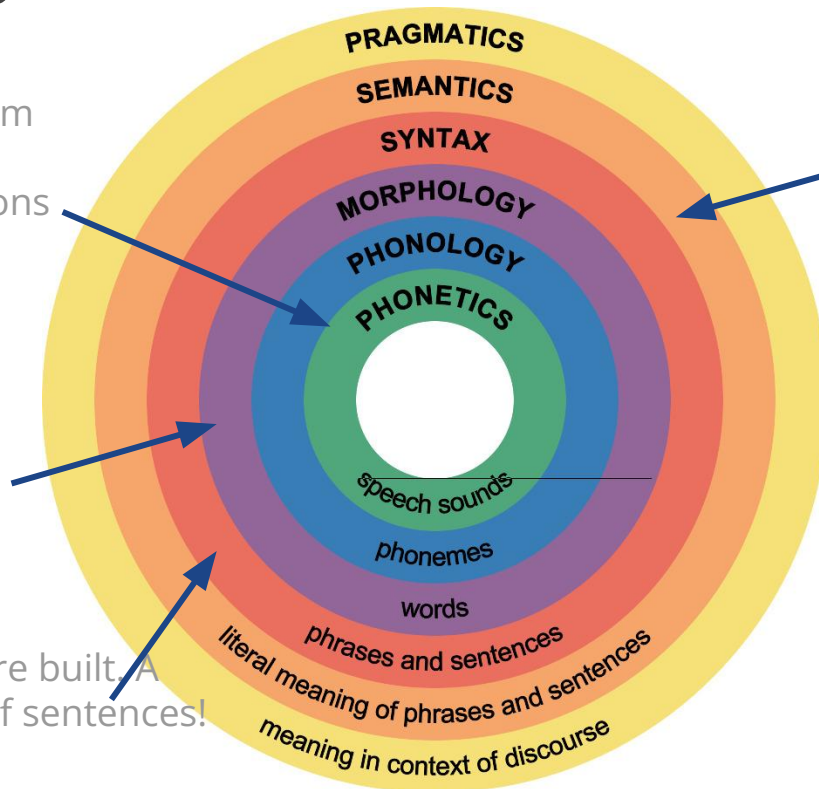
Speech processing is distinct from NLP = focus on text. But NLP researchers work on transcriptions from speech.

Morphology:

How words are built. Studies on new words creation: people are very creative!

Syntax:

How sentences are built. A infinite number of sentences!



Semantics:

Literal meaning of the words and utterances. What do you mean?

Main levels of linguistics structure

Phonetics-Phonology:

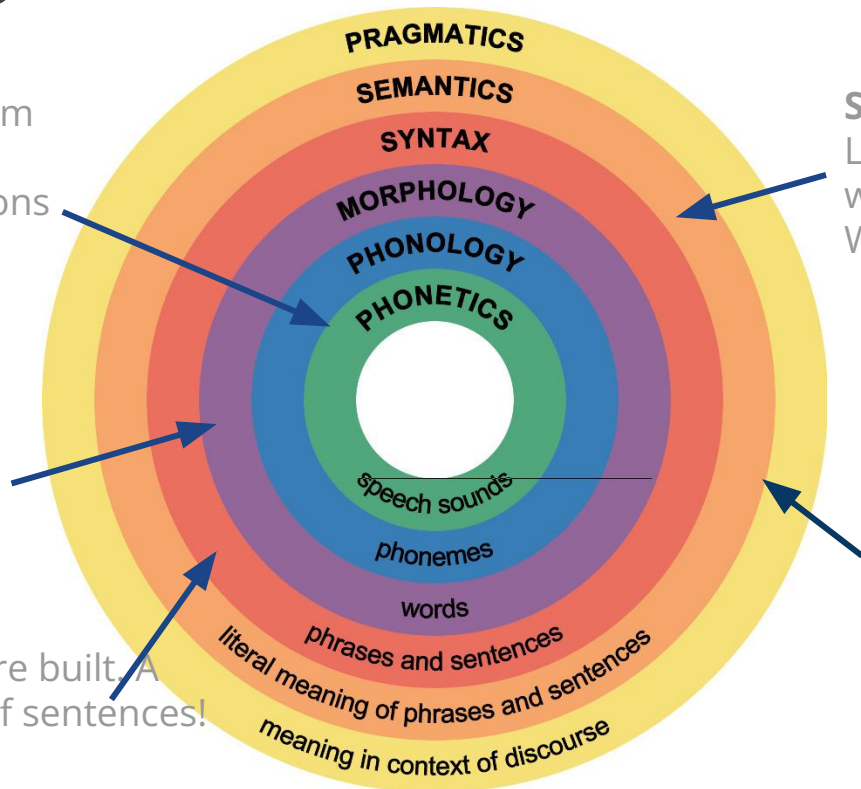
Speech processing is distinct from NLP = focus on text. But NLP researchers work on transcriptions from speech.

Morphology:

How words are built. Studies on new words creation: people are very creative!

Syntax:

How sentences are built. A infinite number of sentences!



Semantics:

Literal meaning of the words and utterances. What do you mean?

Pragmatics:

Meaning in context. What do you really mean? How can we understand each other?

Main levels of linguistics structure

Phonetics-Phonology:
Speech processing
NLP = focus on text
researchers work
from speech.

Morphology:
How words are built
new words creation
very creative!

Syntax:
How words are
inflected

Lexical Ambiguity

The presence of two or more possible meanings within a single word.



"I saw her duck."

Syntactic Ambiguity

The presence of two or more possible meanings within a single sentence or sequence of words.



"The chicken is ready to eat."

Semantics:
Meaning of the
word utterances.
What do you mean?

Pragmatics:
Meaning in context. What
do I really mean?
How can we understand
each other?

Pre-processing text data

Basic operations / pre-processing:

- tokenisation
- sentence segmentation
- normalization:
 - lower-casing
 - lemmatization, stemming
 - removing stop-words



- current models often include some preprocessing steps (e.g. tokenisation) but:
- still others needed (e.g. sentence split)
 - useful with different architectures / rule-based models
 - **important to understand what is done** and what we expect.

Words / Tokens

Text = sequences of characters, spaces, punctuation encoded the same way

→ We need to find the words boundaries = **tokenization**

→ Crucial: words are meaningful units, minimal input to NLP systems

What is a word? = sequence of characters separated with a blank (space, line) or punctuation?



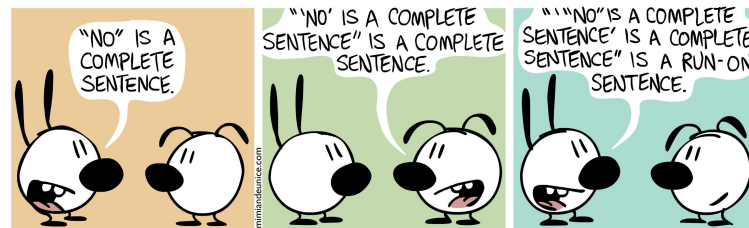
- apostroph: *I don't know* = 3 or 4 words?
- Multiword expressions: *in spite of; fleur bleue; perdre la tête* = how many words?
- Proper nouns: *San Francisco, Tour Eiffel*
- Other languages:
 - German: *rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz* = "the law for the delegation of monitoring beef labeling."
 - no word boundaries explicitly marked for some languages (= word segmentation)

Sentence: 这是一篇有趣的文章

Words: 这是 一篇 有趣 的 文章

(zhèshì yīpiān yǒuqù de wénzhāng)
(This is an interesting article)

Sentences



Text = sequences of characters, spaces, punctuation encoded the same way

→ We need to find the sentence boundaries = **sentence segmentation**

→ Crucial: Sentences are the input of many systems; Syntactic analysis is based on sentences

- **'** is very ambiguous, a period may denote:
 - an abbreviation (47% of the periods in the Wall Street Journal),
 - decimal point,
 - an ellipsis,
 - an email address
- **?** and **!** are less ambiguous (in English), but they may appear in embedded quotations, emoticons, computer code, and slang

In Oct. 2013, M. Obama will visit Paris. He will meet ministers, deputies, parliamentarians etc. to discuss.

→ Context is crucial to find sentence boundaries

Sentences



Text = sequences of characters, spaces, punctuation encoded the same way

→ We need to find the sentence boundaries = **sentence segmentation**

→ Crucial: Sentences are the input of many systems; Syntactic analysis is based on sentences

- **'** is very ambiguous, a period may denote:
 - an abbreviation (47% of the periods in the Wall Street Journal),
 - decimal point,
 - an ellipsis,
 - an email address
- **?** and **!** are less ambiguous (in English), but they may appear in embedded quotations, emoticons, computer code, and slang

In **Oct.** 2013, **M.** Obama will visit Paris. He will meet ministers, deputies, parliamentarians **etc.** to discuss.

→ Context is crucial to find sentence boundaries

Normalization

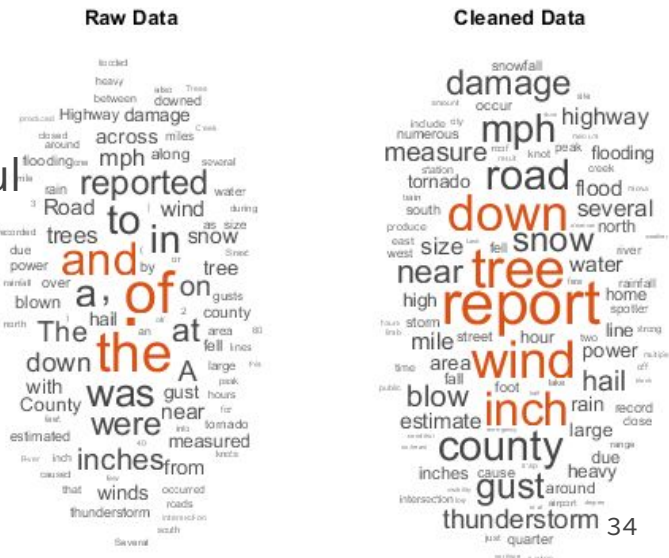
Often, we do some normalization on words, to improve generalization:

- lower-casing: **“The”** → **“the”**
- **lemmatization** = reduce to “base form”, remove inflectional endings
 - e.g. **“managed”** → **“manage”**,
 - e.g. **“is/was/were/being”** → **“be”**
 - → introduce ambiguity: **“changed”** → **“change”** = Verb or Noun
 - → no clear base form for some words
 - e.g. pronouns: in Spacy **“I”** → **“PRON”**
- removing **stop-words** = very frequent, less meaningful

words e.g. **“i, me, the, to, when, ...”**

Many lists, see e.g. list in NLTK: <https://www.nltk.org/book/ch02.html>

Less useful for new models, partly included: ‘cased’ or ‘uncased’
Language Models ; subwords tokenization for unknown words.



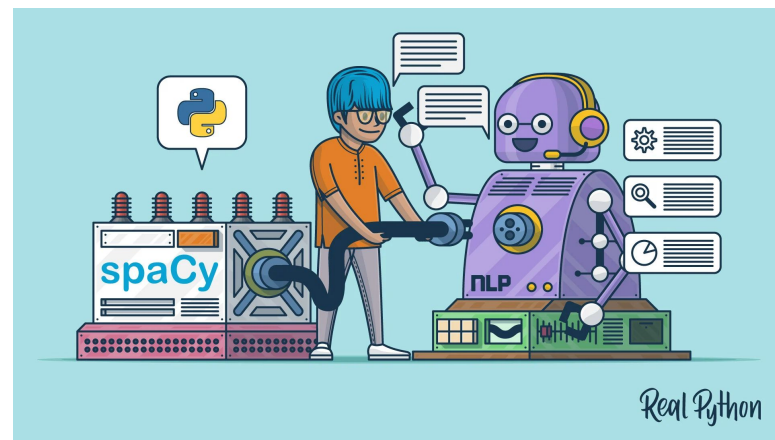
Linguistic Analysis

Additional processing that can provide rich information:

- Part-of-Speech (POS) tagging
- Word sense disambiguation
- Named Entity recognition
- Syntactic parsing
- Discourse parsing

→ statistical models

→ Varied performance, can be very hard especially for high-level tasks (i.e. semantics) and low-resources languages / domains.



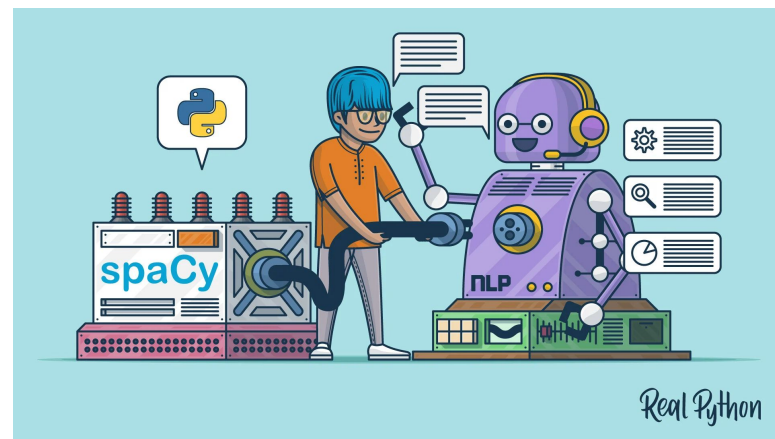
Linguistic Analysis

Additional processing that can provide rich information:

- **Part-of-Speech (POS) tagging**
- Word sense disambiguation
- **Named Entity recognition**
- **Syntactic parsing**
- Discourse parsing

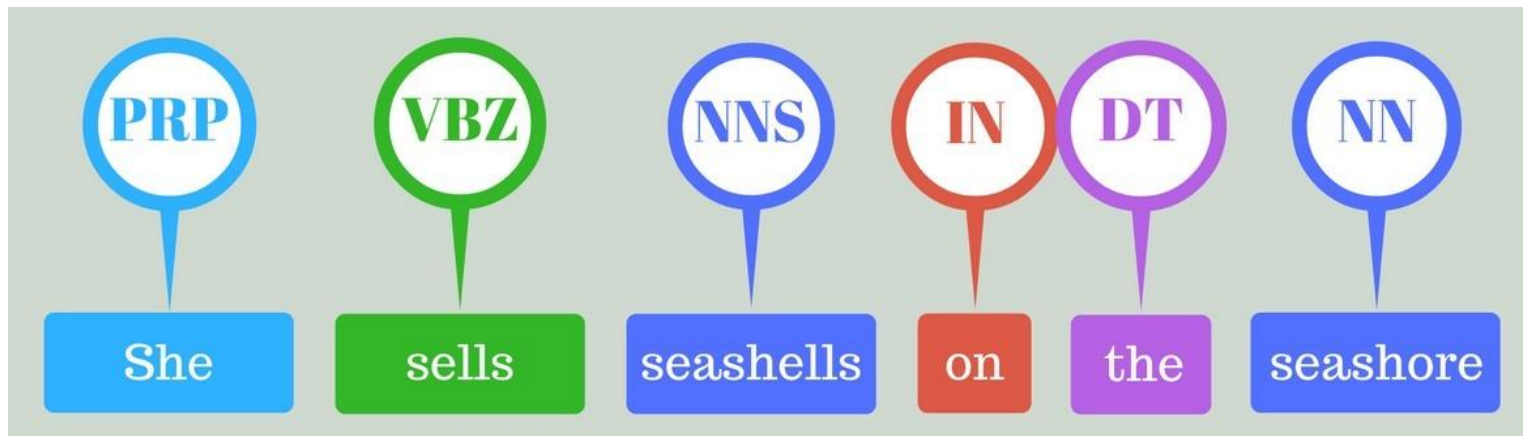
→ statistical models

→ Varied performance, can be very hard especially for high-level tasks (i.e. semantics) and low-resources languages / domains.



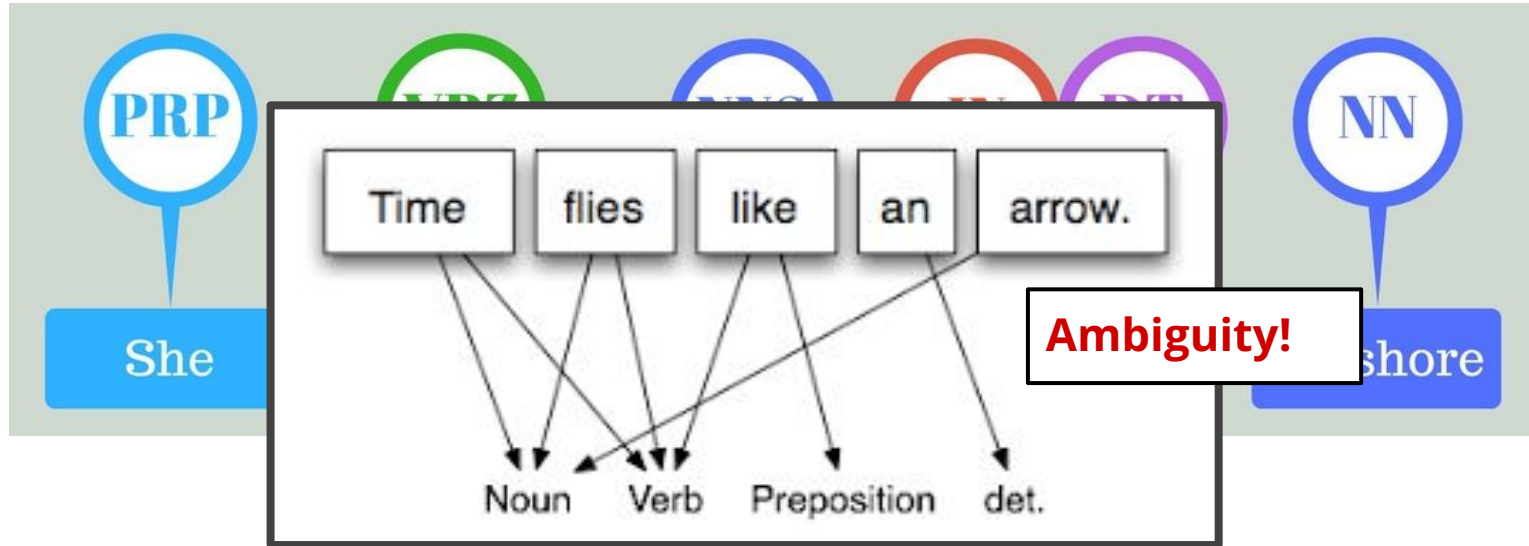
Part-Of-Speech Tagging

= Associating a morpho-syntactic category to each word
→ Noun, Verb, Pronoun, Adjective, Adverb ...



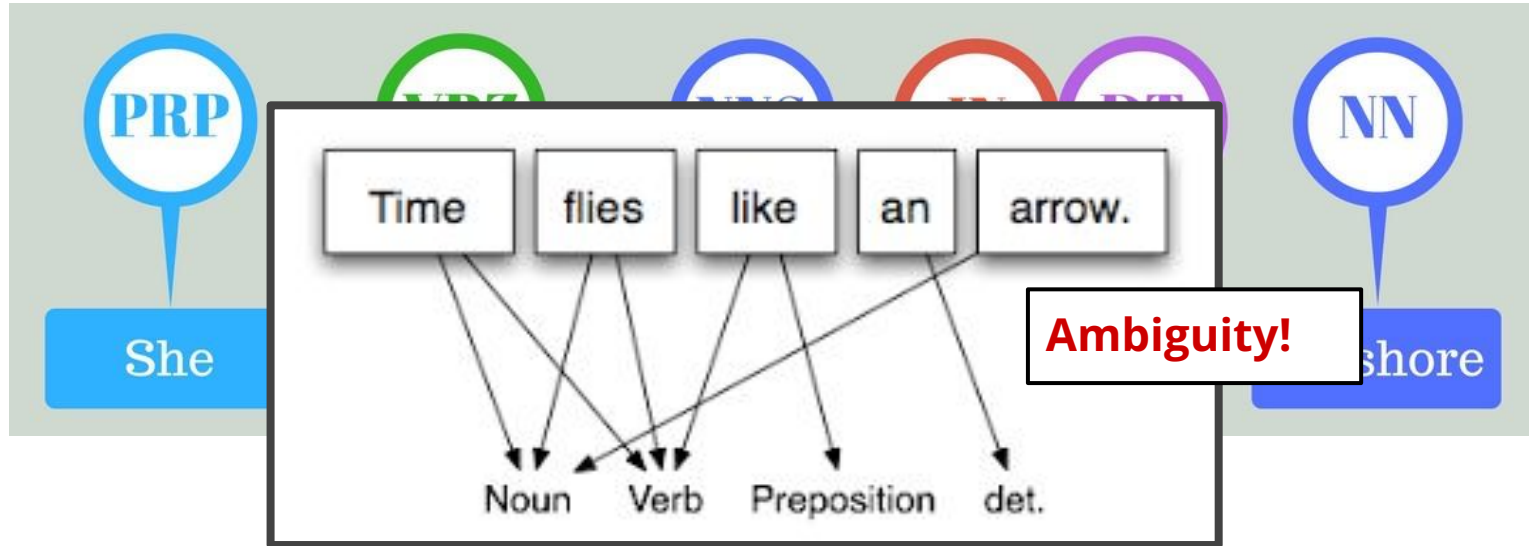
Part-Of-Speech Tagging

= Associating a morpho-syntactic category to each word
→ Noun, Verb, Pronoun, Adjective, Adverb ...



Part-Of-Speech Tagging

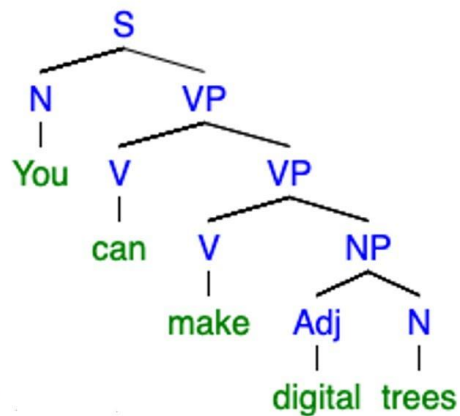
= Associating a morpho-syntactic category to each word
→ Noun, Verb, Pronoun, Adjective, Adverb ...



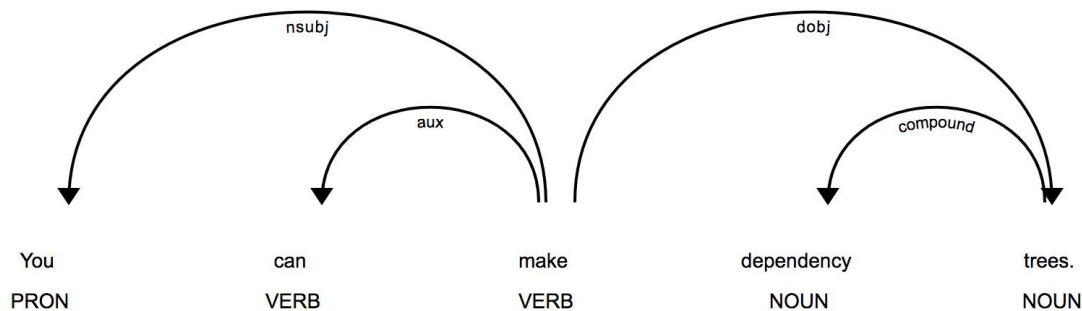
- useful for syntactic parsing, disambiguation, and many applications
- state-of-the-art: English ≈ 97%

Syntactic analysis

Two main paradigms: constituency and dependency



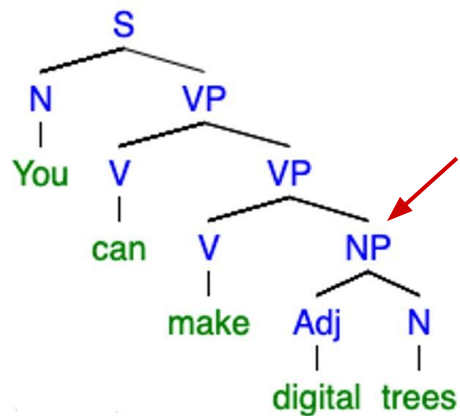
Constituency trees



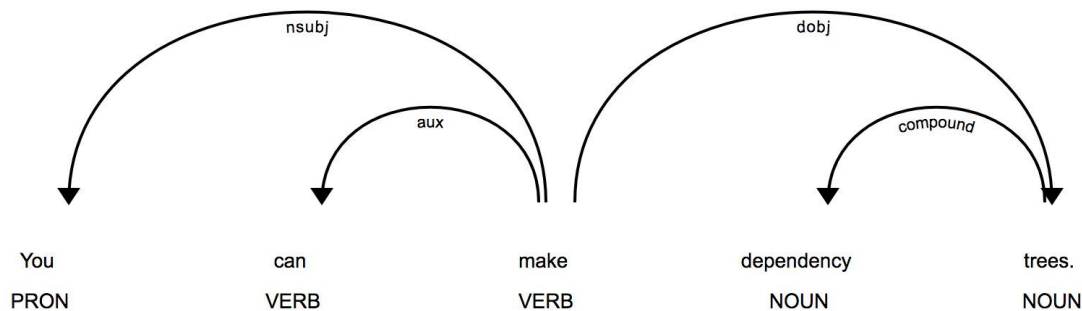
Dependency trees

Syntactic analysis

Two main paradigms: constituency and dependency



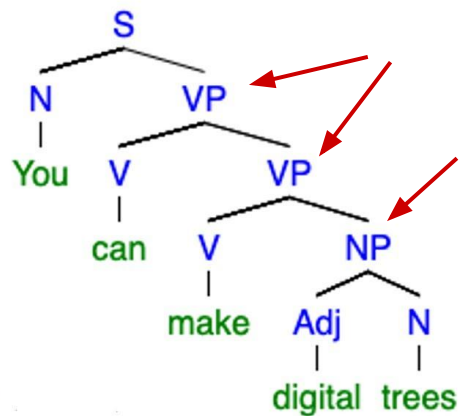
Constituency trees



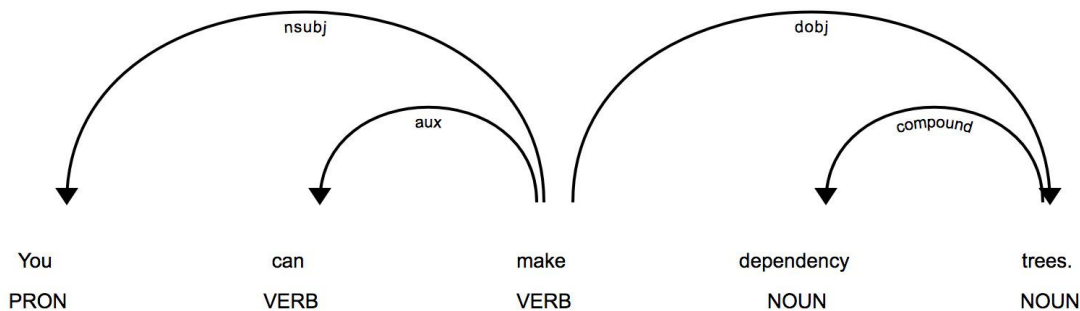
Dependency trees

Syntactic analysis

Two main paradigms: constituency and dependency



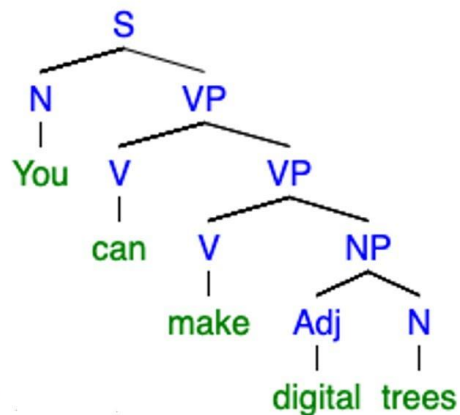
Constituency trees



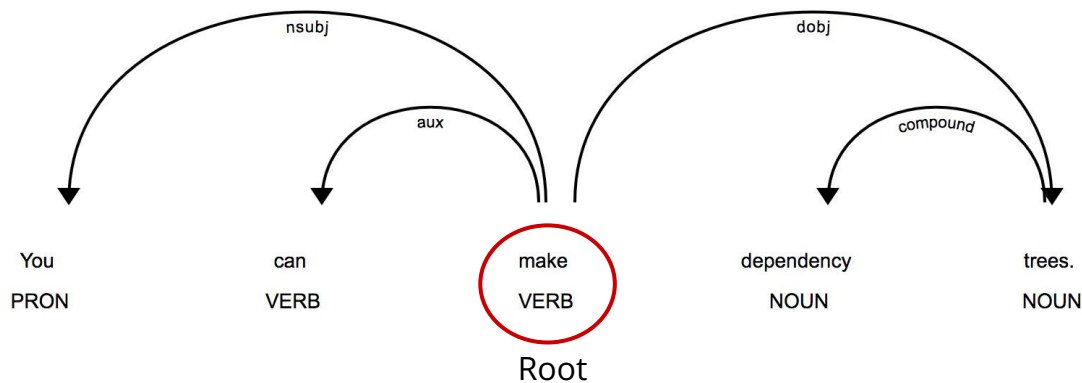
Dependency trees

Syntactic analysis

Two main paradigms: constituency and dependency



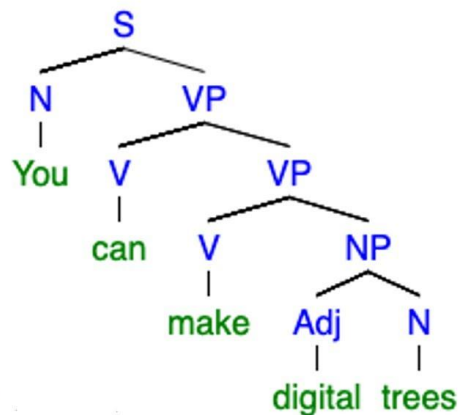
Constituency trees



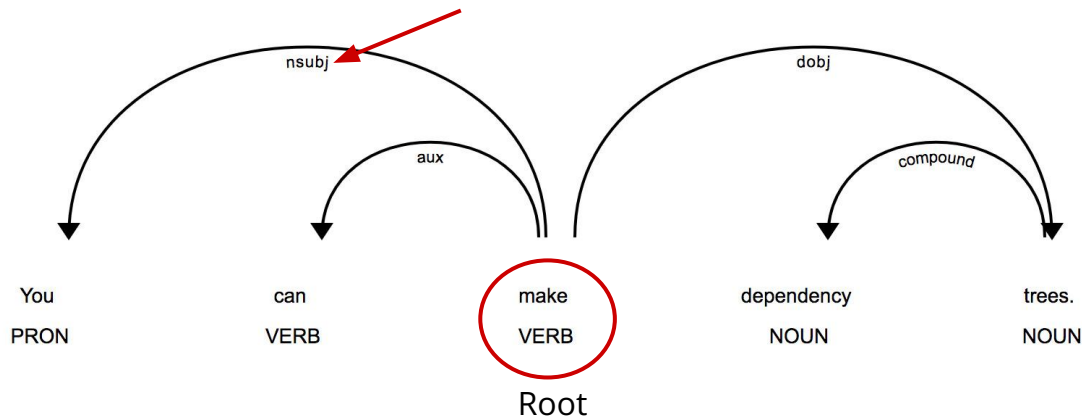
Dependency trees

Syntactic analysis

Two main paradigms: constituency and dependency



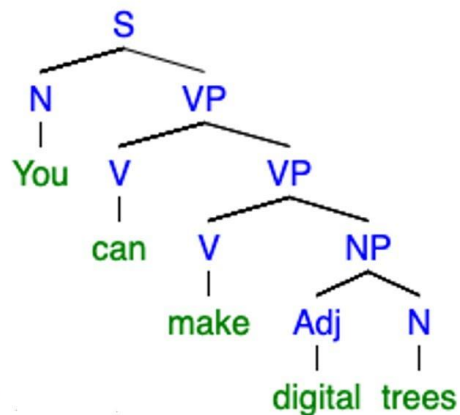
Constituency trees



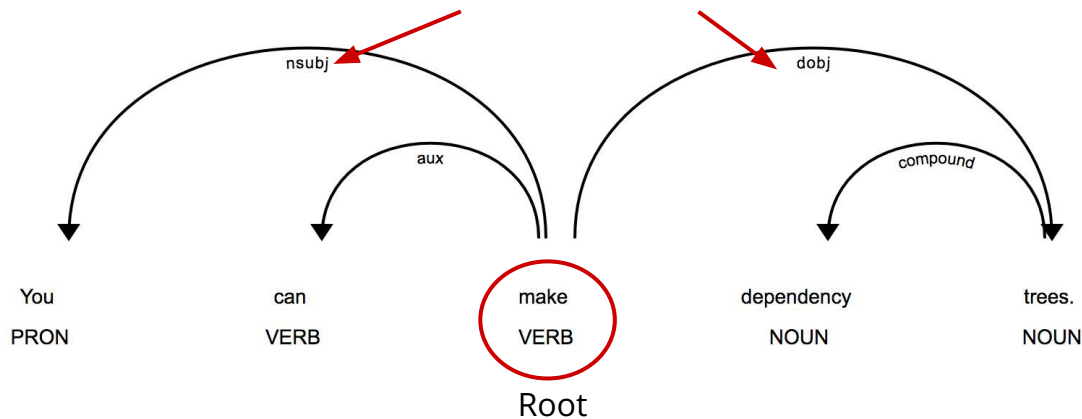
Dependency trees

Syntactic analysis

Two main paradigms: constituency and dependency

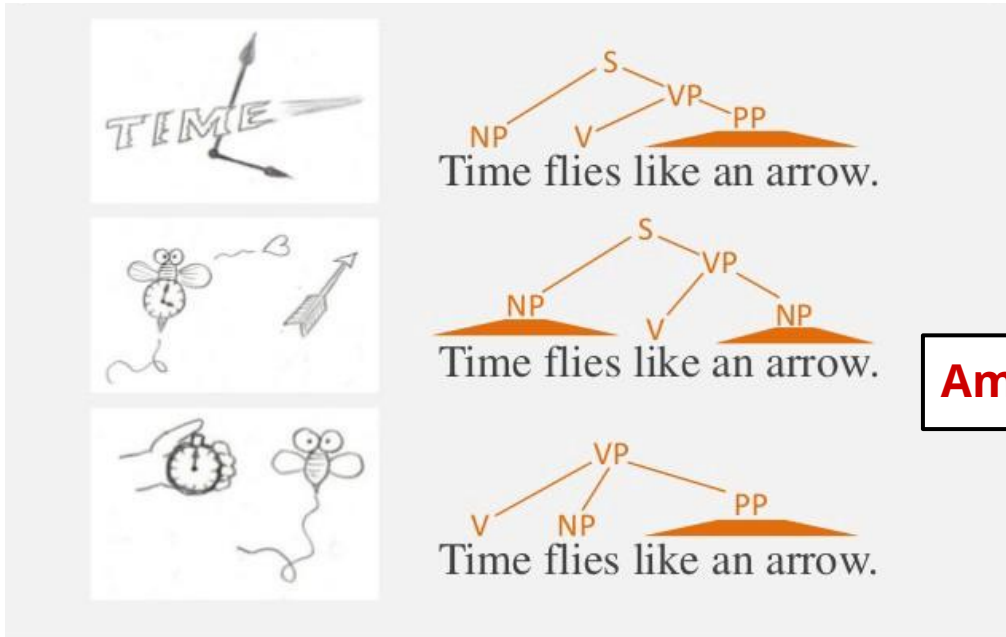


Constituency trees



Dependency trees

Syntactic analysis



Ambiguity!

- Useful for correcting syntactic errors, semantic analysis (e.g. identifying the subject / agent of an action), generation etc
- State-of-the-art system for English Constituency ≈ 96% ; Dependency ≈ 97%

Named Entity Recognition (NER)

= Identifying Named Entities, Typing them

- Types: person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages
- State-of-the-art: English ~ 50-94% (depending on the difficulty of the task)

Elon Musk PERSON apparently wasn't aware that his company SpaceX had a Facebook ORG page. The SpaceX and Tesla PRODUCT CEO has responded to a comment on Twitter GPE calling for him to take down the SpaceX, Tesla and Elon Musk ORG official pages in support of the #deletefacebook movement by first ORDINAL acknowledging he didn't know one existed, and then following up with promises that he would indeed take them down.

He's done just that, as the SpaceX NORP Facebook page is now gone, after having been live earlier today DATE (as you can see from the screenshot included taken at around 12:10 PM ET TIME).

Named Entity Recognition

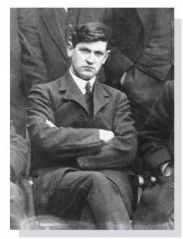
“Paris is the capital of France”



What is the average gas mileage of a Lincoln?



Ambiguity = One name. Multiple entities.



Michael Collins
Mícheál Ó Coileáin



Michael Collins
(1996 movie)



Michael Collins
Astronaut

Michael Collins was born
in Rome and graduated
in the Class of 1952 from
the United States
Military Academy



Knowledgebase

Michael Collins
Astronaut

Collins in 1969

Born	October 31, 1930 <u>Rome</u> , Italy
Died	April 28, 2021 (aged 90) Naples, Florida, US
Resting place	Arlington National Cemetery
Alma mater	<u>United States Military Academy</u> (BS, 1952)
Occupations	Fighter pilot • test pilot • astronaut

Pre-processing tools

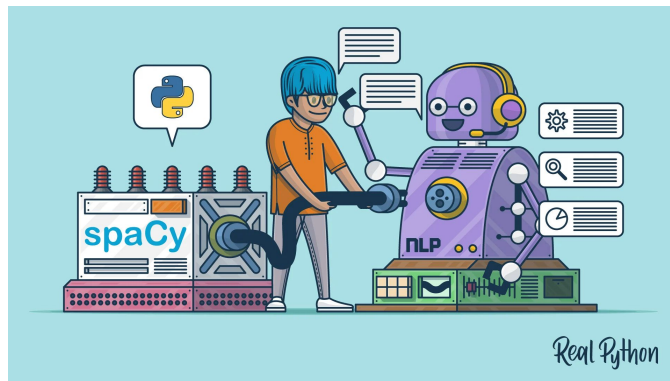
Many existing tools:

- NLTK is a bit outdated, more used for teaching
- Spacy is very easy to use
- Many other tools, often specialized e.g. for tokenization, syntactic parsing ..



Natural Language Analysis
with Python NLTK

<https://www.nltk.org/>



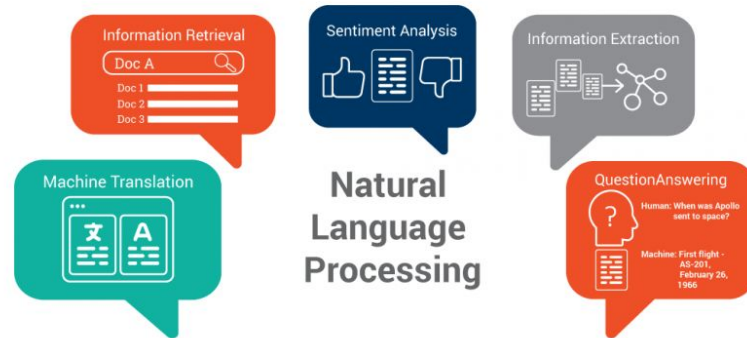
<https://spacy.io/>



nlp-uoregon/
trankit



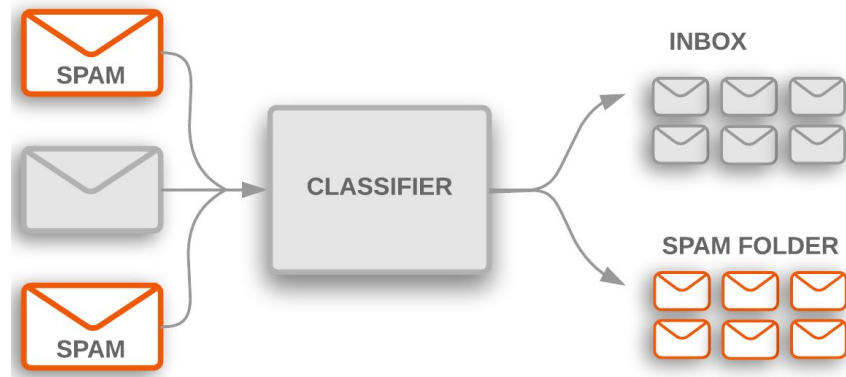
NLP: tasks and applications



Classification

Idea: assign a (known) category to an object

- Word Sense disambiguation
- Classification of books: by genre, author, topic...
- Sentiment analysis: classification of books, tweets, ...
- Language detection
- Fake news detection
- Lie and fraud detection (e.g. in scientific publications...)
- Gender, Political side ... detection
- predicting patients' trajectories, e.g. Identify cancer in clinical notes [\[Rohanian et al. 2023\]](#)
- Classification of clauses in legal documents: categories e.g. Immigration, crime ; unfair under some law etc [\[Guha et al. 2023\]](#)
- Psychological trouble detection
-



Sequence Labelling

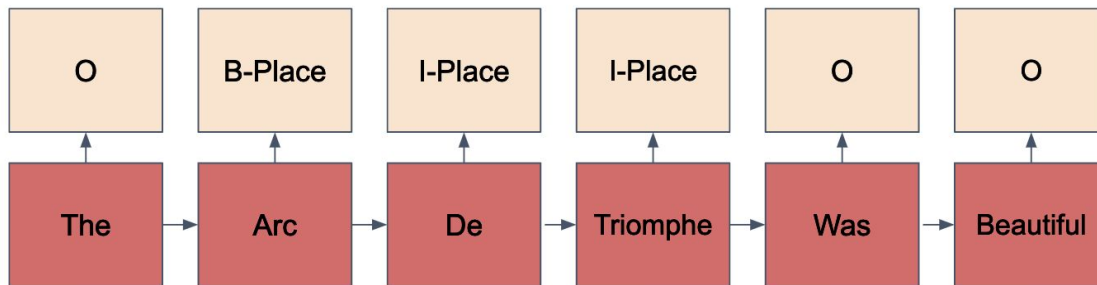
Idea: we want to take into account context, e.g. surrounding words

→ making the optimal label for a given element dependent on the choices of nearby elements

- POS tagging
- Named Entity Recognition

→ Without neurons: = CRF, HMMs ...

→ Now: e.g. with Recurrent Neural Network + CRF



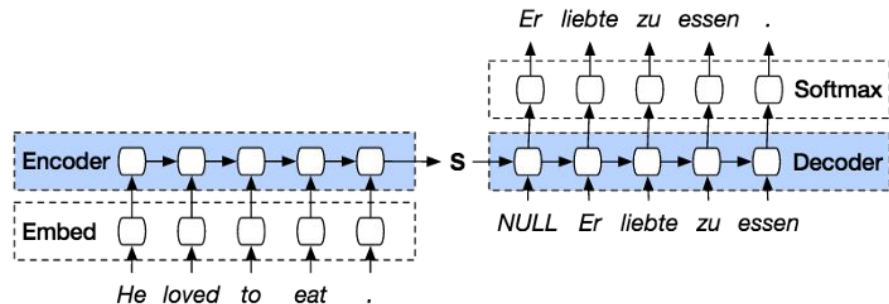
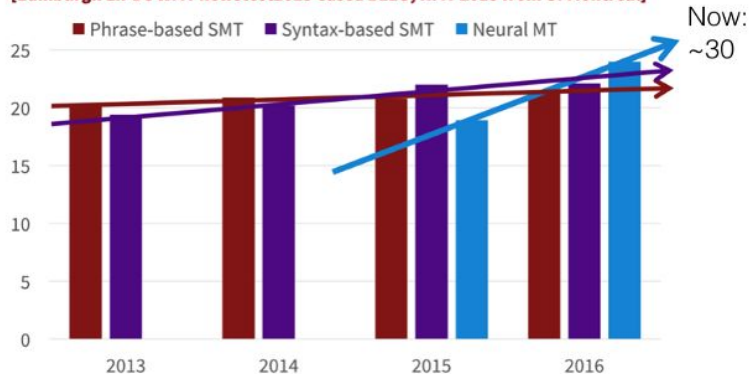
Sequence to Sequence

Idea: we want to take a sequence as input, and to output another sequence, e.g.:

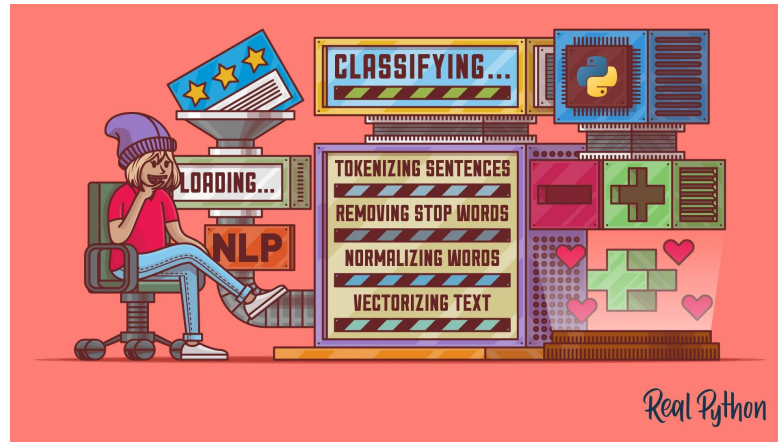
- Machine translation
- Question-answering, e.g. on legal rules [\[Guha et al. 2023\]](#)
- Summarization
- Generation in general:
 - e.g. generating subtitles, captions, poetry...
 - interface for patients to ask questions and access relevant information
 - Report generation, e.g. from radiology reports [\[Kim et al. 2023\]](#)
 - simplification: Patient-friendly clinical notes [\[Trienes et al. 2022\]](#)

Progress in Machine Translation

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]



Learning from textual data



Learning regularities from textual data

NLP systems are based on:

- 1980's Early approach: Symbolic methods = hand-written rules
 - ex: tokenizer based on regular expressions
 - advantages: based on linguistics expertise, very precise
 - inconvenients: lack of coverage, time consuming
- 1990's Statistical approaches: machine learning algorithms
 - learn rules automatically = (mostly) linear functions
 - rather fast to train, still good baselines
- ≈ 2010 Neural methods
 - combine linear and non-linear functions
 - improved performance (in general)
 - harder to interpret ("black-box")

Learning regularities from textual data

NLP systems are based on:

- 1980's Early approach: **Symbolic methods** = handwritten rules
 - ex: tokenizer based on regular expressions
 - advantages: based on linguistics expertise, very precise
 - inconvenients: lack of coverage, time consuming
- 1990's Statistical approaches: machine learning algorithms
 - learn rules automatically = (mostly) linear functions
 - rather fast to train, still good baselines
- ≈ 2010 Neural methods
 - combine linear and non-linear functions
 - improved performance (in general)
 - harder to interpret ("black-box")

Learning regularities from textual data

NLP systems are based on:

- 1980's Early approach: Symbolic methods = handwritten rules
 - ex: tokenizer based on regular expressions
 - advantages: based on linguistics expertise, very precise
 - inconvenients: lack of coverage, time consuming
- 1990's Statistical approaches: **machine learning** algorithms
 - learn rules automatically = (mostly) linear functions
 - rather fast to train, still good baselines
- ≈ 2010 Neural methods
 - combine linear and non-linear functions
 - improved performance (in general)
 - harder to interpret ("black-box")



Learning regularities from textual data

NLP systems are based on:

- 1980's Early approach: Symbolic methods = handwritten rules
 - ex: tokenizer based on regular expressions
 - advantages: based on linguistics expertise, very precise
 - inconvenients: lack of coverage, time consuming
- 1990's Statistical approaches: machine learning algorithms
 - learn rules automatically = (mostly) linear functions
 - rather fast to train, still good baselines
- ≈ 2010 **Neural methods** (still machine learning)
 - combine linear and non-linear functions
 - improved performance (in general)
 - harder to interpret ("black-box")



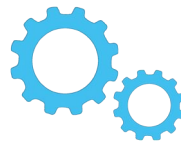
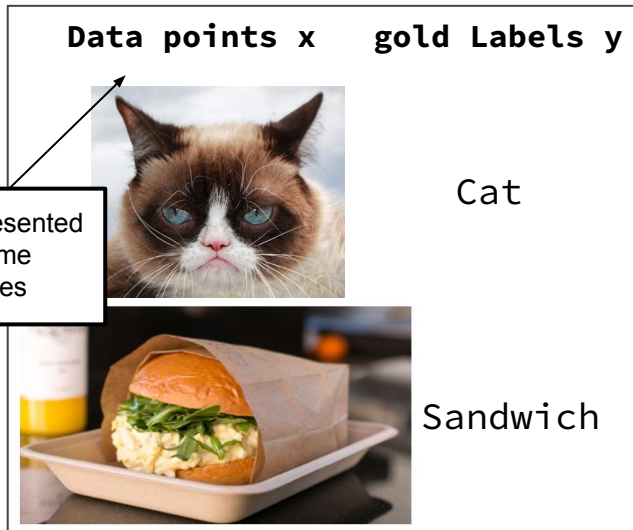
Learning from data: supervised classification

Data preparation

Training

Evaluating

Input Data / Training set
(Labeled examples)



Test set



output

Cat

20/20

Très bien !



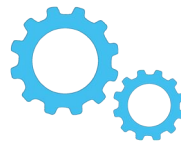
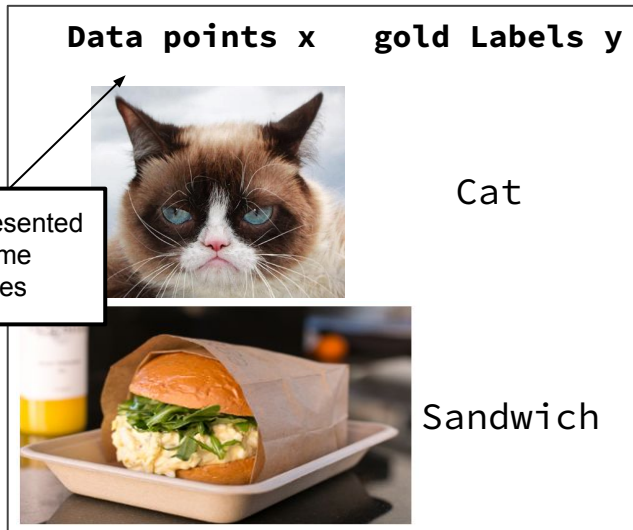
Learning from data: supervised classification

Data preparation

Training

Evaluating

Input Data / Training set
(Labeled examples)



learning some
function f with
parameters

Test set



output

Cat

20/20
Très bien !



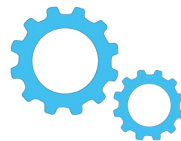
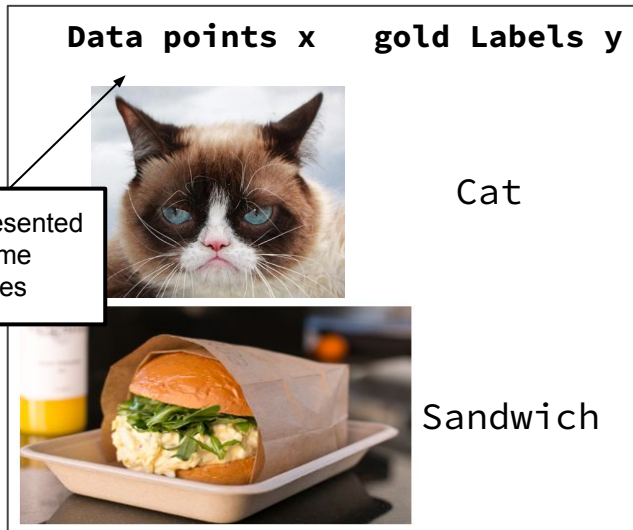
Learning from data: supervised classification

Data preparation

Training

Evaluating

Input Data / Training set
(Labeled examples)



learning some
function f with
parameters

Test set



Use f to
predict a
label

output

Cat

20/20
Très bien !



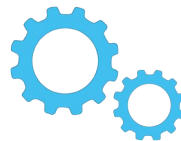
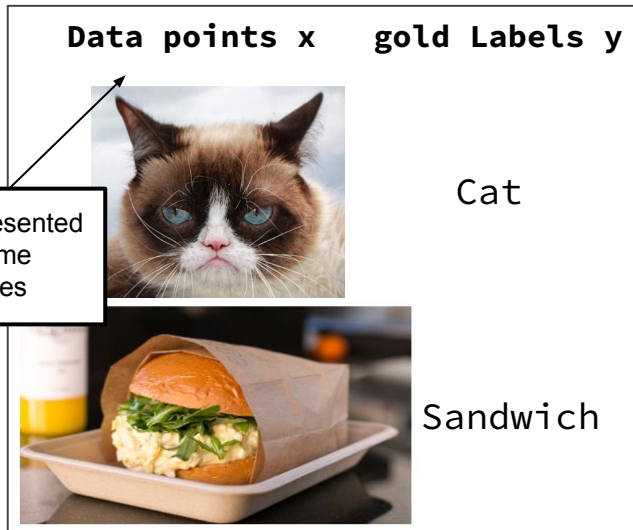
Learning from data: supervised classification

Data preparation

Training

Evaluating

Input Data / Training set
(Labeled examples)



learning some
function f with
parameters

Test set



output

Cat

20/20
Très bien !



Use f to
predict a
label

Compute
score using
some
performance
metrics

Learning from data: supervised classification

Data preparation

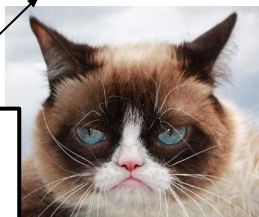
Training

Evaluating

Input Data / Training set
(Labeled examples)

Test set

Data points x gold Labels y

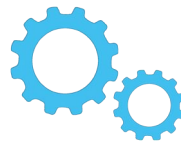


Cat

Represented
by some
features



Sandwich



learning some
function f with
parameters



output

Cat

20/20
Très bien !



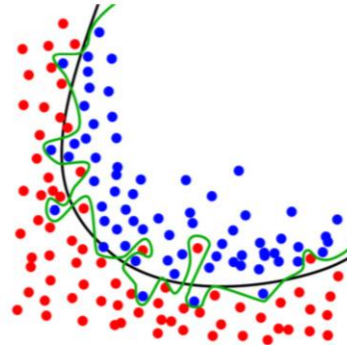
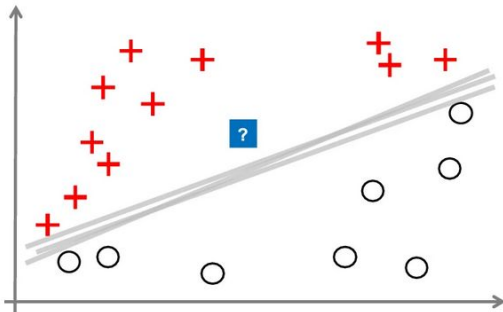
Use f to
predict a
label

Compute
score using
some
performance
metrics

Classical machine learning approach: linear models

→ intuition (2 dimensions): find a line (=linear models) that split the data according to their labels

- many possible lines / hypothesis
- best line: no error (or not too many), good generalization

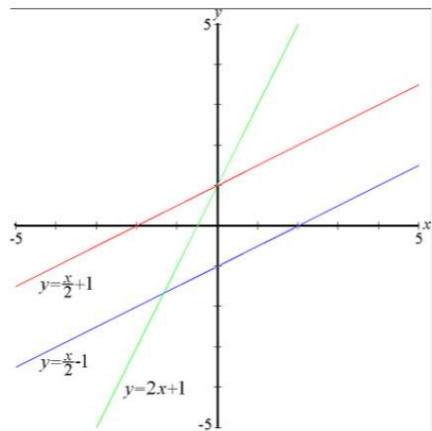


Linear Classifiers

The decision boundary is a **linear function** of the input: in the binary case, it's a line (2 dimensions / features), a plane (3 d) or an hyperplane (n d) separating the two classes

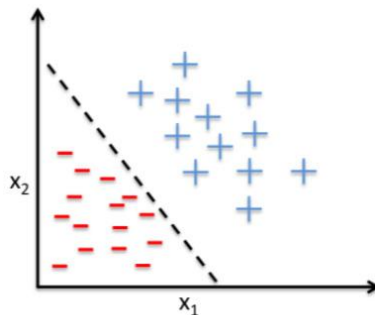
Decision boundary:

$y = w_0 \cdot x_0 + b$: a line

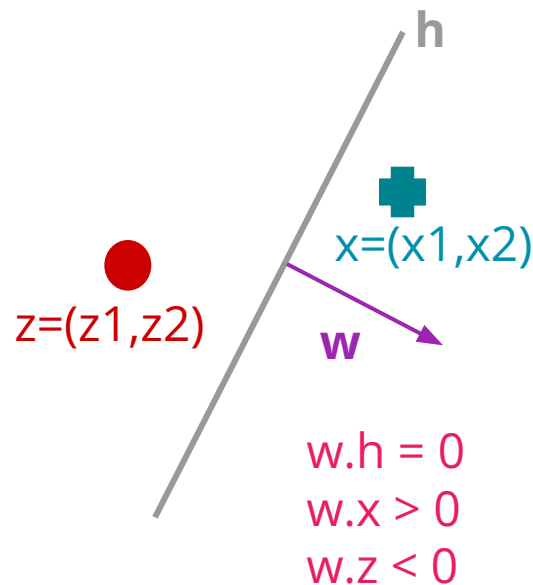


Prediction:

if $\hat{y} = \mathbf{w} \cdot \mathbf{x} > 0$, predict positive class.



Example of a linear decision boundary for binary classification.

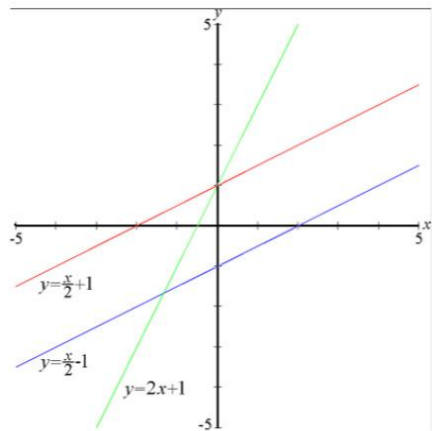


Linear Classifiers

The decision boundary is a **linear function** of the input: in the binary case, it's a line (2 dimensions / features), a plane (3 d) or an hyperplane (n d) separating the two classes

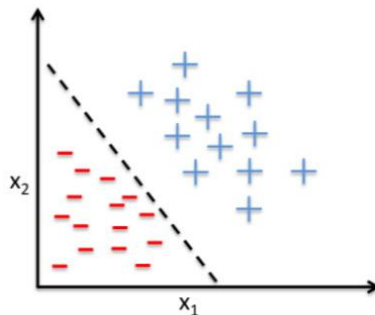
Decision boundary:

$y = w_0 \cdot x_0 + b$: a line

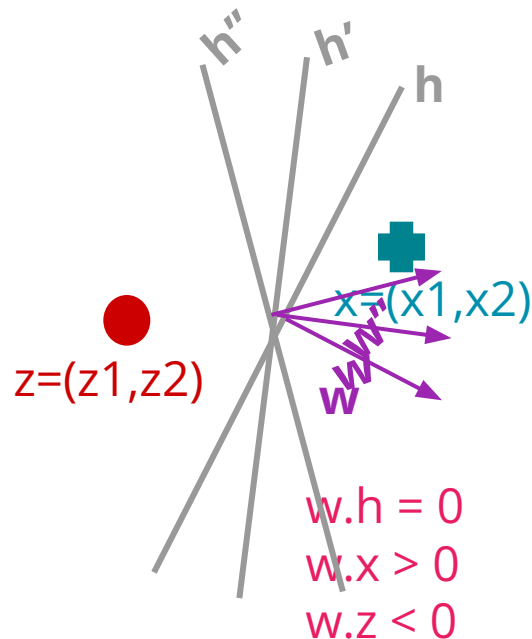


Prediction:

if $\hat{y} = \mathbf{w} \cdot \mathbf{x} > 0$, predict positive class.



Example of a linear decision boundary for binary classification.



Linear Classifiers

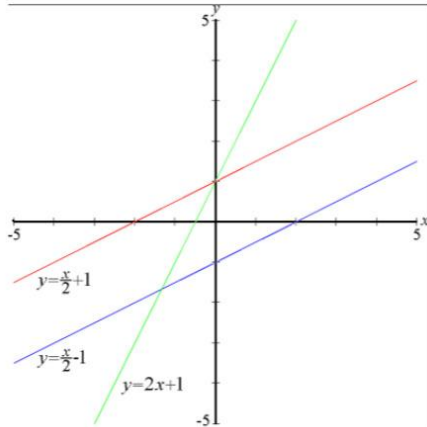
$\mathbf{x} = \langle x_0, x_1, \dots, x_n \rangle$ representing a data point is a vector

→ **how do we build it?**

The decision boundary is a **linear function** of the input: in the binary case, it's a line (2 dimensions / features), a plane (3 d) or an hyperplane (n d) separating the two classes

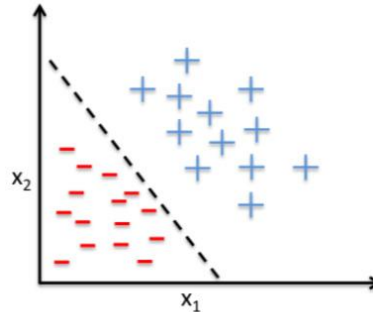
Decision boundary:

$y = w_0 \cdot x_0 + b$: a line

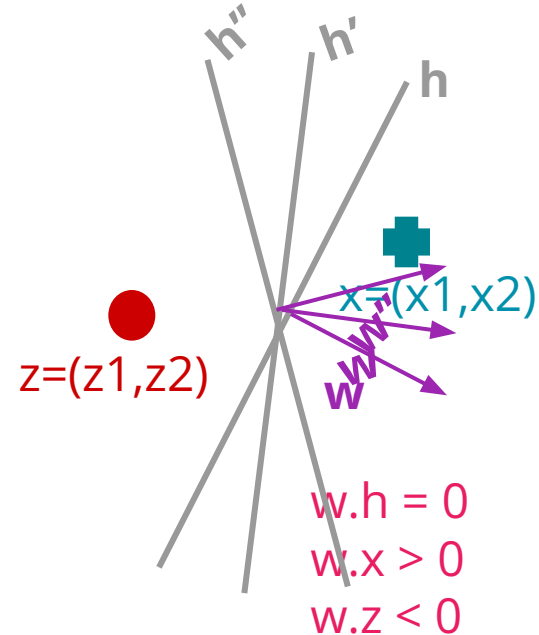


Prediction:

if $\hat{y} = \mathbf{w} \cdot \mathbf{x} > 0$, predict positive class.



Example of a linear decision boundary for binary classification.

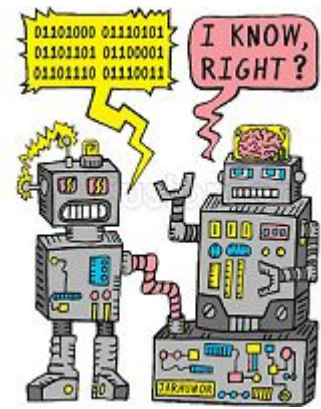


Learning from text data

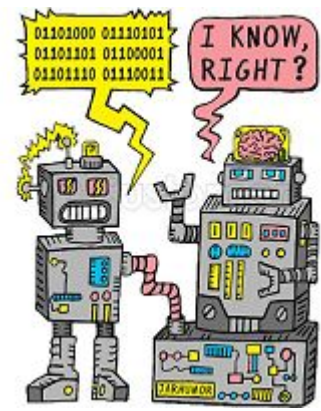
Main issue:

- how to represent text?

e.g. how to transform a sentence into a vector of numerical values?



Learning from text data



Main issue:

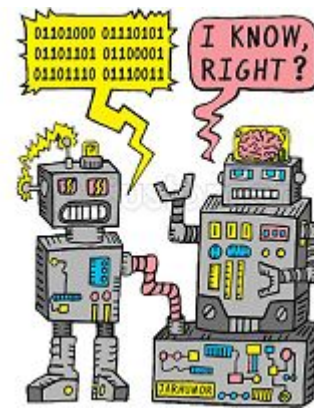
- how to represent text?

e.g. how to transform a sentence into a vector of numerical values?

Bag-of-Words (BOW):

- one vector where each dimension is a word / feature in our vocabulary
- if the word / feature is present in the document, associate a specific value

Learning from text data



Main issue:

- how to represent text?

e.g. how to transform a sentence into a vector of numerical values?

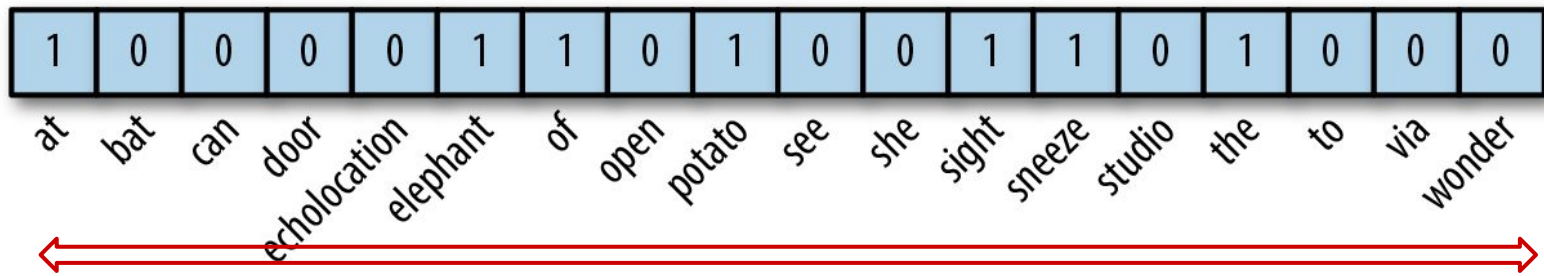
Bag-of-Words (BOW):

- one vector where **each dimension is a word / feature in our vocabulary**
- if the word / feature is present in the document, associate a specific value

BOW: One-hot encoding

The elephant sneezed
at the sight of potatoes.

- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1

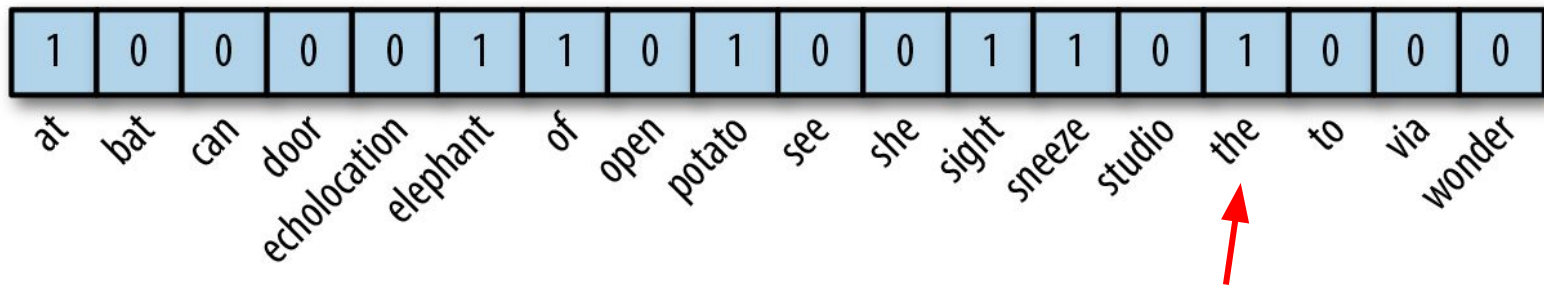


18 words / dimensions

BOW: One-hot encoding

The elephant sneezed
at the sight of potatoes.

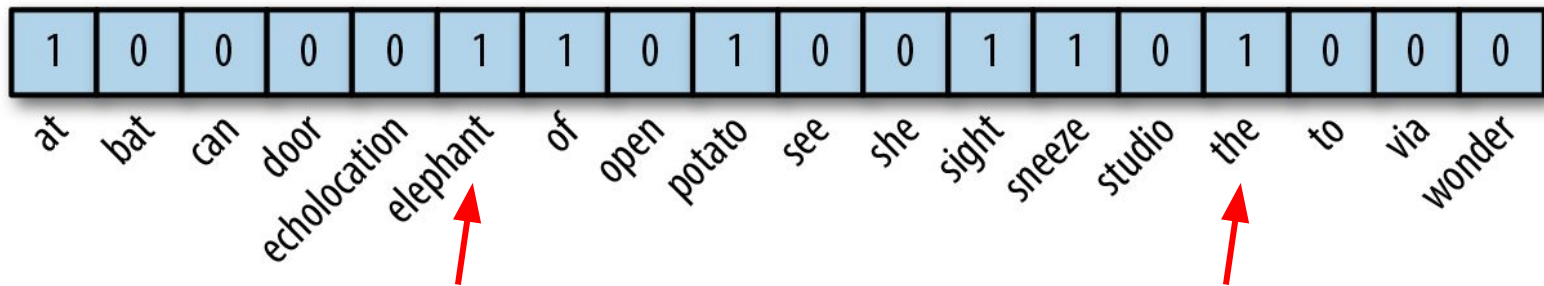
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



BOW: One-hot encoding

The elephant sneezed
at the sight of potatoes.

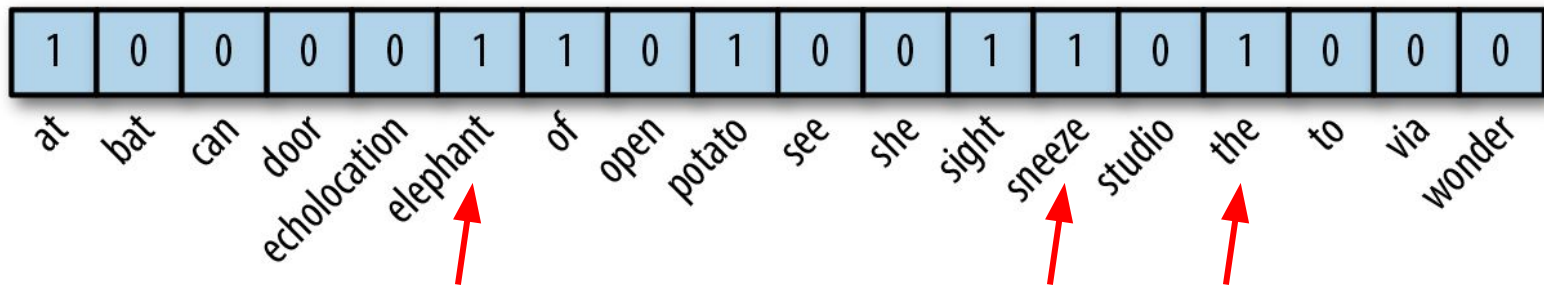
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



BOW: One-hot encoding

The elephant sneezed
at the sight of potatoes.

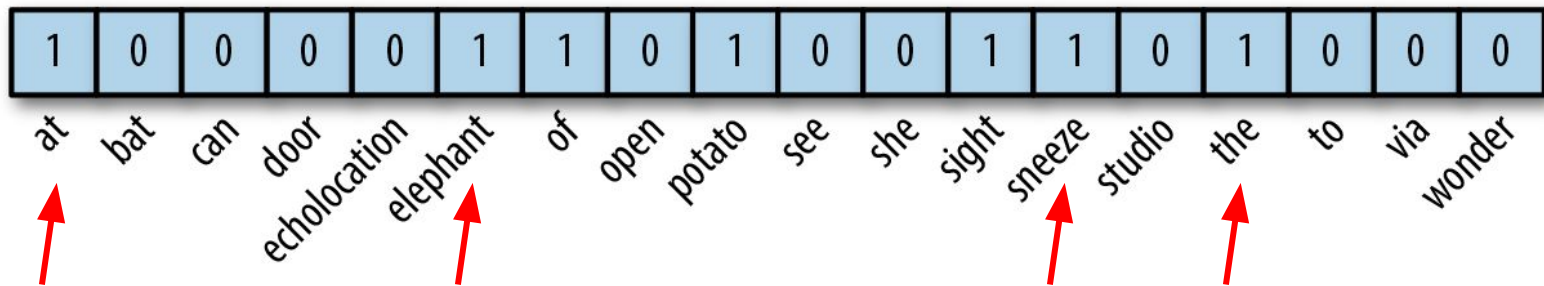
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



BOW: One-hot encoding

The elephant sneezed
at the sight of potatoes.

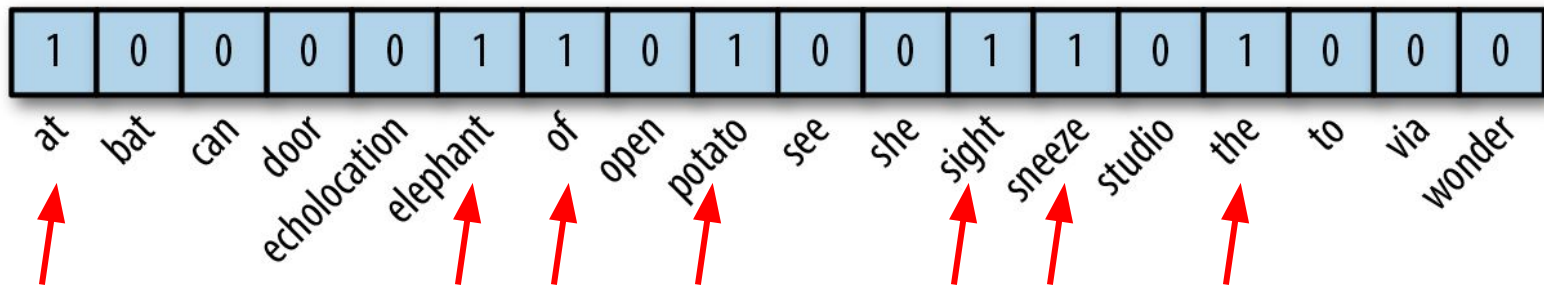
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



BOW: One-hot encoding

The elephant sneezed
at the sight of potatoes.

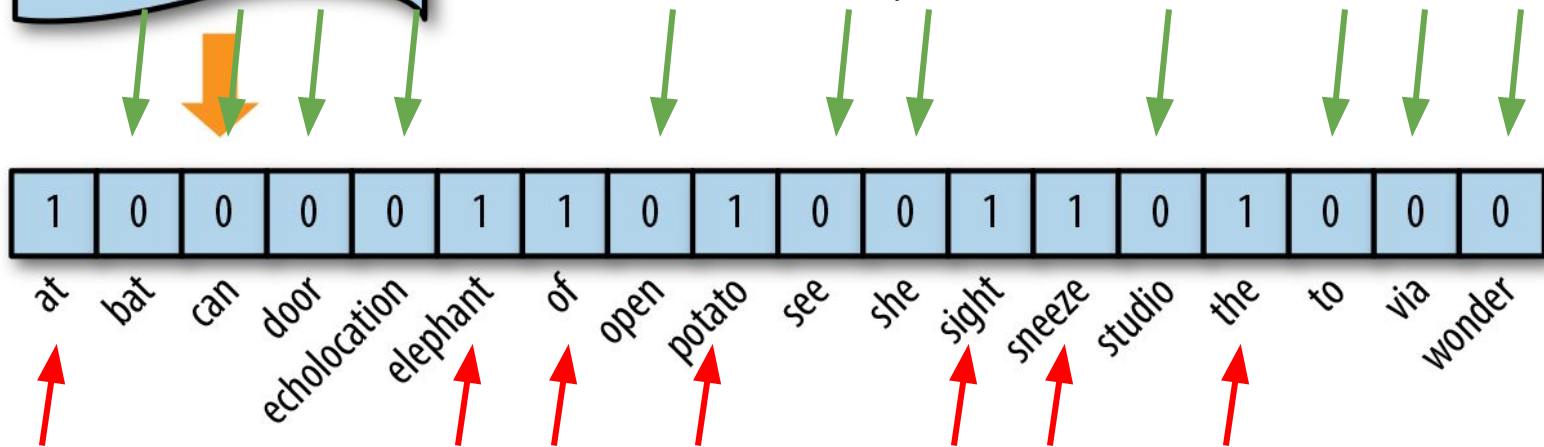
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



BOW: One-hot encoding


The elephant sneezed
at the sight of potatoes.

- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1
- The other words: value = 0 → present in the training data, but not in this specific sentence / document



Bag-of-Words

- Easy to use: now the computer can “read” your sentence


 **The elephant sneezed at the sight of potatoes.**
<1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0>

Varied flavors:

- **Binary**
- Raw frequencies: some words are repeated = more important
- Normalizing with TF-IDF: take into account the distribution of the words in the entire corpus
 - “the”: very frequent but not very crucial
 - “magnificent”: rare, but crucial

Bag-of-Words

- Easy to use: now the computer can “read” your sentence

 **The elephant sneezed at the sight of potatoes.**
<1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 2, 0, 0, 0>

Varied flavors:

‘the’

- Binary
- **Raw frequencies**: some words are repeated = more important
- Normalizing with TF-IDF: take into account the distribution of the words in the entire corpus
 - “the”: very frequent but not very crucial
 - “magnificent”: rare, but crucial

Bag-of-Words

- Easy to use: now the computer can “read” your sentence



The elephant sneezed at the sight of potatoes.

<1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 2, 0, 0, 0>

Varied flavors:

- Binary
- Raw frequencies: some words are repeated = more important
- **Normalizing with TF-IDF**: take into account the distribution of the words in the entire corpus
 - “the”: very frequent but not very crucial
 - “magnificent”: rare, but crucial

$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF

Term x within document y

$\text{tf}_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Bag of any features

Can be used to take into account any information, e.g. POS tags:

The/D elephant/N sneezed/V at/P the/D sight/N of/P potatoes/N

1	0	0	0	0	1	1	0	1	0	0	1	1	0	1	0	0	0	1	1	1	1	0
at	bat	can	door	echolocation	elephant	of	open	potato	see	she	sight	sneeze	studio	the	to	via	wonder	D	N	V	P	A

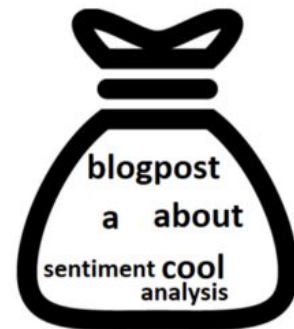
We can encode any information:

- presence of a syntactic relation
- presence of a Named Entity
- word associated to a sense if disambiguated
- words in the next sentence
- ...

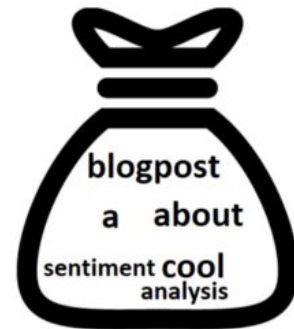
Problems and extensions

1. **Very high dimensional:**

- 18 dimensions for the previous sentence but could be 100k dimensions!
 - Curse of dimensionality: makes learning hard, prone to overfitting
- Solutions:
 - ignoring specific words, e.g. stop words
 - keeping only the most frequent / highest TF-IDF
 - grouping words: semantic categories, clusters (Brown)



Problems and extensions



1. Very high dimensional:

- 18 dimensions for the previous sentence but could be 100k dimensions!
 - Curse of dimensionality: makes learning hard, prone to overfitting
- Solutions:
 - ignoring specific words, e.g. stop words
 - keeping only the most frequent / highest TF-IDF
 - grouping words: semantic categories, clusters (Brown)

2. Bag-of-Words representation **ignores word ordering and context**

- crucial:
 - *"I don't know why **I like this movie.**" vs "**I don't like this movie** and I know why."*
- solutions: n-grams, i.e. use combination of multiple words
 - e.g. trigrams such as "do not like", "like this movie"
 - but even more dimensions!

Learning algorithms for classification

- Naive Bayes
- **Linear classifiers:**
 - perceptron
 - passive-aggressive
 - Logistic Regression aka MaxEnt
 - linear SVM
- Non linear SVM
- Neural networks

See [the doc on supervised learning](#)

See [the tutorial: working with text](#)

Example: Logistic Regression

- Spam (binary) classification
- We take word frequency as features

$$\text{score}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

\mathbf{x}_1 : “**Pharmacie** en ligne: **viagra** **meilleure** **offre!**”

$$\begin{aligned} \text{score}(\mathbf{x}_1) &= 0.4 \times 1 + 1.2 \times 1 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 0 + (-1.7) \\ &\times 0 + 0.1 = 2.1 \end{aligned}$$

feature f_i	weight w_i
pharmacie	0.4
viagra	1.2
meilleure	0.2
offre	0.2
demande	-0.8
transmets	-1.7
bias	0.1

Example: Logistic Regression

- Spam (binary) classification
- We take word frequency as features

$$\text{score}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

\mathbf{x}_1 : “**Pharmacie** en ligne: **viagra** **meilleure** **offre!**”

$$\text{score}(\mathbf{x}_1) = 0.4 \times 1 + 1.2 \times 1 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 0 + (-1.7) \times 0 + 0.1 = 2.1$$

feature f_i	weight w_i
pharmacie	0.4
viagra	1.2
meilleure	0.2
offre	0.2
demande	-0.8
transmets	-1.7
bias	0.1

Example: Logistic Regression

- Spam (binary) classification
- We take word frequency as features

$$\text{score}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

\mathbf{x}_1 : “**Pharmacie** en ligne: **viagra** **meilleure** **offre!**”

$$\text{score}(\mathbf{x}_1) = 0.4 \times 1 + 1.2 \times 1 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 0 + (-1.7) \times 0 + 0.1 = 2.1$$

feature f_i	weight w_i
pharmacie	0.4
viagra	1.2
meilleure	0.2
offre	0.2
demande	-0.8
transmets	-1.7
bias	0.1

Example: Logistic Regression

- Spam (binary) classification
- We take word frequency as features

$$\text{score}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

\mathbf{x}_1 : “**Pharmacie** en ligne: **viagra** **meilleure** **offre!**”

$$\text{score}(\mathbf{x}_1) = 0.4 \times 1 + 1.2 \times 1 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 0 + (-1.7) \times 0 + 0.1 = 2.1$$

\mathbf{x}_2 : “Suite à votre **demande**, je vous **transmets** notre **meilleure** **offre**”

$$\text{score}(\mathbf{x}_2) = 0.4 \times 0 + 1.2 \times 0 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 1 + (-1.7) \times 1 + 0.1 = -2.0$$

feature f_i	weight w_i
pharmacie	0.4
viagra	1.2
meilleure	0.2
offre	0.2
demande	-0.8
transmets	-1.7
bias	0.1

Example: Logistic Regression

- Spam (binary) classification
- We take word frequency as features

linear function

$$\text{score}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

\mathbf{x}_1 : “**Pharmacie** en ligne: **viagra** **meilleure** **offre!**”

$$\text{score}(\mathbf{x}_1) = 0.4 \times 1 + 1.2 \times 1 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 0 + (-1.7) \times 0 + 0.1 = 2.1$$

\mathbf{x}_2 : “Suite à votre demande, je vous transmets notre meilleure offre”

$$\text{score}(\mathbf{x}_2) = 0.4 \times 0 + 1.2 \times 0 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 1 + (-1.7) \times 1 + 0.1 = -2.0$$

feature f_i	weight w_i
pharmacie	0.4
viagra	1.2
meilleure	0.2
offre	0.2
demande	-0.8
transmets	-1.7
bias	0.1

Example: Logistic Regression

- Spam (binary) classification
- We take word frequency as features

linear function

$$\text{score}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

\mathbf{x}_1 : “Pharmacie en ligne: viagra meilleure offre!”

$$\text{score}(\mathbf{x}_1) = 0.4 \times 1 + 1.2 \times 1 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 0 + (-1.7) \times 0 + 0.1 = 2.1$$

\mathbf{x}_2 : “Suite à votre demande, je vous transmets notre meilleure offre”

$$\text{score}(\mathbf{x}_2) = 0.4 \times 0 + 1.2 \times 0 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 1 + (-1.7) \times 1 + 0.1 = -2.0$$

feature f_i	weight w_i
pharmacie	0.4
viagra	1.2
meilleure	0.2
offre	0.2
demande	-0.8
transmets	-1.7
bias	0.1

scores

Example: Logistic Regression

- Spam (binary) classification
- We take word frequency as features

$$\text{score}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

linear function

\mathbf{x}_1 : “**Pharmacie** en ligne: **viagra** **meilleure** **offre!**”

$$\text{score}(\mathbf{x}_1) = 0.4 \times 1 + 1.2 \times 1 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 0 + (-1.7) \times 0 + 0.1 = 2.1$$

\mathbf{x}_2 : “Suite à votre demande, je vous transmets notre meilleure offre”

$$\text{score}(\mathbf{x}_2) = 0.4 \times 0 + 1.2 \times 0 + 0.2 \times 1 + 0.2 \times 1 + (-0.8) \times 1 + (-1.7) \times 1 + 0.1 = -2.0$$

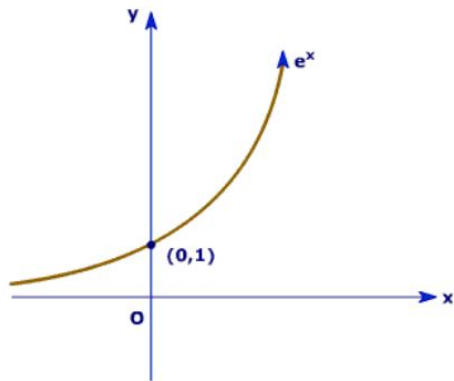
These weights are learned during training

feature f_i	weight w_i
pharmacie	0.4
viagra	1.2
meilleure	0.2
offre	0.2
demande	-0.8
transmets	-1.7
bias	0.1

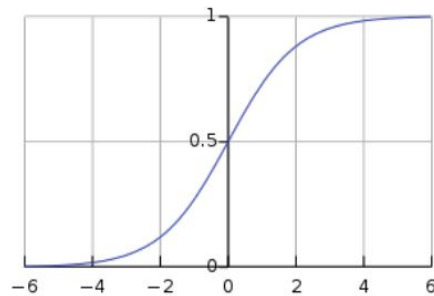
scores

Example with Logistic Regression

- Linear scores range $[-\infty, \infty]$ difficult to interpret, we prefer probabilities
- Linear scores are transformed using (non linear) logistic functions



exponential function (e^x)



logistic function ($\frac{1}{1+e^{-x}}$)

- $f(\text{score}(x_1)) = \frac{1}{1+e^{-2.1}} = .89$
- $f(\text{score}(x_2)) = \frac{1}{1+e^{2.0}} = .12$

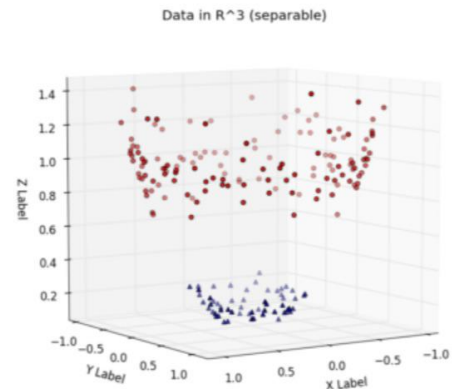
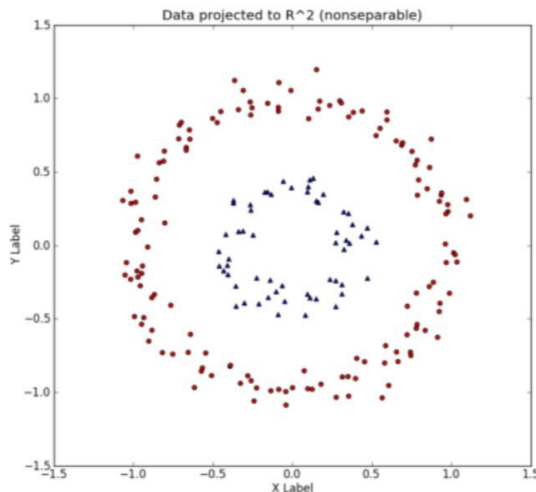
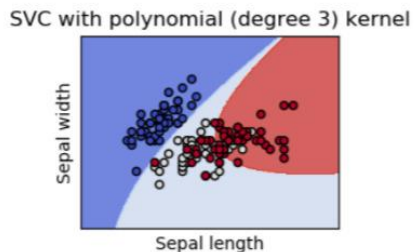
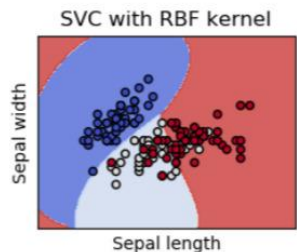
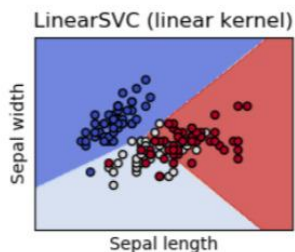
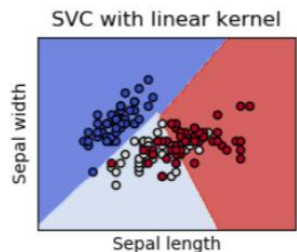
x_1 : “Pharmacie en ligne: **viagra meilleure offre!**”

x_2 : “Suite à votre demande, je vous transmets notre meilleure offre”

Introducing non-linearity

SVM with non-linear kernel

- mapping of the original input feature space to a **higher-dimensional** feature space,
- with the hope that data may be **linearly separable** in this new space

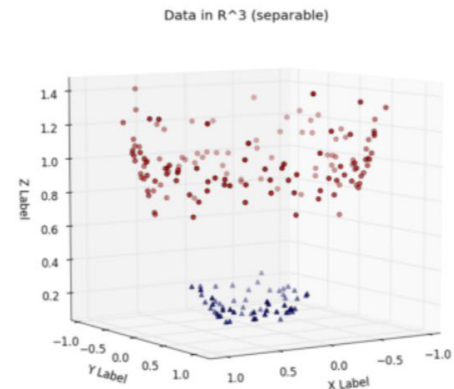
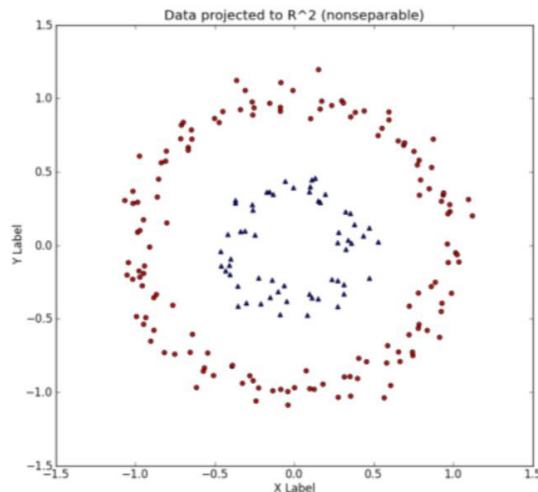
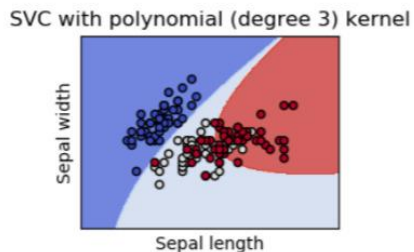
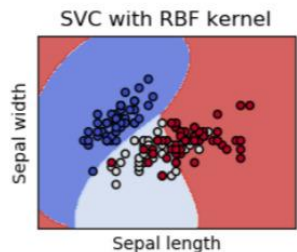
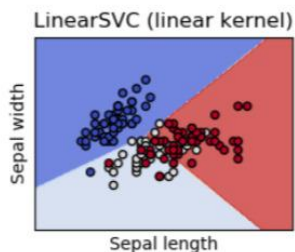
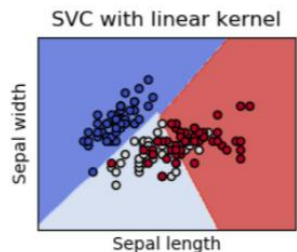


Introducing non-linearity

Neural Network: keep non-linearity and transformation of the input space.

SVM with non-linear kernel

- mapping of the original input feature space to a **higher-dimensional** feature space,
- with the hope that data may be **linearly separable** in this new space



Playing with data (without neurons): scikit-learn

Toolkit for machine learning in Python (developed at INRIA):

- Contains the implementation of many algorithms:
 - Naive Bayes
 - Logistic Regression
 - SVM
 - Decision Trees
 - Perceptron
 - Passive-agressive
 - ...
- Makes also easy to:
 - pre-process / vectorize text data
 - optimize and evaluate models
 - select features
- Classification, clustering, regression



Neural architectures



Limits of classical statistical methods

- Data representation: problems of sparsity and word similarity
- Learning architectures: Non-linearity for non linearly separable data

Standard / classical approach:

- **linear model** trained over high-dimensional but **very sparse** feature vectors

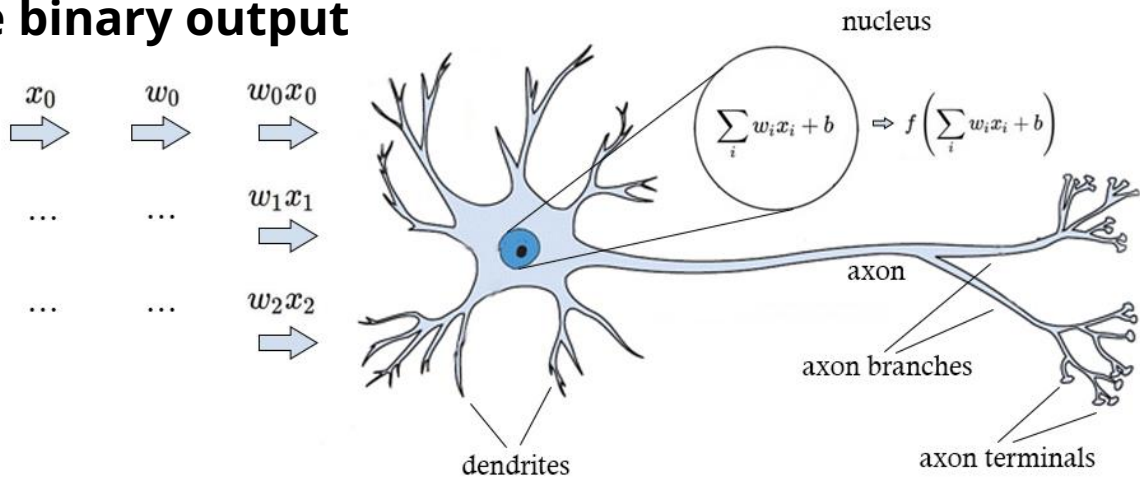
Neural approach:

- **non-linear** neural networks over **dense input vectors**

Neural architecture: the brain-inspired metaphor

Brain computation units = neurons [Perceptron, Rosenblatt, 1957]:

- A neuron has scalar inputs and outputs
- Each input has an associated **weight** to control its importance: the neuron multiplies each input by its weight and then sums = **linear combination**
- If the weighted sum is greater than the activation potential → the neuron “fires” = produces a **single binary output**
- The neurons are connected and form a **network** → the output of a neuron may feed into the inputs of one or more neurons

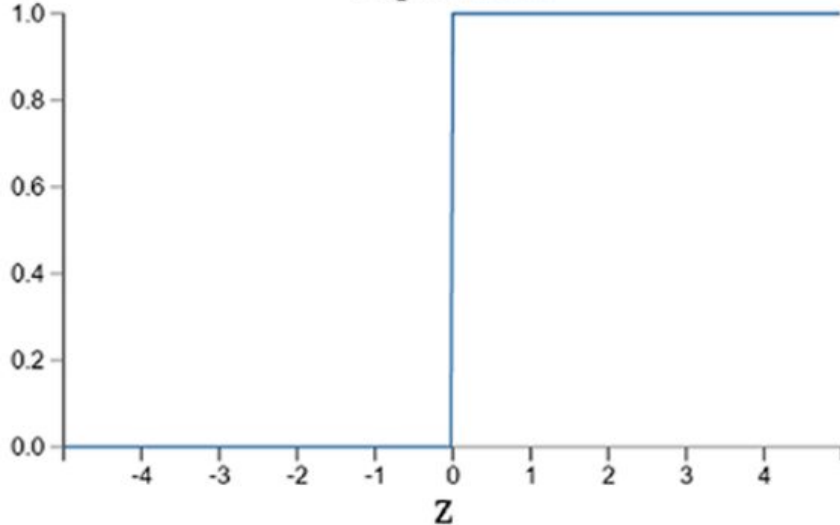


Artificial neuron

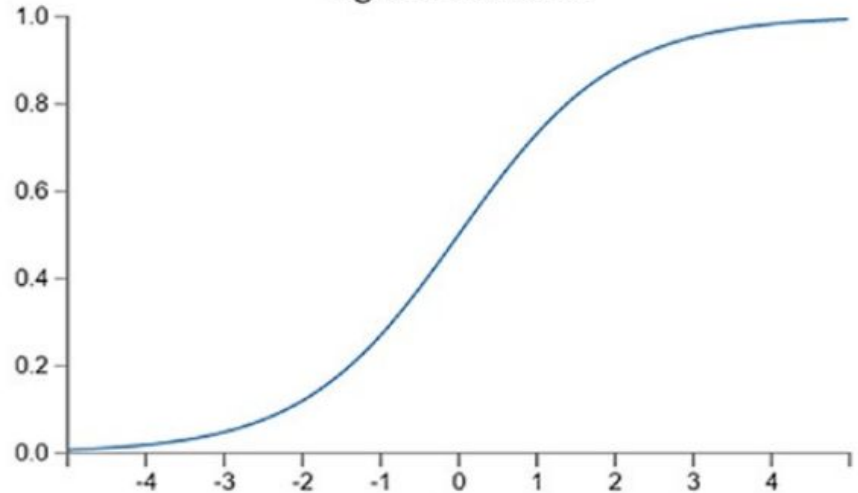
Using a binary output: not very practical

→ We prefer having small change in weight leading to small change in output

step function



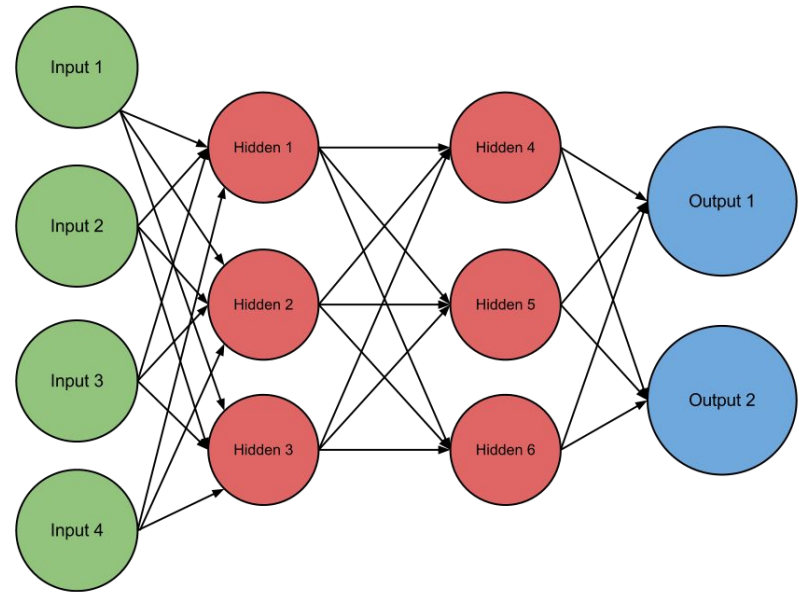
sigmoid function



Feed-forward Neural Network

Also called Multi-Layered Perceptron (MLP)

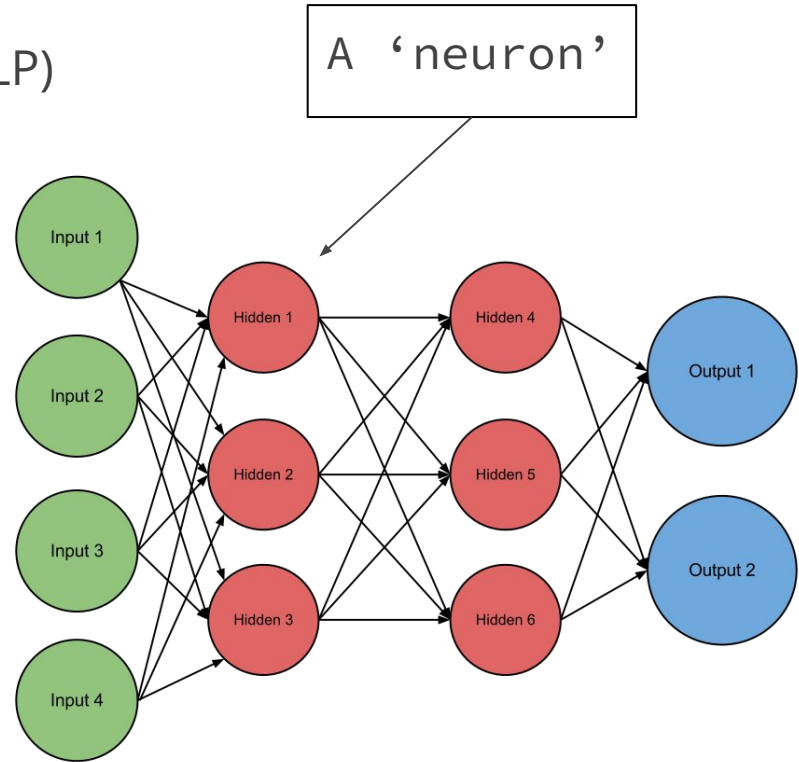
- standard neural network approach
- Fully connected layers
- Can be used as drop-in replacement for typical classifiers with one-hot inputs (but in general used with dense inputs)



Feed-forward Neural Network

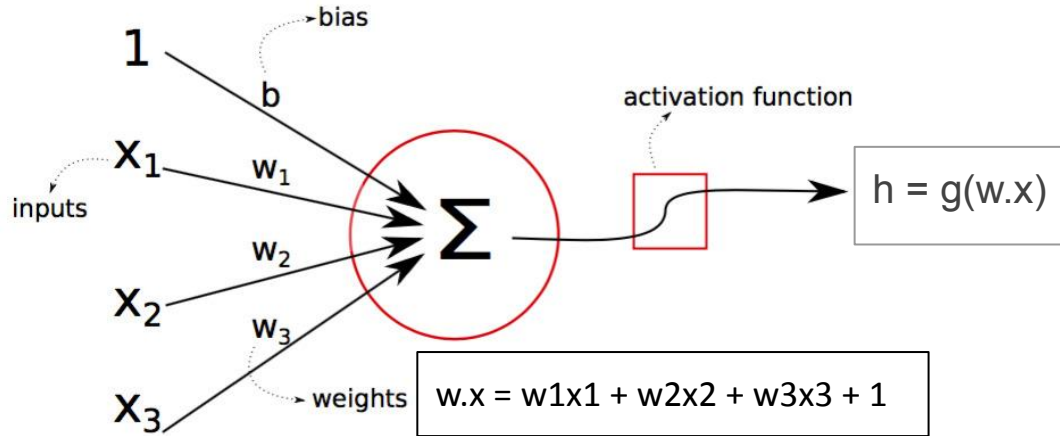
Also called Multi-Layered Perceptron (MLP)

- standard neural network approach
- Fully connected layers
- Can be used as drop-in replacement for typical classifiers with one-hot inputs (but in general used with dense inputs)



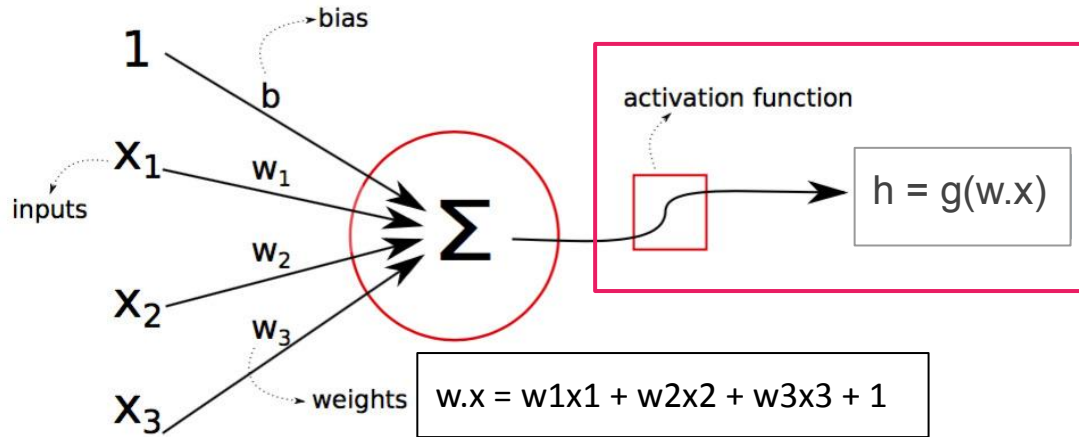
A 'neuron' in neural networks

- Layers are made of neurons = building blocks of neural networks
- A single neuron works like logistic regression, i.e. linear combination + a (non linear) function
 - We can feed an input vector to a bunch of LR functions and get an output vector
 - which can be fed to another layer of LR functions

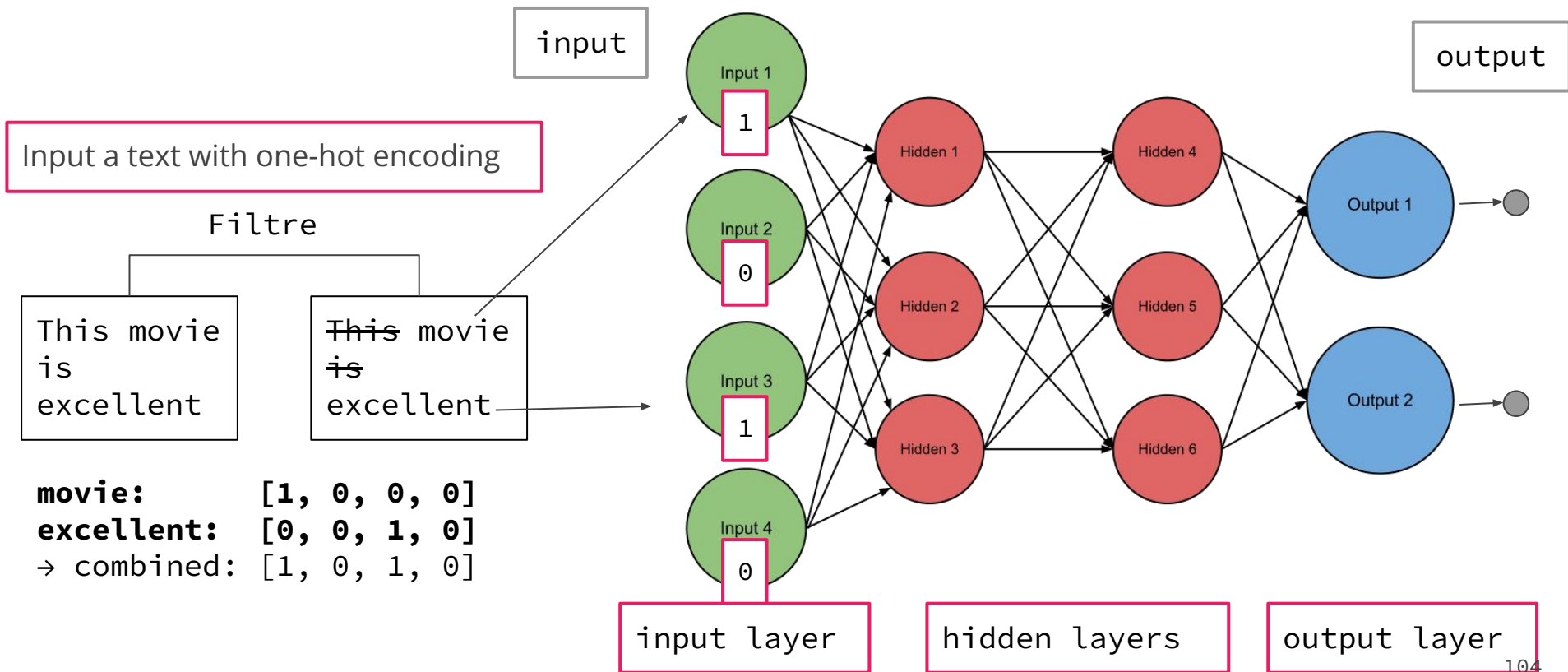


A 'neuron' in neural networks

- Layers are made of neurons = building blocks of neural networks
- A single neuron works like logistic regression, i.e. linear combination + a (non linear) function
 - We can feed an input vector to a bunch of LR functions and get an output vector
 - which can be fed to another layer of LR functions



Example with text classification



Example with text classification

Do some calculations through the network

Input a text with one-hot encoding

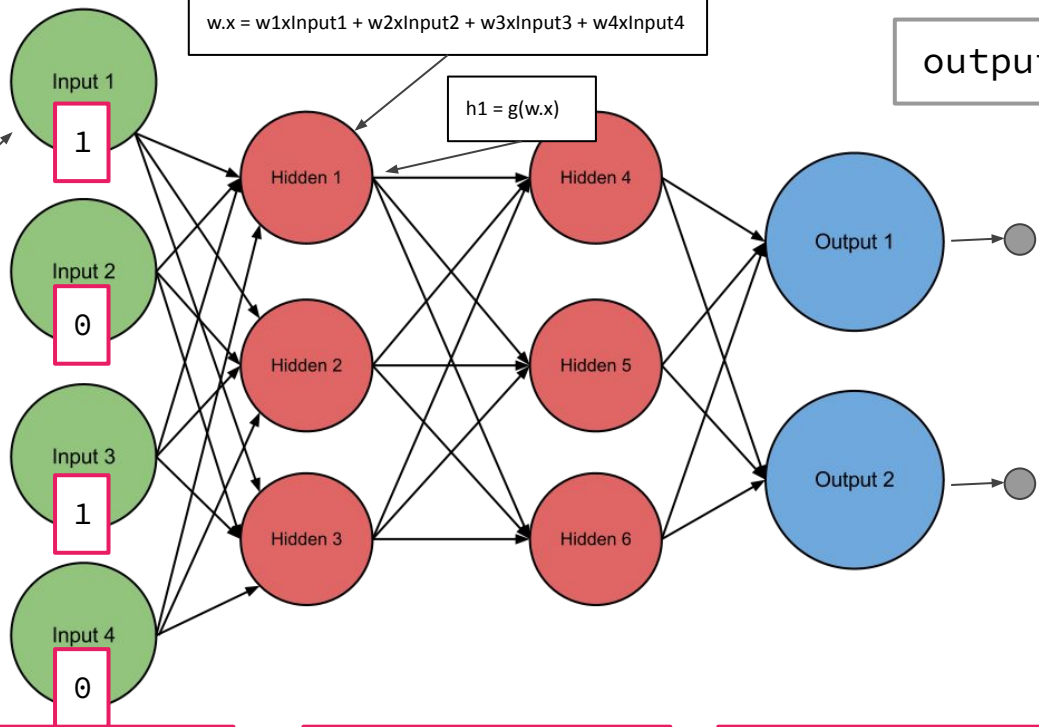
Filtre

This movie
is
excellent

This movie
~~is~~
excellent

movie: [1, 0, 0, 0]
excellent: [0, 0, 1, 0]
→ combined: [1, 0, 1, 0]

input



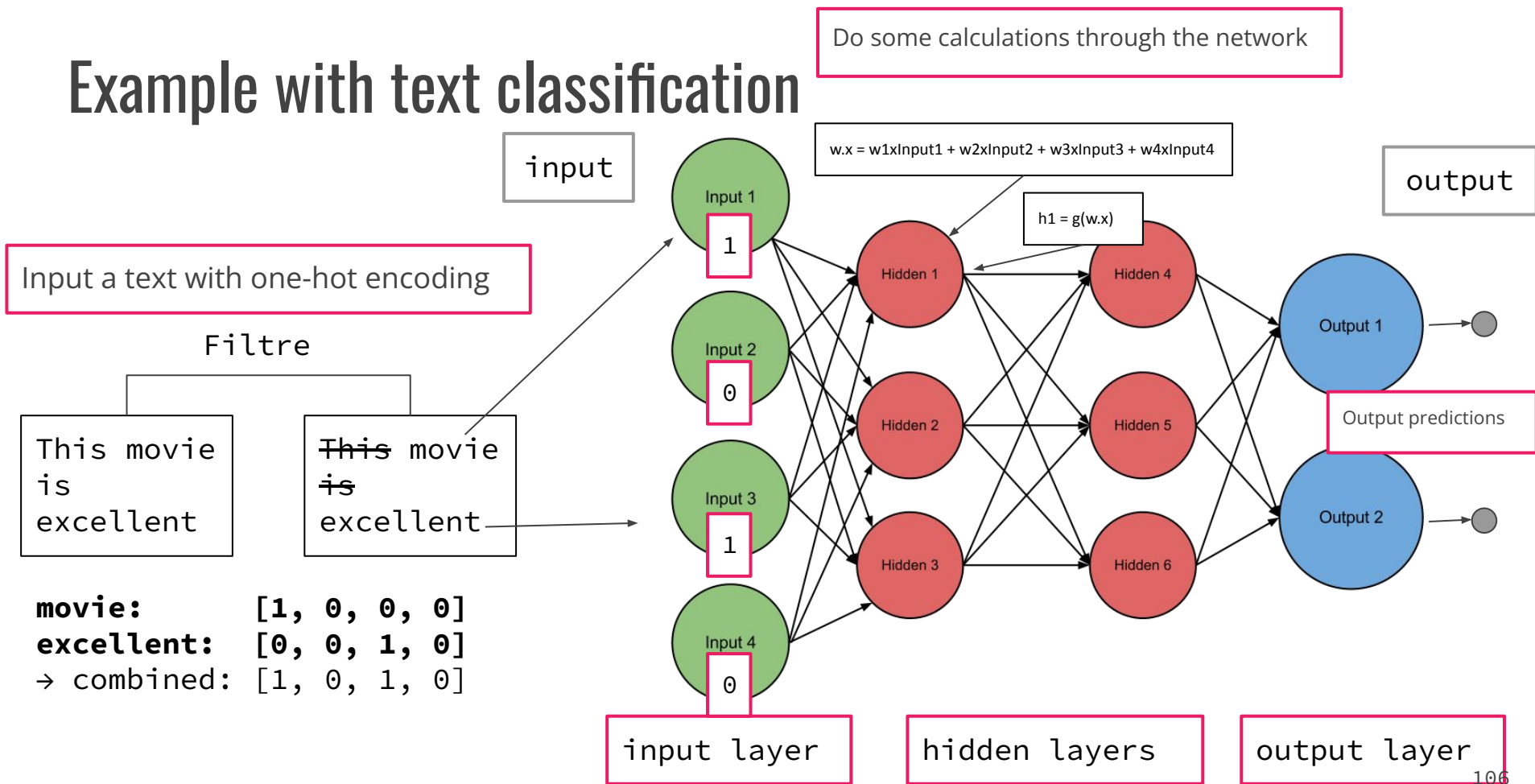
output

input layer

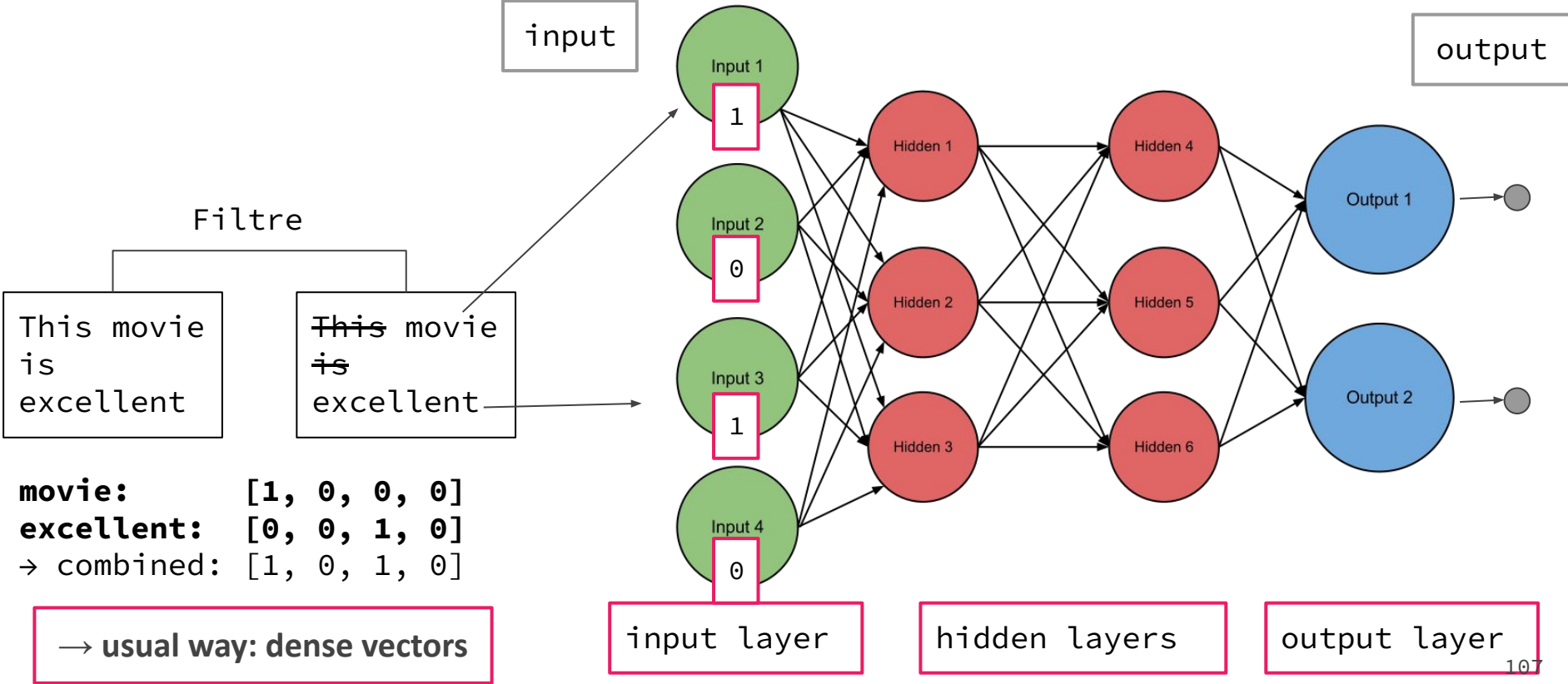
hidden layers

output layer

Example with text classification



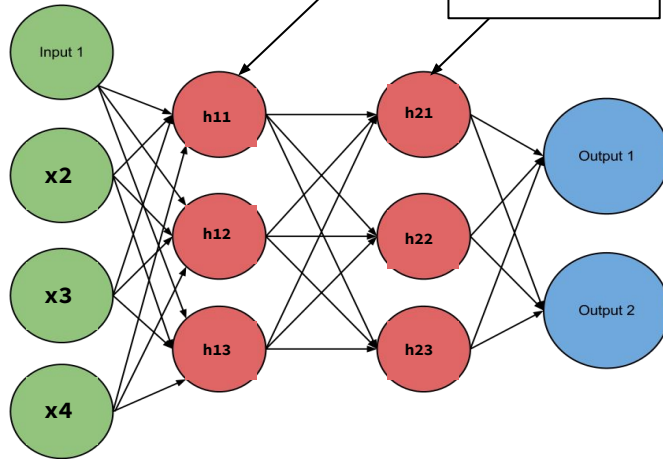
Example with text classification



Feed-Forward Neural Network

feed an "input" vector to a bunch of LR functions

repeat



= **Deep** learning
(here Depth = 2)

input layer

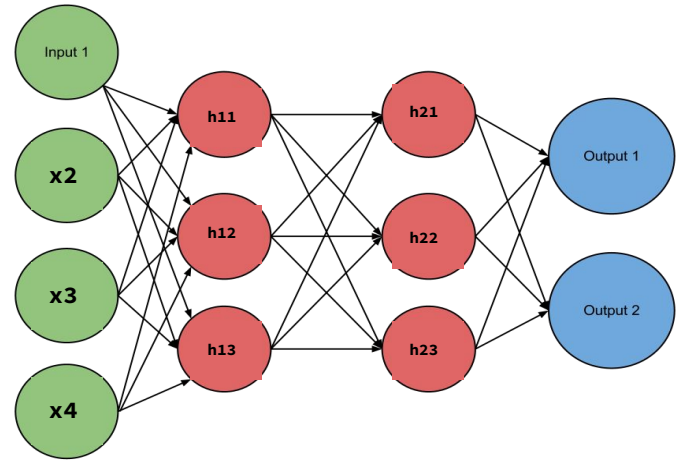
hidden layers

output layer

Feed-Forward Neural Network

Layers are all vectors of numerical values

- **layer** = vector resulting from a linear transformation
- each hidden layer is followed by a **non-linear activation**



MLP is a basic architecture, several improvements / modifications have been proposed

→ Before introducing more refined architectures: how to get a dense vector representing the input ?

Back to: text data representation

- Previous approach: defining features has to be done **manually**: require expertise and tests
- A word is represented with a **one-hot vector**: easy to implement

[o **1** o o o o]



[o o o o o o o **1** o o o o o o o o o o o o o o o o]



Data representation

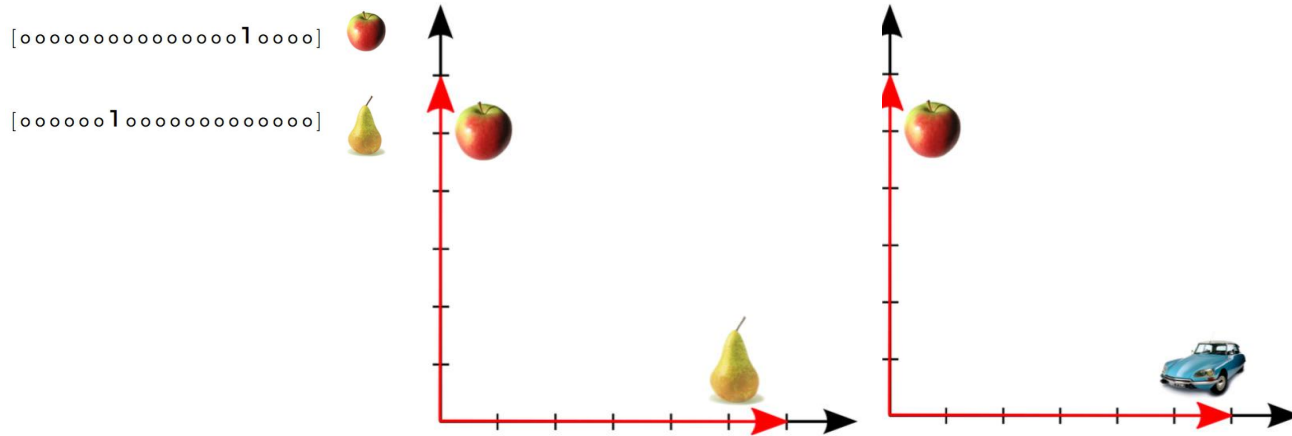
But does not represent the meaning of words → **no notion of similarity**

[ooooooooooooooooo1oooo] 

[oooooo1ooooooooooooo] 

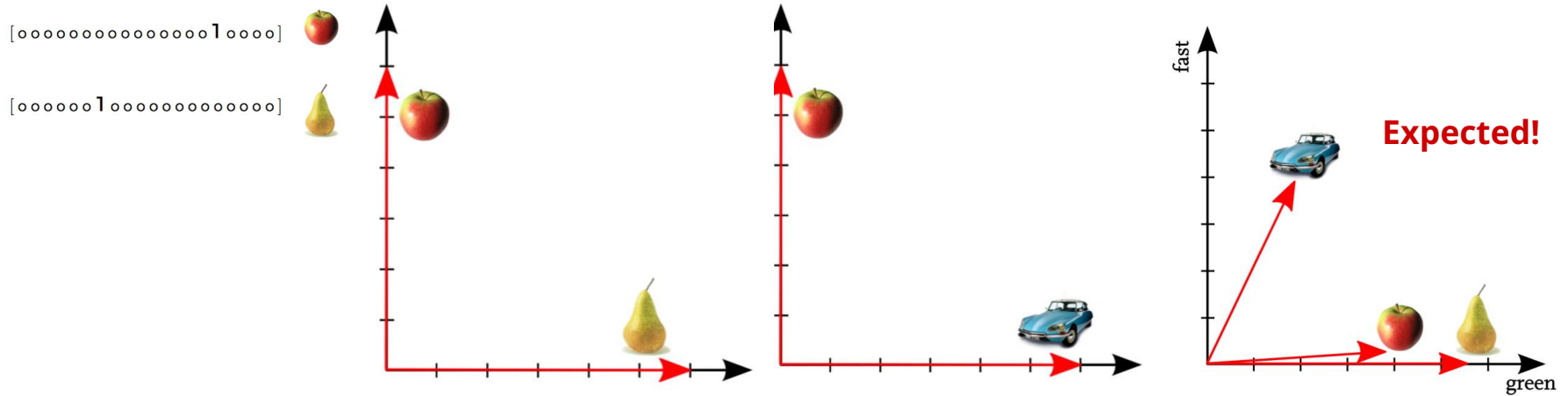
Data representation

But does not represent the meaning of words → **no notion of similarity**



Data representation

But does not represent the meaning of words → **no notion of similarity**



Word distribution

“You shall know a word by the company it keeps.”

—J.R. Firth



- Rather old idea: **distributional hypothesis** → 1950's!

Example (from Tim van der Cruys):

- delicious *sooluceps*
- sweet *sooluceps*
- stale *sooluceps*
- freshly baked *sooluceps*

→ Guess what is a *sooluceps* ?

Word distribution

“You shall know a word by the company it keeps.”

—J.R. Firth



- Rather old idea: **distributional hypothesis** → 1950's!

Example (from Tim van der Cruys):

- delicious *sooluceps*
- sweet *sooluceps*
- stale *sooluceps*
- freshly baked *sooluceps*

Food!



→ Guess what is a *sooluceps* ?

Looking at the context of use of a word, you can guess its meaning

Word embeddings

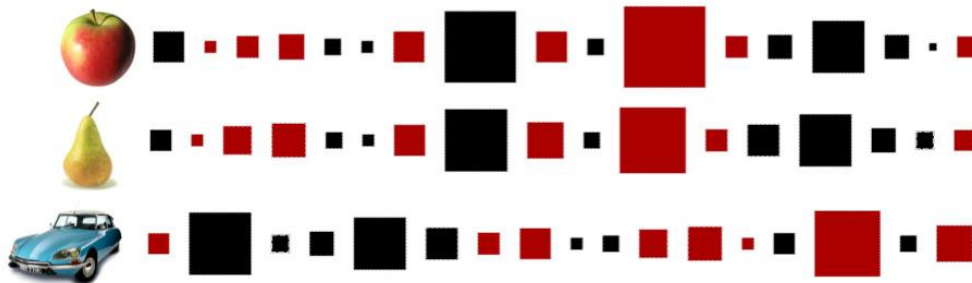
Many studies on representing the meaning of words using context

Before neural networks:

- build a matrix over all the words appearing in a corpus
- count the number of time words appear together
- reduce the dimensions (e.g. PCA)

Now: **Train a neural network to build a representation** / language model

- massive amount of data
- task = predicting a linguistic unit (word, sentence...)



Word2vec

For each target word (blue), consider some context words (here, window = 5)

CBOW:

- predict the target word given the context

The quick brown fox jumps over the lazy dog.

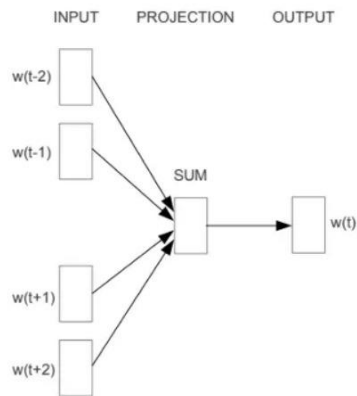
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

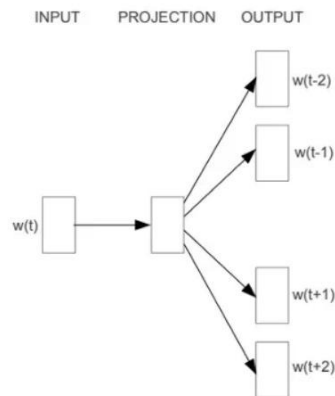
Skip-gram:

- predict the context words given the target word

→ the hidden layer is used as the representation of the target word = word embeddings



CBOW



Skip-gram

Word2vec

For each target word (blue), consider some context words (here, window = 5)

CBOW:

- **predict the target word** given the context

The quick **brown** fox jumps over the lazy dog.

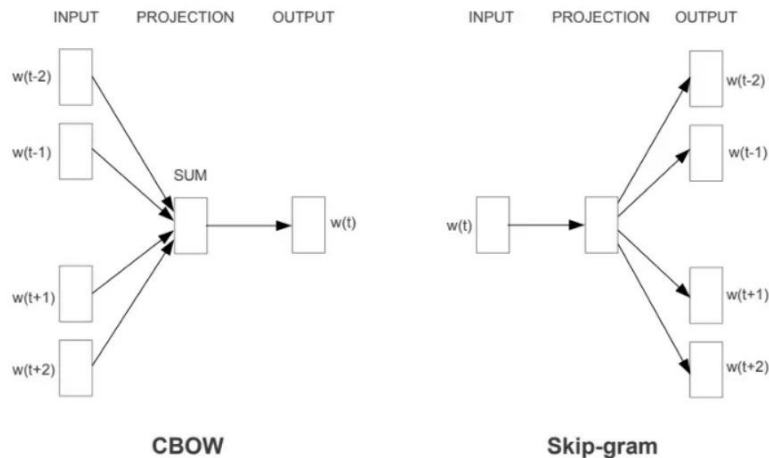
The quick brown **fox** jumps over the lazy dog.

The quick brown fox **jumps** over the lazy dog.

Skip-gram:

- predict the context words given the target word

→ the hidden layer is used as the representation of the target word = word embeddings



Word2vec

For each target word (blue), consider some context words (here, window = 5)

CBOW:

- predict the target word given the context

The quick **brown** fox jumps over the lazy dog.

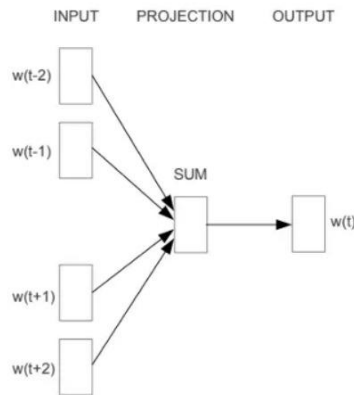
The quick brown **fox** jumps over the lazy dog.

The quick brown fox **jumps** over the lazy dog.

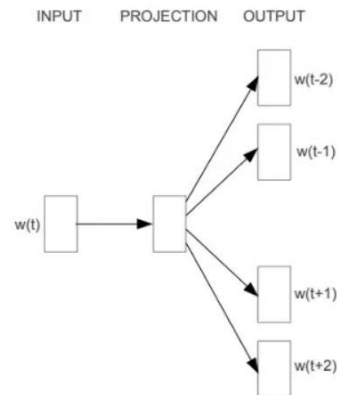
Skip-gram:

- **predict the context words** given the target word

→ the hidden layer is used as the representation of the target word = word embeddings



CBOW



Skip-gram

Word2vec

For each target word (blue), consider some context words (here, window = 5)

CBOW:

- predict the target word given the context

The quick brown fox jumps over the lazy dog.

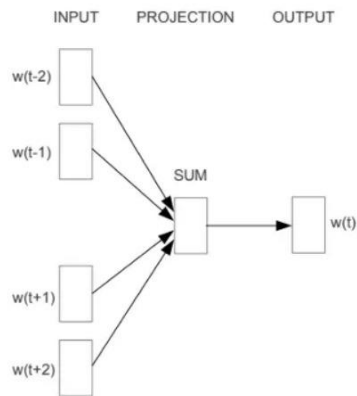
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

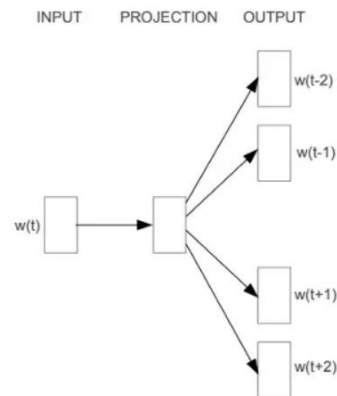
Skip-gram:

- predict the context words given the target word

→ the **hidden layer** is used as the **representation** of the target word = **word embeddings**

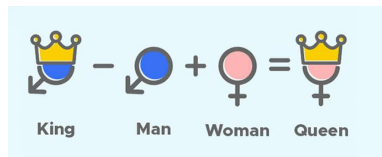


CBOW

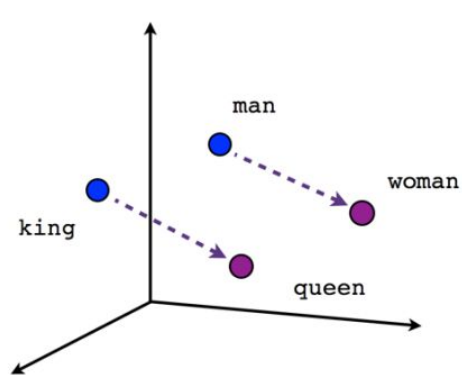


Skip-gram

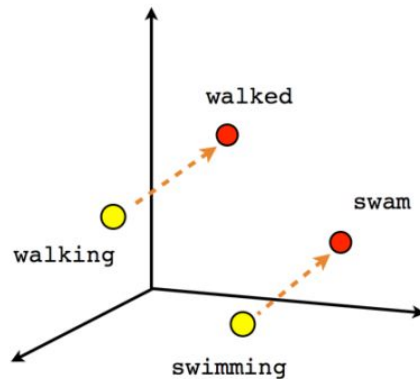
Word2vec



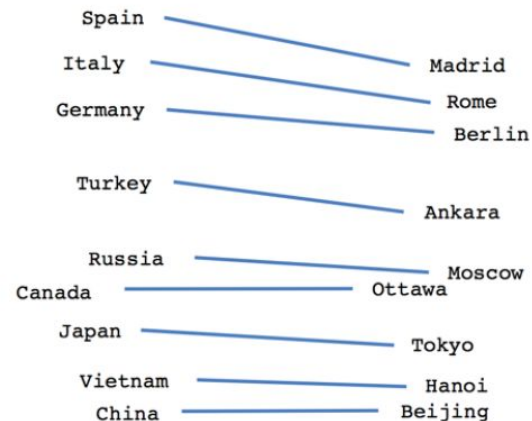
Learned word embeddings allow semantic inferences:



Male-Female



Verb tense



Country-Capital

Visualizing embeddings

<https://projector.tensorflow.org/>

5 tensors found
Word2Vec 10K

Label by: word
Color by: No color map

Supervise with: word
No ignored label

Edit by: word
Tag selection as

Load Publish Download Label

Spheroize data

Checkpoint: Demo datasets
Metadata: oss_data/word2vec_10000_200d_labels.tsv

UMAP T-SNE PCA CUSTOM

Dimension: 2D 3D

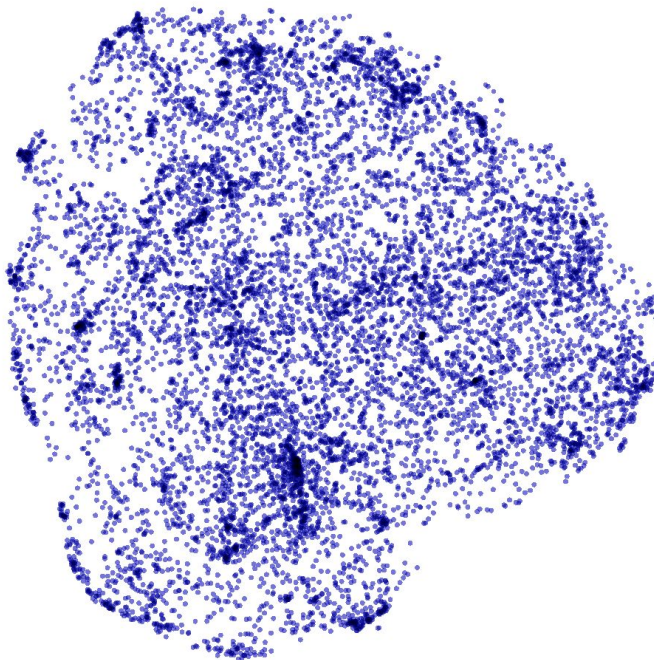
Perplexity: 25

Learning rate: 10

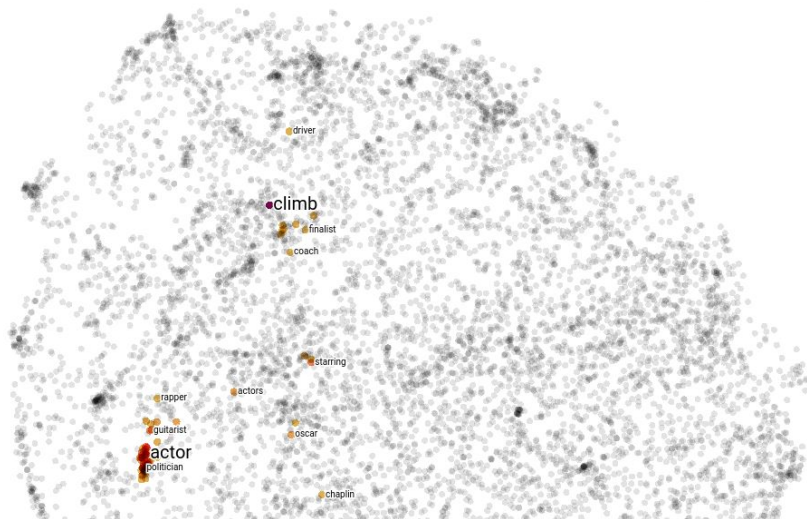
Supervise: 0

Re-run Pause Perturb

Iteration: 262



Visualizing embeddings



actor

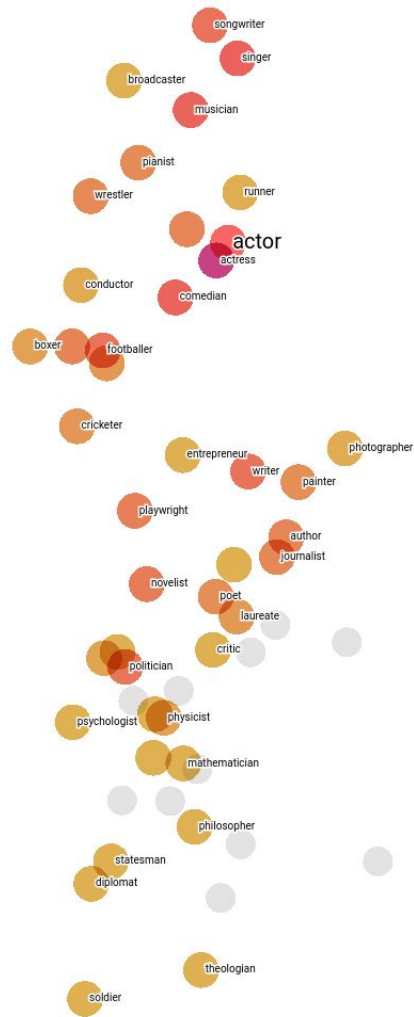
Search by actor word

neighbors 100

distance COSINE EUCLIDEAN

Nearest points in the original space:

actress	0.182
singer	0.308
comedian	0.323
musician	0.326
writer	0.375
songwriter	0.380
footballer	0.382
politician	0.390
novelist	0.417
playwright	0.422
athlete	0.442
composer	0.445
starring	0.448
author	0.449
journalist	0.457
wrestler	0.462
pianist	0.466



Data representation

→ Before NN: expertise needed to find good data representations

→ Now: feed your NN with word embeddings! but....

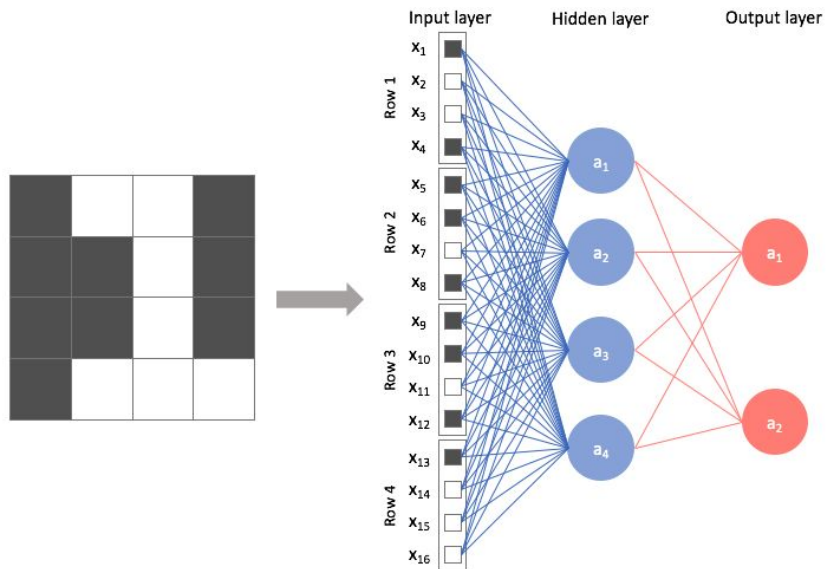
- Setting:
 - which ones? GloVe, FastText, Word2Vec, ELMO, BeRT, RobeRTa, GPT-2, GPT-3, XLNet...
 - which size, window size, number of iterations?
- Other issues:
 - how to combine them into a sentence / document?
 - what about other information: POS / syntax / pragmatics?
 - what about different languages and domains?
 - problem with evaluation: e.g. natural language inference tasks seem inadequate
 - choice of the data / problem with models: bias and representativeness

→ expertise still needed

Neural architectures

The Feed-forward neural networks are general purpose classification architectures, not tailored specifically for language data or sequences)

→ Feed-forward network assumes fixed dimensional input

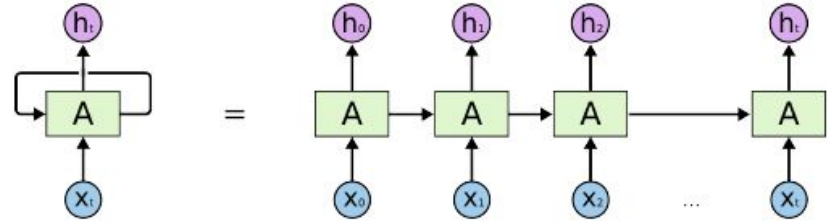


Neural architectures: dealing with text sequences

How to represent variable number of features, e.g. words in a sentence?

- Continuous Bag of Words (CBOW): sum embedding vectors of corresponding features
 - no ordering info e.g. "not good quite bad" = "not bad quite good"
- Convolutional layer
 - 'Sliding window' approach that takes local structure into account
 - Combine individual windows to create vector of fixed size
 - NLP: identifying informative ngrams in a sequence of text, regardless of their position but while taking local ordering patterns into account
- **Recurrent layer**
 - Allow to take into account the whole history / sequence
 - designed to capture subtle patterns and regularities in sequences

Recurrent Neural Network



- Main idea:
 - if we have data in a sequence such that one data point depends upon the previous data point
 - \rightarrow modify the neural network to **incorporate the dependencies** between these data points
- RNNs have the **concept of 'memory'** = store the states or information of previous inputs to generate the next output of the sequence.

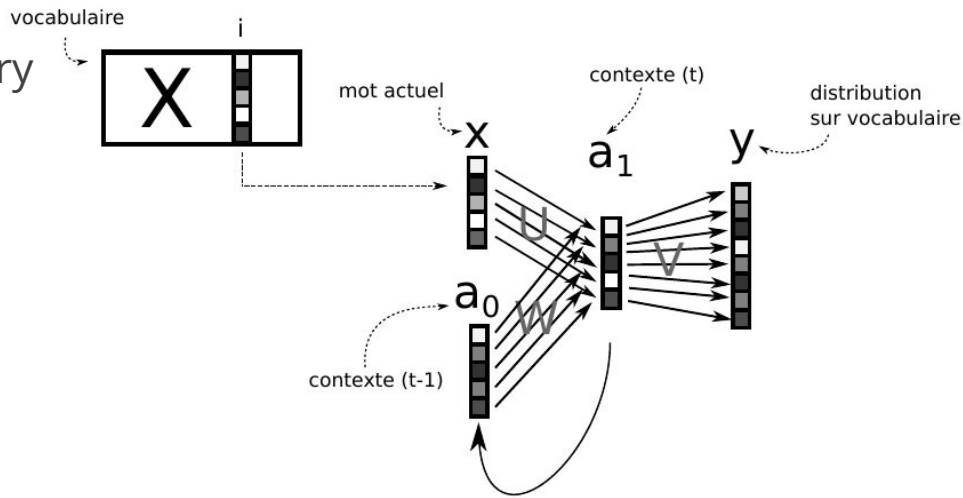
e.g. to predict the next word in a sentence, you need the previous outputs/words

\rightarrow LSTM or GRU are specific implementations of RNN

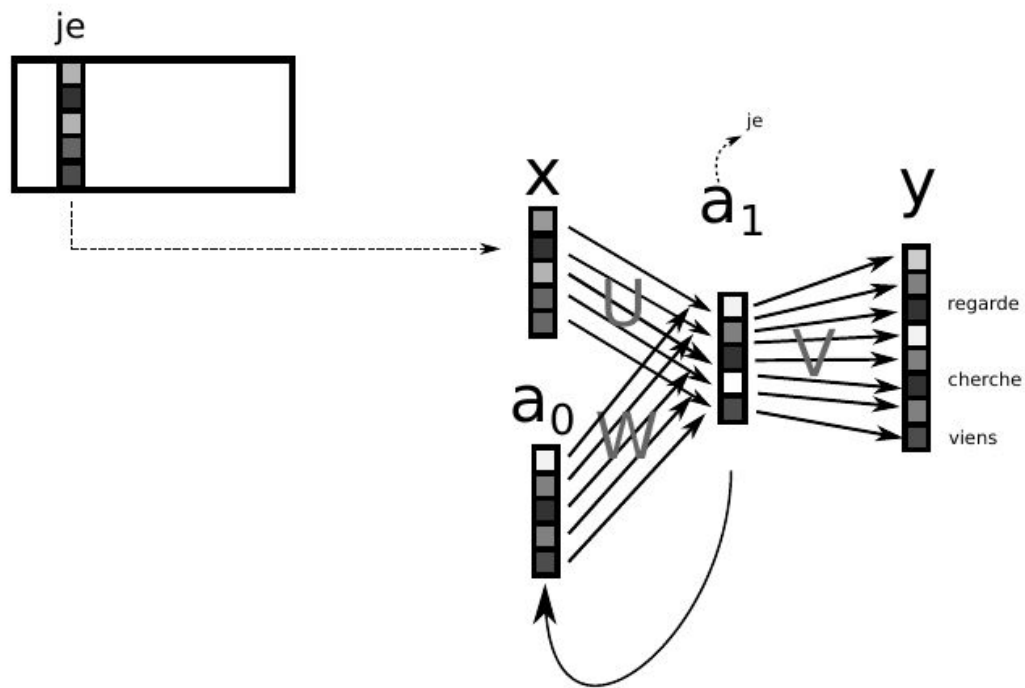
RNN: language model

- allow to condition on the entire history
- can act as **language models** → learning the likelihood of occurrence of a word based on the previous sequence of words (or based on characters, sentences, paragraphs)

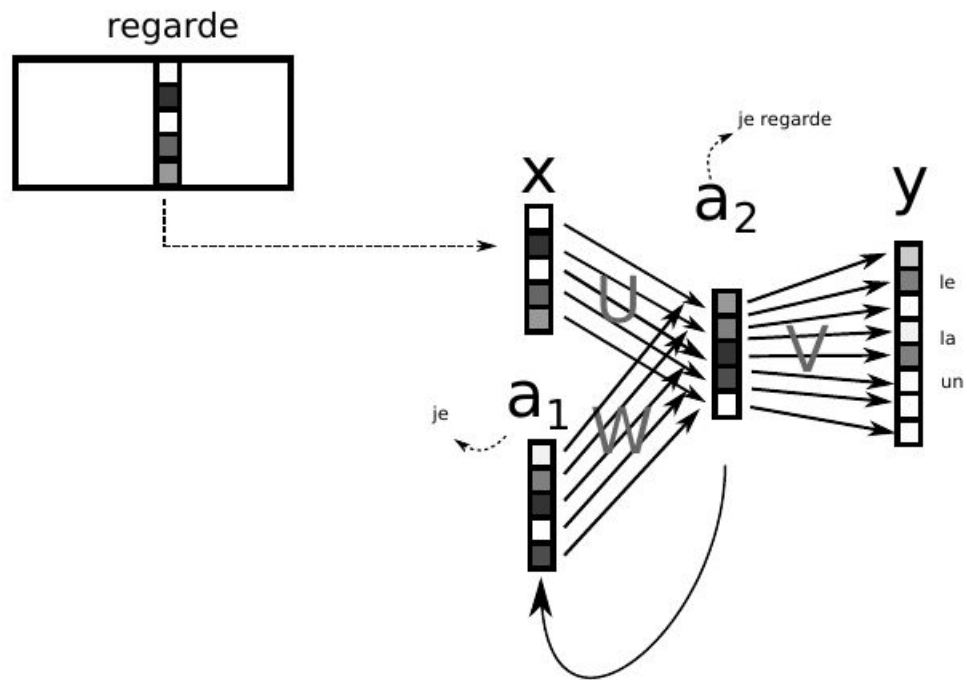
→ suitable for use as **generators**:
generating natural language sequences



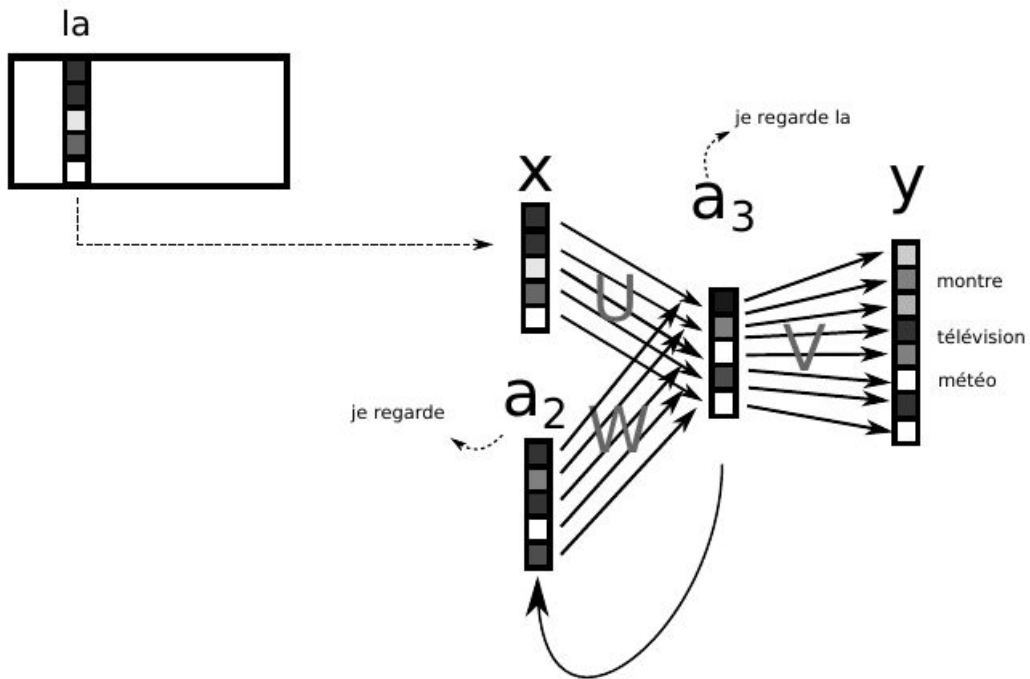
RNN: language model



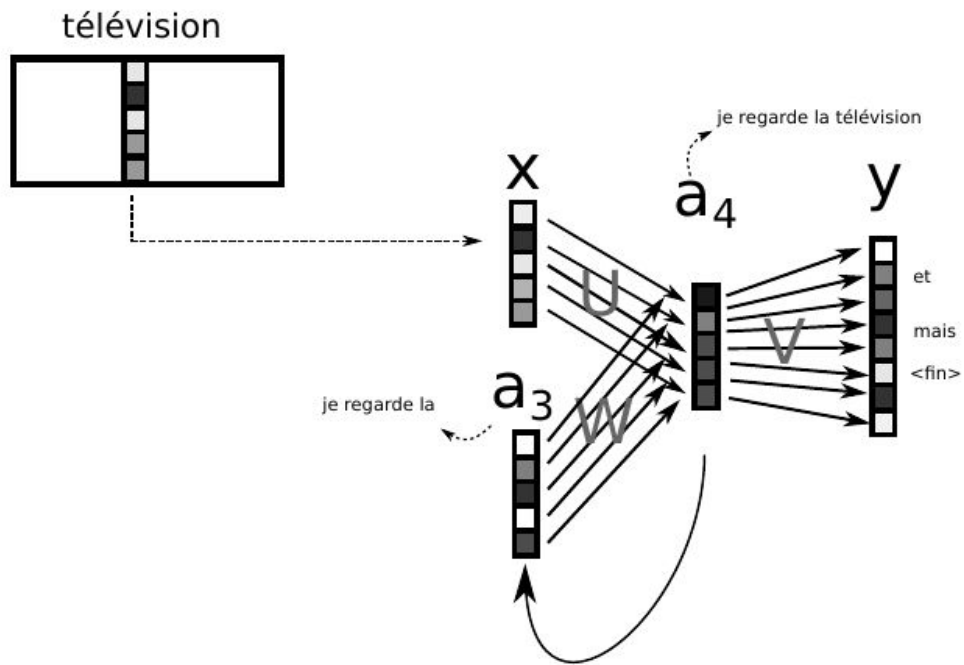
RNN: language model



RNN: language model



RNN: language model



Encoder-Decoder

- Until now RNN = generating the next token \mathbf{t}_{j+1}
 - based on the previously generated tokens $\mathbf{t}_{1:j}$
- **Conditioned generation** = the same
 - + an **additional conditioning context (vector) \mathbf{c}**

→ Combining encoding + generation = **encoder-decoder / sequence to sequence**

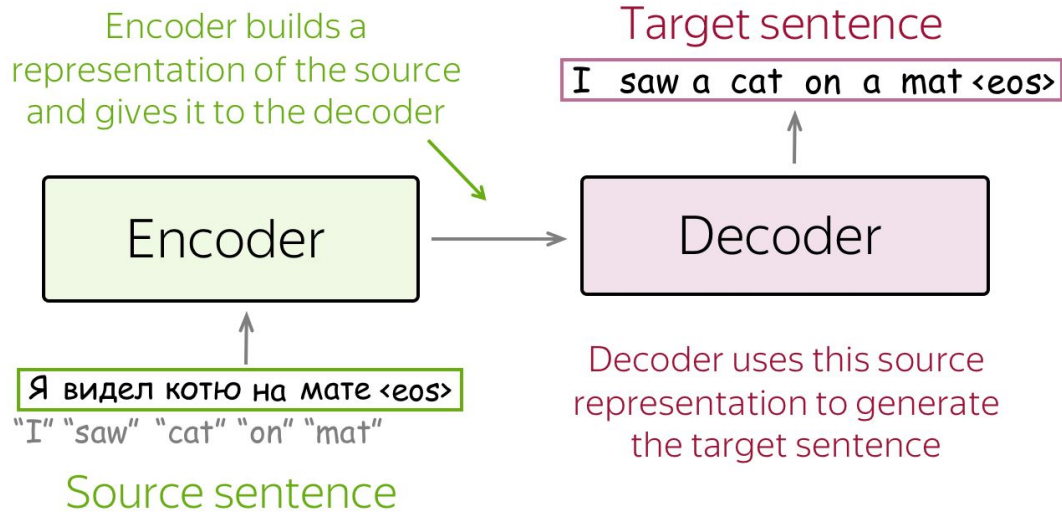
- **encoder:** produce a representation of the input (in general a sequence)
- **decoder:** generate an output (in general a sequence)

Typical example: Machine translation

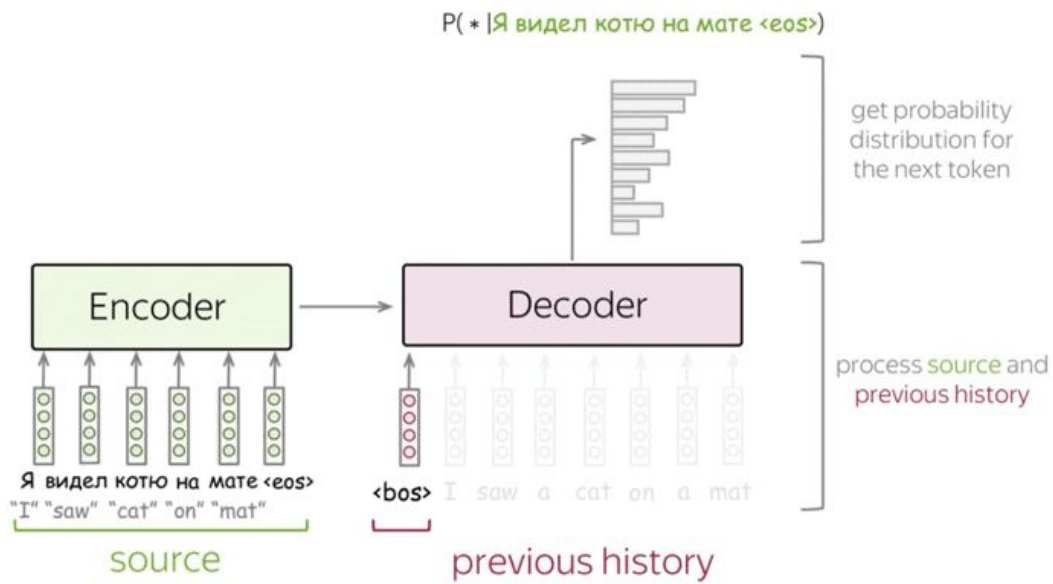
e.g. : Machine translation

- encoding the input in source language = produce a representation
- decoder: use the representation to condition the output in target language

decoder = generator of target language



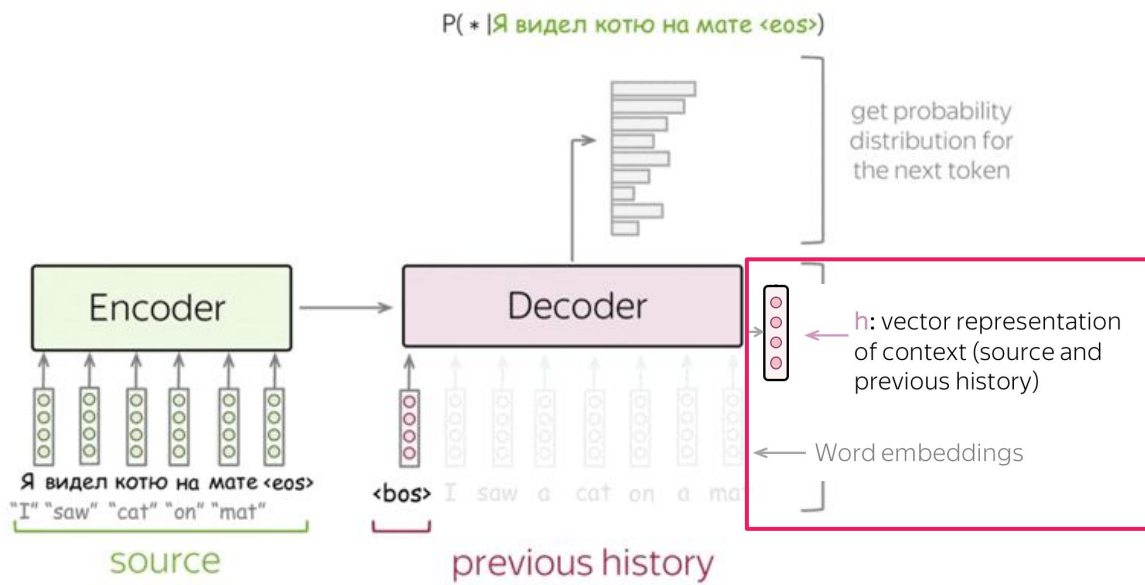
General idea



Pipeline:

- feed source and previously generated target words into a network;
- get vector representation of context (both source and previous target);
- from this vector representation, predict a probability distribution for the next token.

General idea



Neural networks learn representations:

- the hidden layers build representations of the input during learning
- the embeddings are updated during learning

→ representations tailored to the task

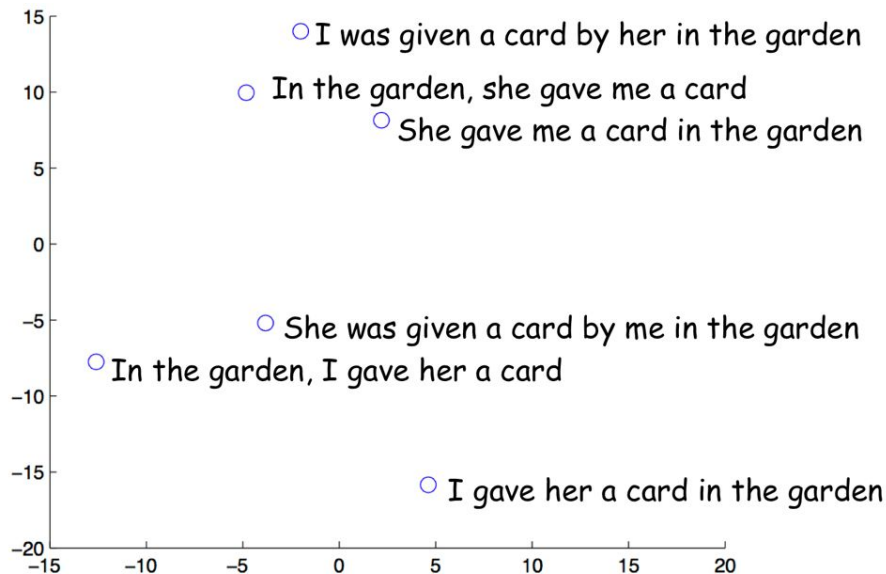
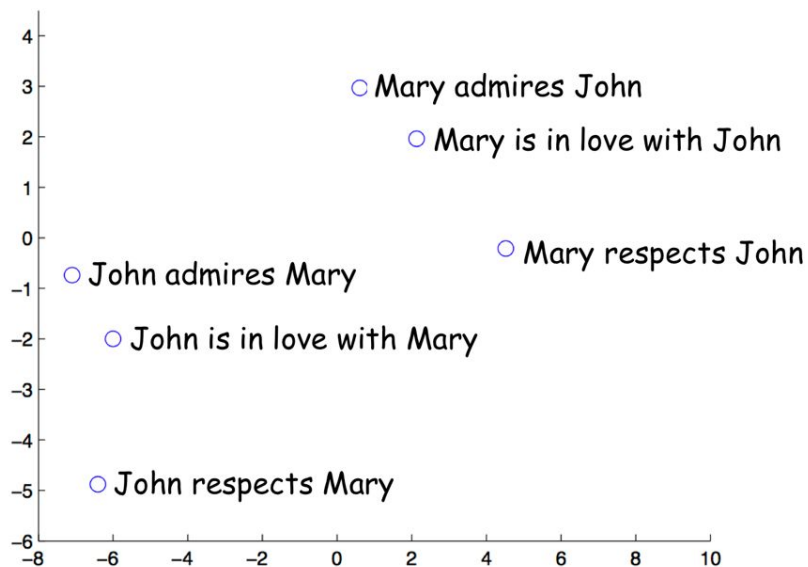
Encoder-Decoder: application examples

→ originally built to solve Seq2Seq problems, useful to map sequences of size **n** to sequences of length **m**

- **Machine translation:** in [\[Sutskever et al. 2014\]](#), they feed the source sentence in reverse (then x_n is the first word)
- **Email auto-response:** map an email to a short answer [Kannan et al 2016]
- **Morphological inflection:** input is a base word + inflection request, the output is an inflected form [Faruqui et al 2016]
- Other uses: almost **any task** can be formulated this way (but there could be better, easier to learn architectures). It has also been used for e.g. **sentence compression** by deletion [Filippova and Altun, 2013], **POS tagging** and **NER** [Gillick et al 2106], **syntactic parsing** using constituency bracketing decisions [Vinyals et al 2014]

Learned representation

In [\[Sutskever et al. 2014\]](#) (MT) they looked at the last encoder state and visualize several examples



Attention

Encoder-decoder = the input sentence is encoded into a single vector

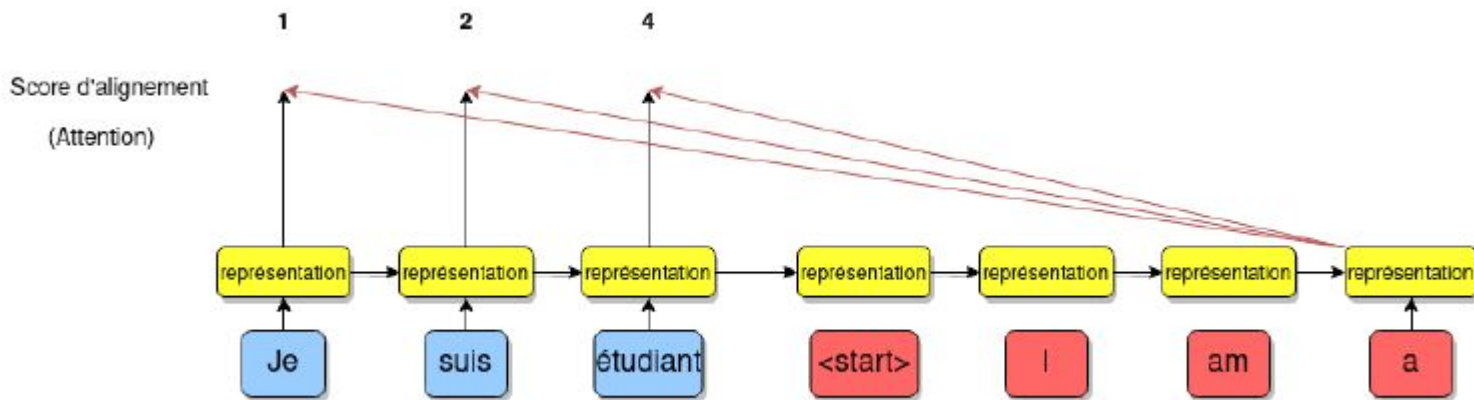
- the **encoder vector c must contain all the information required**
 - hard to compress a sentence
- the generator must extract information from this fixed-length vector
 - but different information may be relevant at different steps

This compression in one representation is suboptimal

→ ***attention mechanism*** to replace the representation learned with an RNN [[Bahdanau et al 2014](#); Luong et al 2015]

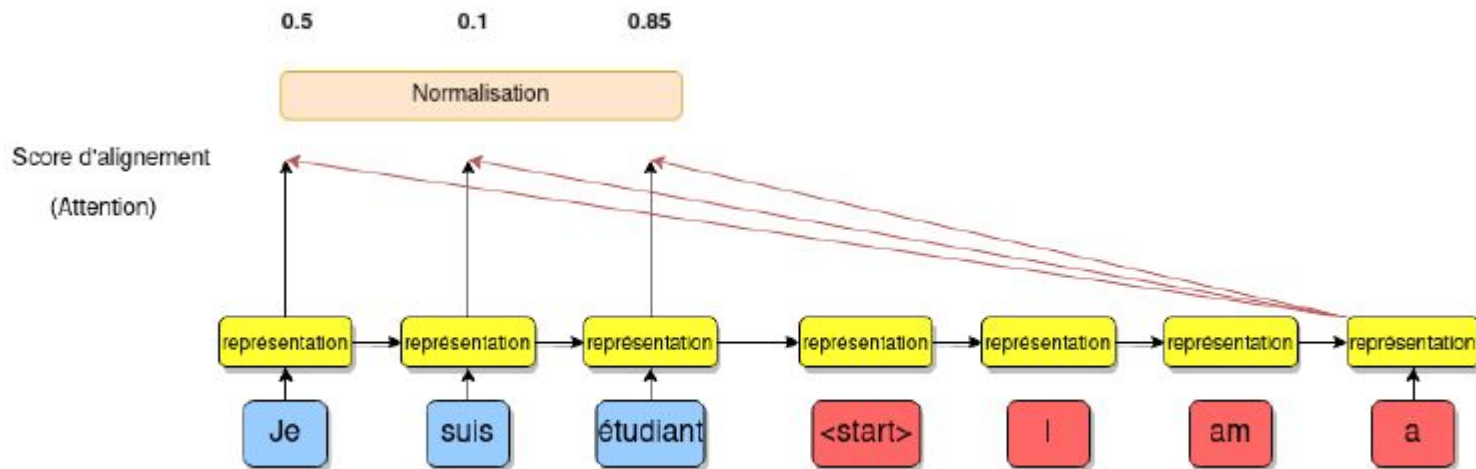
Attention

- at different steps, let a model 'focus' on different parts of the input
- at each step, the decoder decides on which parts of the encoding input it should focus / which source parts are more important



Attention

1. Assign some weights (and normalize)

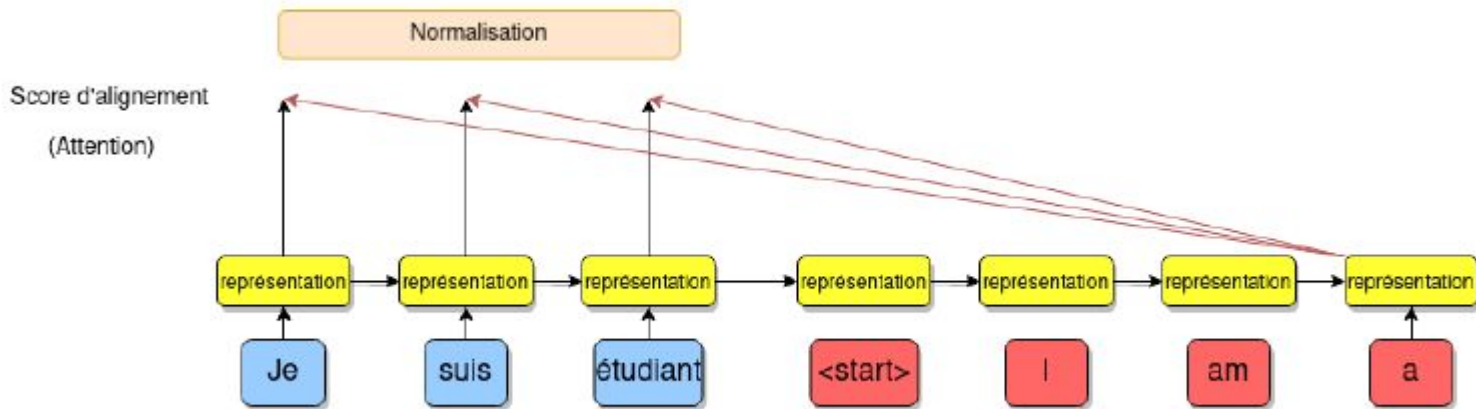


Attention

1. Assign some weights (and normalize)
2. Compute a context vector using weights and word embeddings

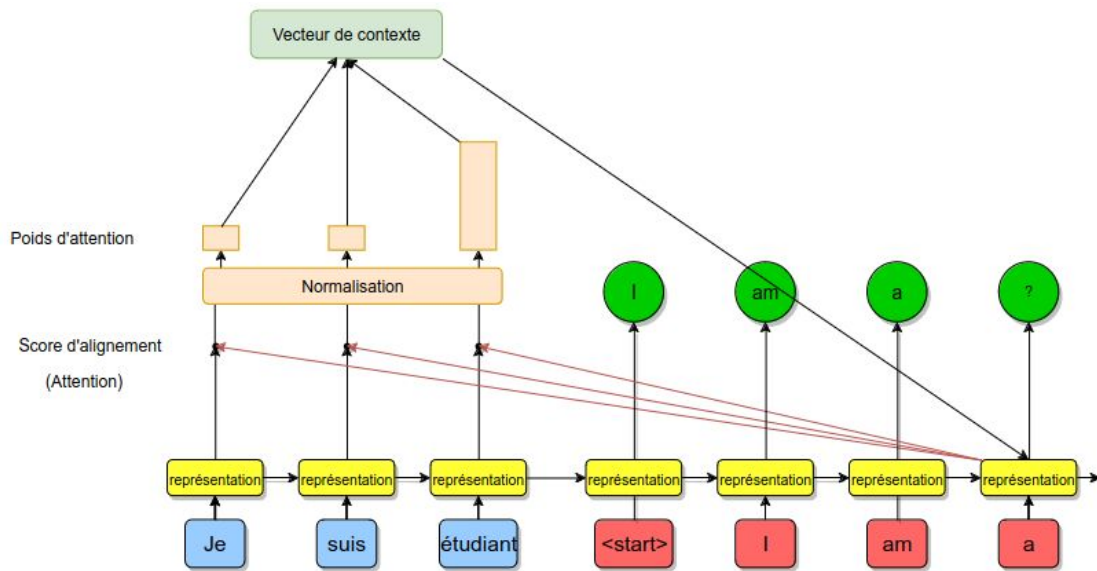
Vecteur de contexte : $0.05 \cdot \text{Je} + 0.1 \cdot \text{suis} + 0.85 \cdot \text{étudiant}$

0.5 0.1 0.85



Attention

1. Assign some weights (and normalize)
2. Compute a context vector using weights and word embeddings
3. Give the context vector to the decoder



Attention is all you need

Transformer models:



	Seq2seq without attention	Seq2seq with attention	Transformer
processing within encoder	RNN/CNN	RNN/CNN	attention
processing within decoder	RNN/CNN	RNN/CNN	attention
decoder-encoder interaction	static fixed-sized vector	attention	attention

- also takes sequence as input
- but based on attention mechanism **without the RNN architecture**
- = it is not required to read **in any order** the sequence

→ easier to parallelize computation, thus to train on larger corpora: leading to BERT, GPT language models

- [\[Vaswani et al 2017\]](#): new state-of-the-art on Machine translation (with “only” 3.5 days on eight GPUs :D), high performance for constituency parsing
- <https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>

General idea

- When encoding a sentence, RNNs won't understand what **bank** means until they read the whole sentence,
- Transformer's encoder tokens interact with each other all at once.

I arrived at the **bank** after crossing thestreet? ...river?

What does **bank** mean in this sentence?



I've no idea: let's wait until I read the end

RNNs

$O(N)$ steps to process a sentence with length N



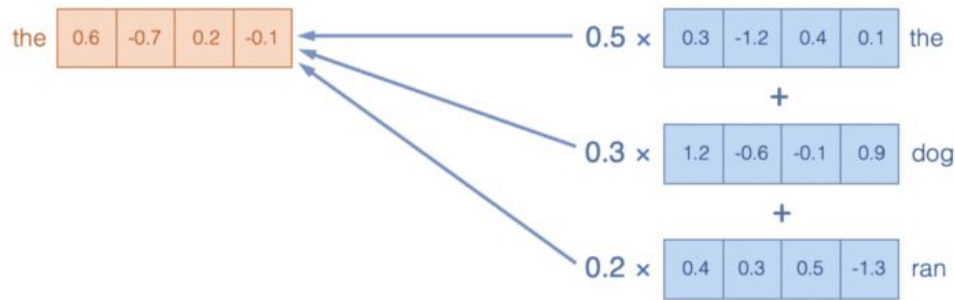
I don't need to wait - I see all words at once!

Transformer

Constant number of steps to process any sentence

- **Encoder < self-attention**: at each step, tokens look at each other, extract information and try to understand each other better in the context of the whole sentence
- **Decoder < self-attention** tokens predicted also interact with each other
- **Encoder-Decoder interaction < attention**: look at the context / encoder states

Self-Attention



Self-Attention = Attention over the sequence itself

Transformer model: relies entirely on **self-attention** to compute representations of its input and output (without using sequence aligned RNNs or convolution)

→ the model must understand how the words relate to each other in the context of the sentence

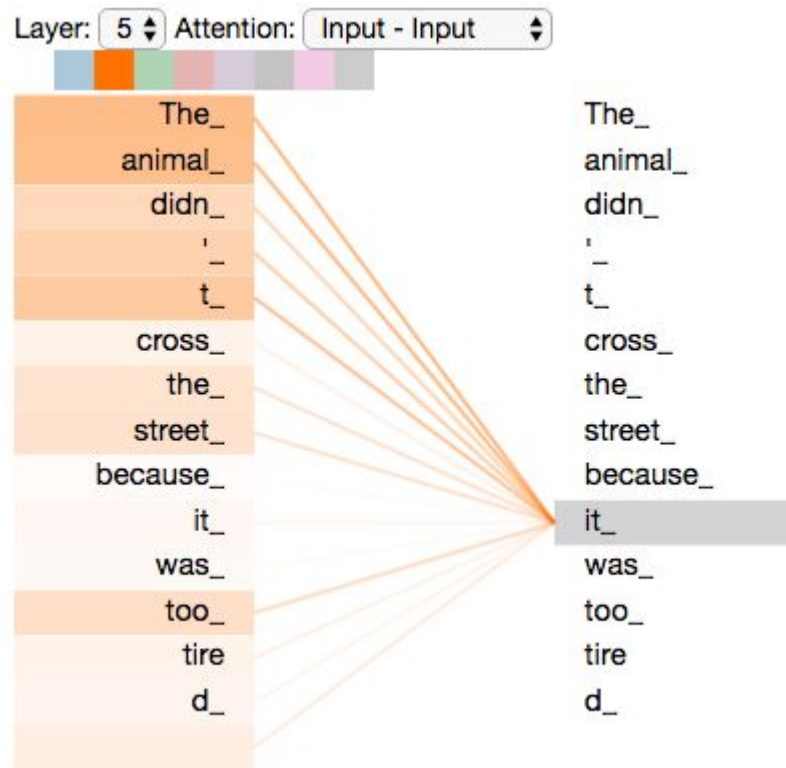
- used for reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [Cheng et al 2016, Parikh et al 2016, Lin et al 2017, Paulus et al 2017]

Self-Attention

Visualization:

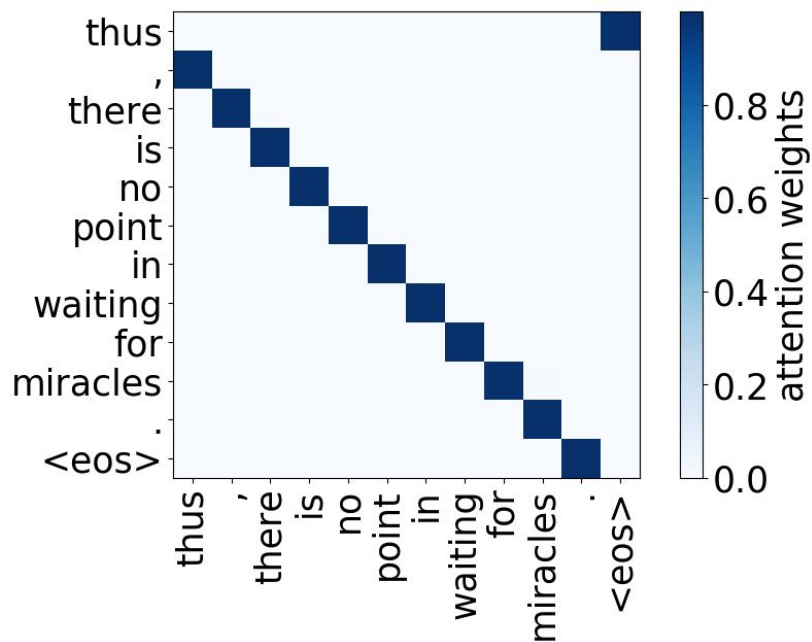
- the model puts a large attention weight between “the” and “animal” and “it”, allowing to ‘understand’ that “it” refers to “animal”

→ similar to the memory of RNNs, allow to keep an history

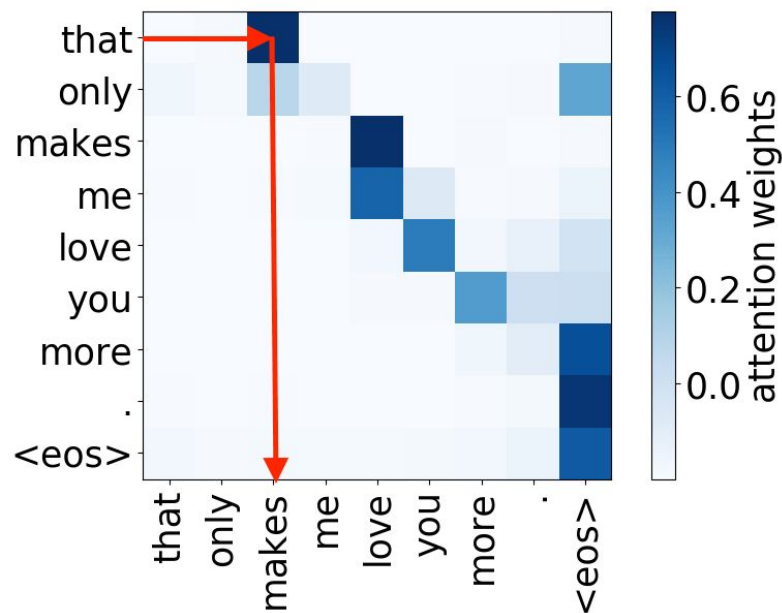


Multi-heads: multiple attention mechanisms

Positional heads

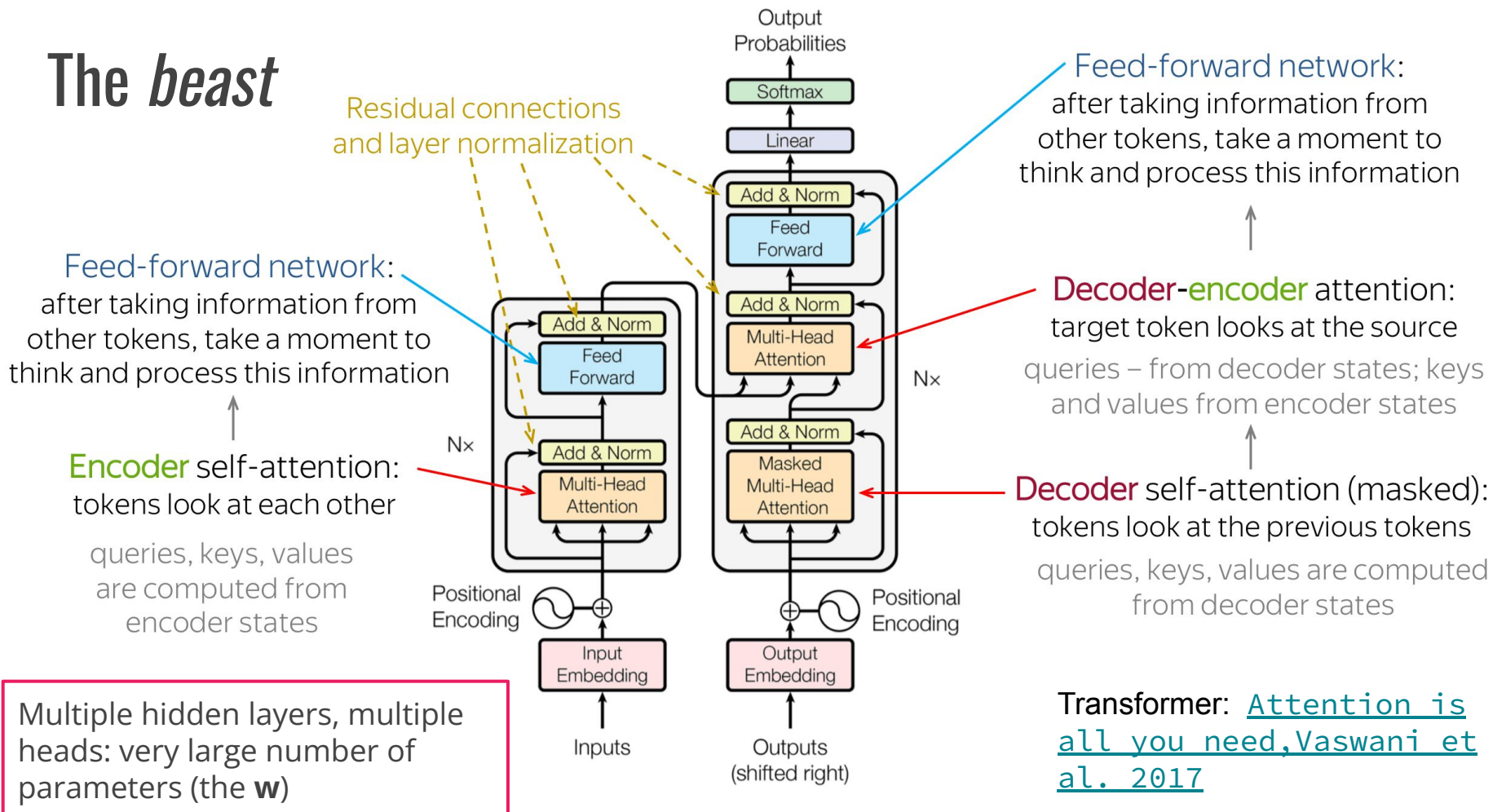


Syntactic heads (subject → verb)



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

The *beast*



Back to: text data representations

From GloVe to ELMo

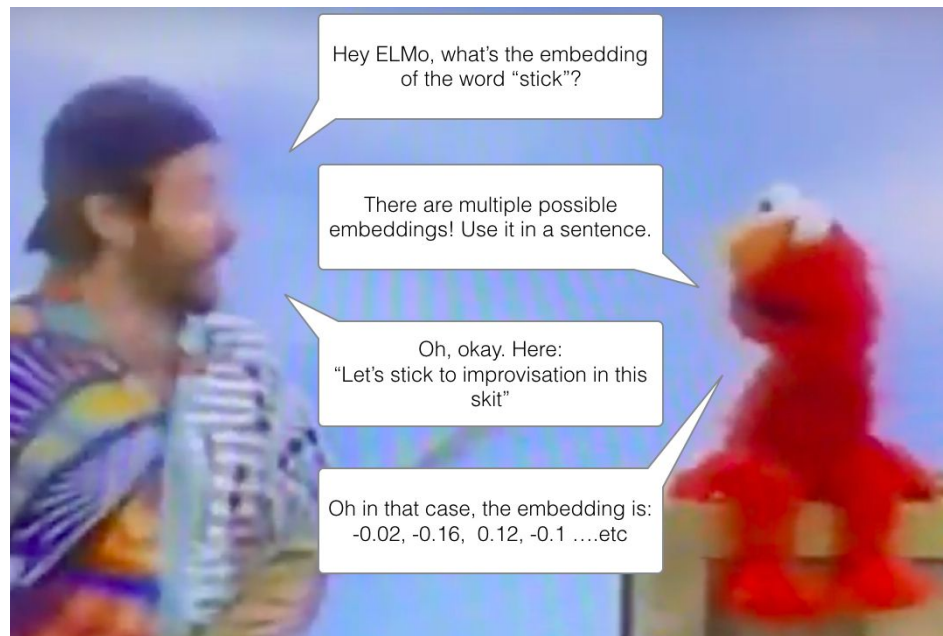
GloVe: the word “stick” would be represented by a unique vector no-matter what the context was.

But “stick” has multiple meanings depending on where it’s used.

→ give it an embedding based on its context

- takes into account an entire sentence as context
- should deal with polysemy

ELMo is based on RNNs, it was the first step toward contextual representations, but complex to use



Moving from ELMo to BERT*

Great ideas from ELMo:

- **contextual** embeddings
- learned from large amount of unlabeled data → **transfer learning**

But issues with sequential models

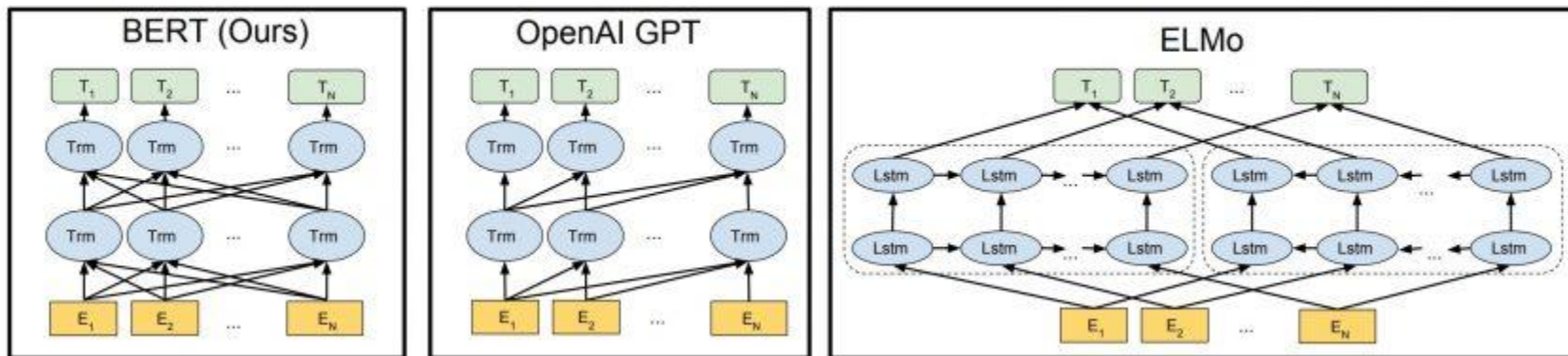
→ **use transformers**



ELMo vs BERT vs GPT

Use Transformers:

- GPT: Generative Pre-trained Transformer (OpenAI) [Radford et al., 2018]
- BERT: Bidirectional Encoder Representations from Transformers (Google) [\[Devlin et al 2019\]](#) and [post](#)

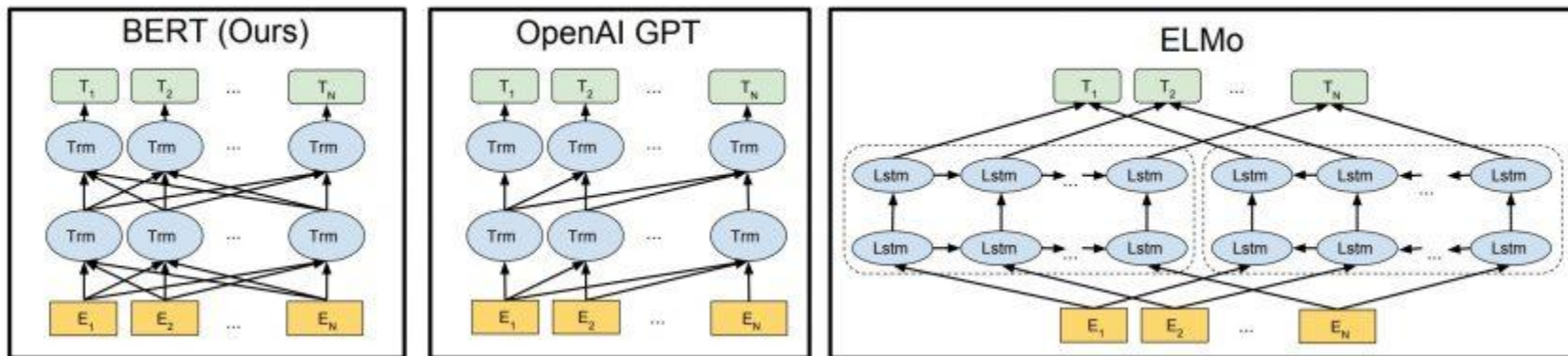


BERT is bi-directional, GPT is unidirectional (information flows only from left-to-right), and ELMo is shallowly bidirectional.

ELMo vs BERT vs GPT

Use Transformers:

- GPT: **Generative** Pre-trained Transformer (OpenAI) [Radford et al., 2018]
- BERT: **Bidirectional** Encoder Representations from Transformers (Google) [Devlin et al 2019] and post



BERT is bi-directional, GPT is unidirectional (information flows only from left-to-right), and ELMo is shallowly bidirectional.

Transformer-based Pre-trained Language Models

[\[Kalyan et al. 2022\]](#)

- PLMs learn universal language representations from large datasets using self-supervised learning
 - make use of unlabeled data to inject universal knowledge about language (and/or image, speech)
 - based on the pseudo supervision provided by pretraining tasks:
 - labels are automatically generated based on data attributes
 - e.g. masked language modelling (MLM), next sentence prediction (NSP) ...
- transfer this knowledge to downstream tasks

Transformer-based **P**re-trained **L**anguage **M**odels

[\[Kalyan et al. 2022\]](#)

- PLMs learn universal **language representations** from large datasets using self-supervised learning
 - make use of unlabeled data to inject universal knowledge about language (and/or image, speech)
 - based on the pseudo supervision provided by pretraining tasks:
 - labels are automatically generated based on data attributes
 - e.g. masked language modelling (MLM), next sentence prediction (NSP) ...
- transfer this knowledge to downstream tasks

Transformer-based **P**re-trained **L**anguage **M**odels

[\[Kalyan et al. 2022\]](#)

- PLMs learn universal **language representations** from large datasets using **self-supervised learning**
 - make use of unlabeled data to inject universal knowledge about language (and/or image, speech)
 - based on the pseudo supervision provided by pretraining tasks:
 - labels are automatically generated based on data attributes
 - e.g. masked language modelling (MLM), next sentence prediction (NSP) ...
- transfer this knowledge to downstream tasks

Transformer-based **P**re-trained **L**anguage **M**odels

[\[Kalyan et al. 2022\]](#)

- PLMs learn universal **language representations** from large datasets using **self-supervised learning**
 - make use of **unlabeled data** to inject universal knowledge about language (and/or image, speech)
 - based on the pseudo supervision provided by pretraining tasks:
 - labels are automatically generated based on data attributes
 - e.g. masked language modelling (MLM), next sentence prediction (NSP) ...
- transfer this knowledge to downstream tasks

Transformer-based **P**re-trained **L**anguage **M**odels

[\[Kalyan et al. 2022\]](#)

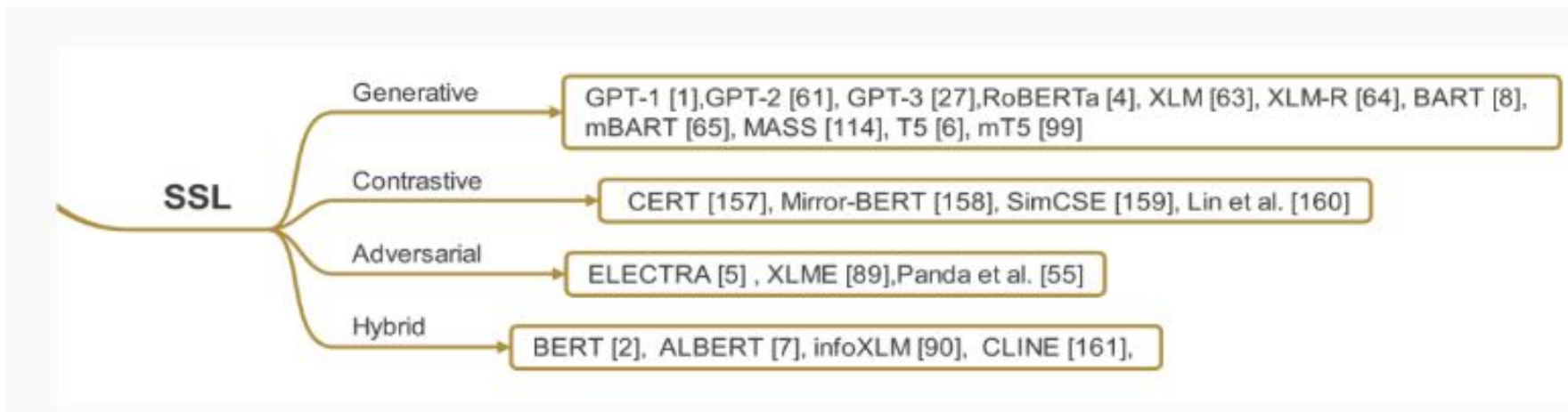
- PLMs learn universal **language representations** from large datasets using **self-supervised learning**
 - make use of **unlabeled data** to inject universal knowledge about language (and/or image, speech)
 - based on the **pseudo supervision** provided by pretraining tasks:
 - **labels are automatically generated** based on data attributes
 - e.g. masked language modelling (MLM), next sentence prediction (NSP) ...
- transfer this knowledge to downstream tasks

Transformer-based **P**re-trained **L**anguage **M**odels

[\[Kalyan et al. 2022\]](#)

- PLMs learn universal **language representations** from large datasets using **self-supervised learning**
 - make use of **unlabeled data** to inject universal knowledge about language (and/or image, speech)
 - based on the **pseudo supervision** provided by pretraining tasks:
 - **labels are automatically generated** based on data attributes
 - e.g. masked language modelling (MLM), next sentence prediction (NSP) ...
- **transfer this knowledge** to downstream tasks

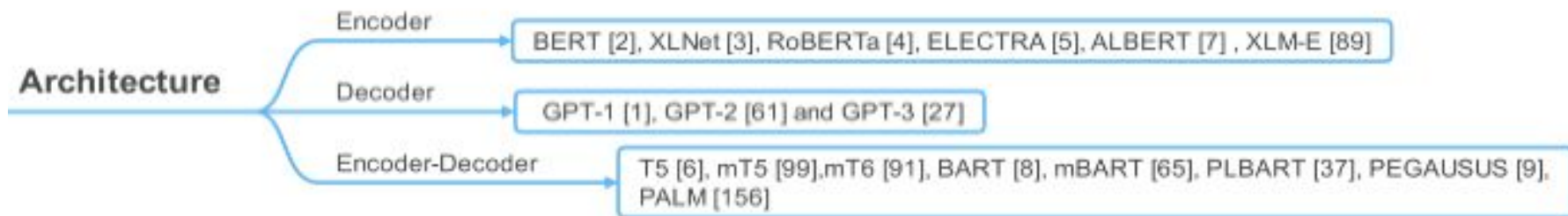
Types of self-supervised learning



- **Generative SSL:** learn by decoding the encoded input:
 - autoregressive = predict the next tokens based on the previous ones (GPT-1)
 - autoencoding = predict the masked tokens based on the unmasked ones (e.g. MLM)
- **Contrastive SSL:** learn by comparing
 - NSP or Sentence Order Prediction
- **Adversarial SSL:** learn to identify whether tokens are replaced or shuffled
- **Hybrid:** e.g. BERT = generative (MLM) and contrastive (NSP)

Types of architectures

- encoder only:
 - embedding layer followed by a stack of encoder layers ; e.g. BERT, XLNet, RoBERTa, ELECTRA, ALBERT and XLM-E - NLU
- decoder only:
 - an embedding layer followed by a stack of decoder layers ; e.g. GPT-1, GPT-2 and GPT-3 - NLG
- encoder-decoder:
 - T5, mT5, mT6, BART, mBART, PLBART, PEGAUSUS and PALM



Example of BERT: Language model with transformer

→ **non-directional**: trying to predict a word without direction, can't be "the next word"

BERT = 2 tasks, trained jointly:

- **Masked Language Model**: 15% of the input tokens are masked, the model has to predict the missing tokens = a classification layer on top of the encoder
- **Next sentence prediction**: the model receives pairs of sentences as input and learns to predict if the second sentence is the subsequent sentence in the original document (50% are randomly replaced) = another classification layer

	Training Compute + Time	Usage Compute
BERT _{BASE}	4 Cloud TPUs, 4 days	1 GPU
BERT _{LARGE}	16 Cloud TPUs, 4 days	1 TPU



Using BERT

The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications.

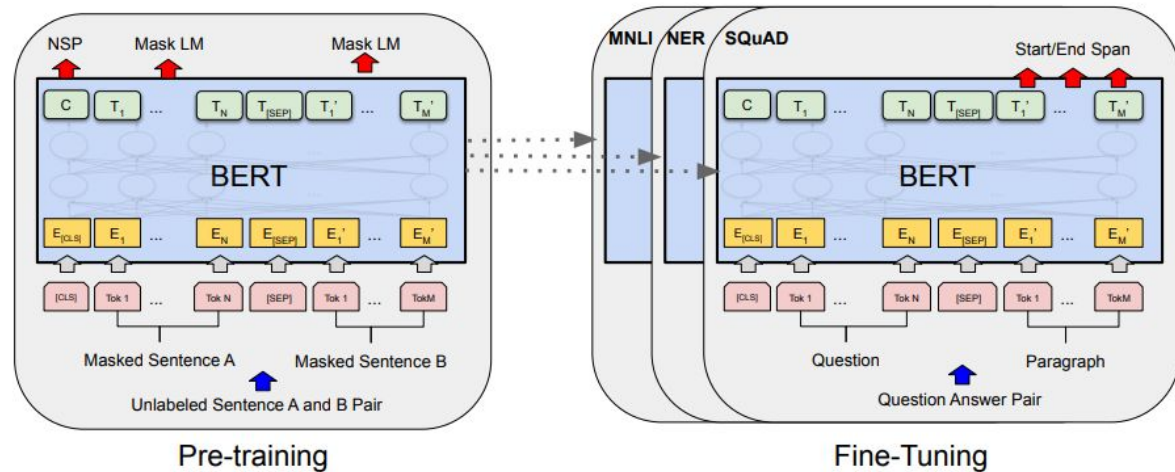


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Little trick

Words are split into frequent subwords ([algorithm WordPiece](#))

→ frequently used words should not be split but rare words should be decomposed into meaningful subwords

- decrease the vocabulary size
- less unknown words
- use morphological information

Model dependent: “Tokenization is difficult”

- Model 1: 'token', 'ization', 'is', 'difficult'
- Model 2: 'to', 'ken', 'ization', 'is', 'difficult'

https://huggingface.co/docs/transformers/tokenizer_summary

BERT: results

The [General Language Understanding Evaluation \(GLUE\) benchmark](#) (Wang et al., 2018a) is a collection of diverse natural language understanding tasks

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

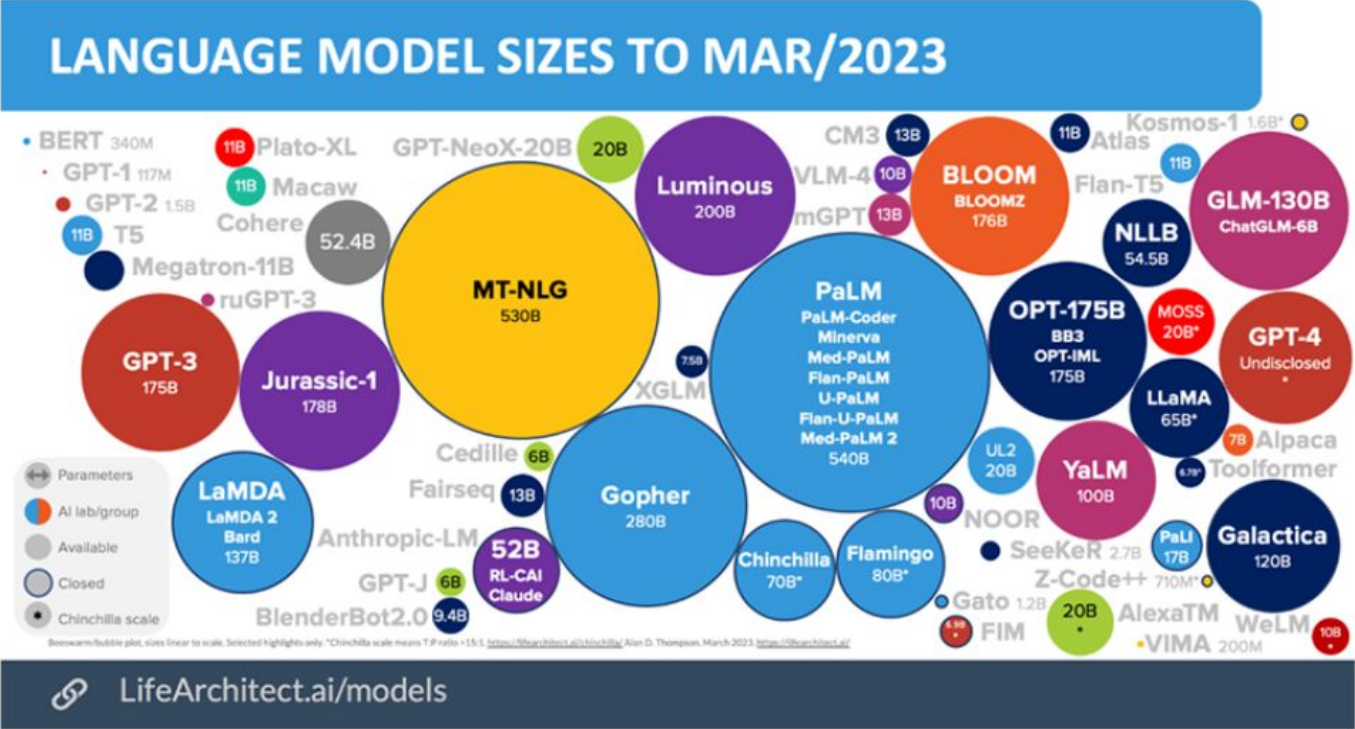
Table 1: GLUE Test results, scored by the evaluation server (<https://gluebenchmark.com/leaderboard>).

ChatGPT and the issues with LLMs

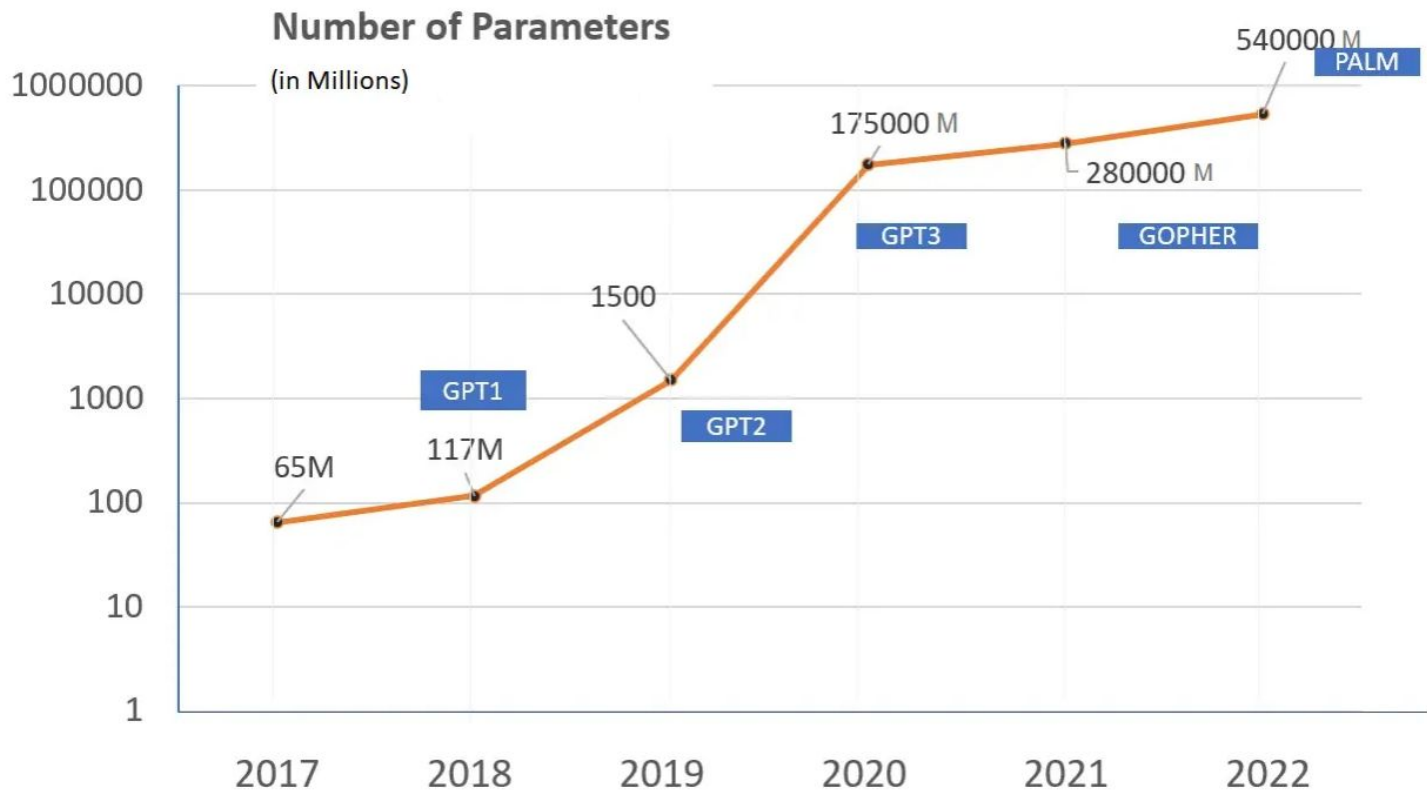
LLM stands for: Large Language Model = PLM that is considered 'large'

- ChatGPT is built on PLMs (GPT3 or 4) = a model trained to predict the next word, and able to output a sequence given an input sequence
 - → ChatGPT is a wonder of engineering (fast in prediction, trained on very very large datasets, fluent) but built on tools known for some time now (not exactly a scientific revolution) + use human feedback
- Training such models is very expensive: that's why they mostly come from GAFAM
- But these companies don't share details about the models, not reproducible, not open-source, not free to use (not open science)
- In particular: we don't know on which data they were trained: they could 'see/learn' our evaluation data, making it impossible to evaluate their progress
- + other issues: they are as biased as the data (and we don't know what the creator do about that), ethical issues (the human annotators behind), data linkage, not trustworthy, not explainable...

Language models size



Language models size



Language models size



How to use BERT and co?



- Models are on: <https://github.com/google-research/bert>
- Easier way: using HuggingFace library Transformers <https://huggingface.co/docs/transformers/index>

BERT like:

- RoBERTa (Robustly Optimized BERT Approach) - Facebook: without the NSP task, faster to train, bigger dataset, more iterations (16GB → 160GB, 100k → 300/500k epochs)
- [ALBERT \(A Lite BERT\)](#): 18× less parameters, just marginally worse performance
- FlauBERT: for French [Hang et al 2020] <https://github.com/getalp/Flaubert>
- [LegalBERT](#): pre-trained on the [Pile of Law](#), a dataset consisting of ~256GB of English language legal and administrative text

Beyond fixed-size sequences: [XLNet](#): using Transformer-XL, improved training methodology, larger data and more computational power

Multilingual : XLMRoBERTa, mBERT ...

Datasets: HuggingFace also makes available [tons of datasets](#) (language, image, video) to play with, including e.g. [LegalBench](#)

Current challenges in NLP



Current challenges in NLP

- across languages and domains, beyond textual data
- bias and "fairness"
- interpretability
- environmental and financial costs
- reproducibility, data sharing
- robustness

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?

[http://faculty.washington.edu/ebender/papers/Stochastic Parrots.pdf](http://faculty.washington.edu/ebender/papers/Stochastic_Parrots.pdf)

Current challenges in NLP

- **across languages and domains, beyond textual data**
- **bias and "fairness"**
- **interpretability**
- **environmental and financial costs**
- reproducibility, data sharing
- robustness

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?

[http://faculty.washington.edu/ebender/papers/Stochastic Parrots.pdf](http://faculty.washington.edu/ebender/papers/Stochastic_Parrots.pdf)

Across languages and domains, beyond text data

- Low-resourced languages
 - 100 languages covered by LM vs 5000-7000 languages in the world (+ sign languages)
- Most work on:
 - specific domains e.g. news, wikipedia ... more and more on social media. More work needed on specific domain: technical, medical, legal etc
 - mostly work on monologues, but increasing work on dialogues / spoken language
- Multi-modality:
 - communication also based on e.g. image, audio (see data [here](#)), also an active domain!

text-to-video prediction: can be used to automatically illustrate a set of instructions



video-to-text: automatic captioning



GT: add some chopped basil leaves into it

VideoBERT: chop the basil and add to the bowl

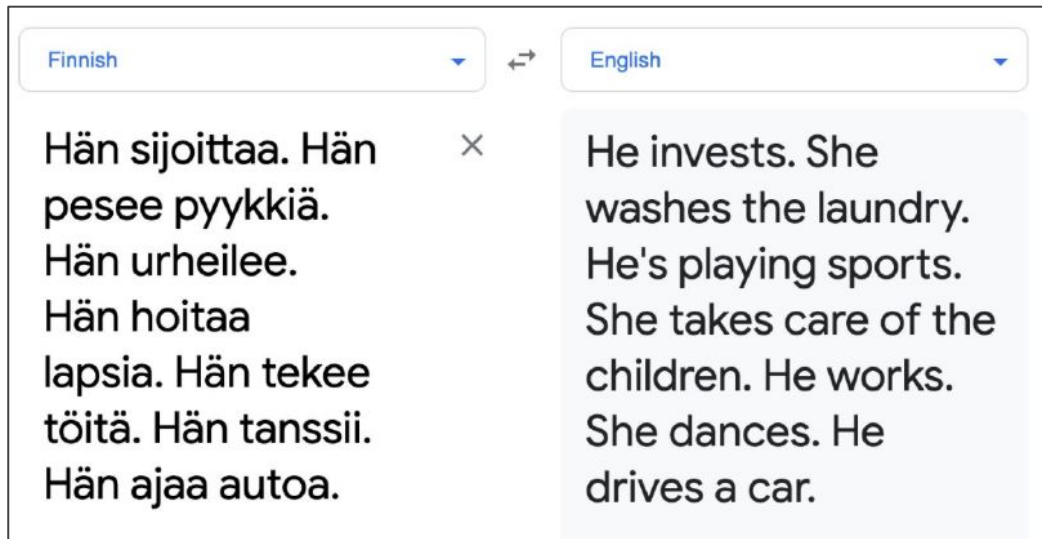
Bias and fairness

Bias and fairness in NLP data / models

- Gender bias
- Racial bias: Tweets written by African Americans are more likely to be flagged as offensive by AI ([see here](#))
- Many ref [here](#)

Biases come from data: our systems encode the stereotypes present in data

→ but it's the responsibility of the algorithm's creator to detect / correct these biases



The screenshot shows a translation interface with two dropdown menus at the top: 'Finnish' on the left and 'English' on the right, with a double-headed arrow between them. Below the 'Finnish' menu is a text input field containing the Finnish sentence: 'Hän sijoittaa. Hän pesee pyykkiä. Hän urheilee. Hän hoitaa lapsia. Hän tekee töitä. Hän tanssii. Hän ajaa autoa.' To the right of this text is a small 'x' icon. Below the 'English' menu is a text output field containing the English translation: 'He invests. She washes the laundry. He's playing sports. She takes care of the children. He works. She dances. He drives a car.'

Risks and ethics

National Eating Disorders Association Phases Out Human Helpline, Pivots to Chatbot

May 31, 2023

(NPR) - For more than 20 years, the National Eating Disorders Association (NEDA) has operated a phone line and online platform for people seeking help with anorexia, bulimia, and other eating disorders. Last year, nearly 70,000 individuals used the helpline. NEDA shuttered that service in May. Instead, the non-profit will use a chatbot called Tessa that was designed by eating disorder experts, with funding from NEDA. ([Read More](#)) <https://bioethics.com/archives/70493>

Risks and ethics

National Eating Disorders Association Phases Out Human Helpline, Pivots to Chatbot

May 31, 2023

(NPR) - For more than 20 years, the National Eating Disorders Association (NEDA) has operated a phone line and online platform for people seeking help with anorexia, bulimia, and other eating disorders. Last year, nearly 70,000 individuals used the helpline. NEDA shuttered that service in May. Instead, the non-profit will use a chatbot called Tessa that was designed by eating disorder experts, with funding from NEDA. ([Read More](#)) <https://bioethics.com/archives/70493>

An AI-powered chatbot that replaced employees at an eating disorder hotline has been shut down after it provided harmful advice to people seeking help.

The saga began earlier this year when the National Eating Disorder Association (NEDA) announced it was shutting down its human-run helpline and replacing workers with a chatbot called "Tessa." That decision came after helpline employees voted to unionize.

AI might be heralded as a way to boost workplace productivity and even make some jobs easier, but Tessa's stint was short-lived. The chatbot ended up providing dubious and even harmful advice to people with eating disorders, such as recommending that they count calories and strive for a deficit of up to 1,000 calories per day, among other "tips," according to critics. <https://www.cbsnews.com/news/eating-disorder-helpline-chatbot-disabled/>

Bias and fairness: IA used in legal domain

Algorithms Were Supposed to Reduce Bias in Criminal Justice—Do They? ([The Brink](#))

Championed as dispassionate, computer-driven calculations about risk, crime, and recidivism, their deployment in everything from policing to bail and sentencing to parole was meant to smooth out what are often unequal decisions made by fallible, biased humans.

- algorithms built upon incomplete or biased data can [replicate or even amplify that bias](#): ProPublica found that one particular system used by courts across the country guessed wrong about two times as often for Black people than for white people (→ "Correctional Offender Management Profiling for Alternative Sanctions": COMPAS system is used in parts of the US to predict whether defendants will commit crime again)
- because these scores *feel* impartial, they can carry a lot of weight with the judges who use them

[Rapport du Conseil de l'Europe](#), [[Barocas and Selbst, 2016](#)]

- AI systems are often "black boxes": opaqueness of (...) decisions [makes it] difficult for people to assess whether they were discriminated against on the basis of, for instance, racial origin.
- AI-driven decision-making can lead to discrimination in several ways. (...) The problems relate to (i) how the "target variable" and the "class labels" are defined; (ii) labelling the training data; (iii) collecting the training data; (iv) feature selection; and (v) proxies. In addition, (vi), AI systems can be used, on purpose, for discriminatory ends

Bias and fairness: IA used in legal domain

Algorithms Were Supposed to Reduce Bias in Criminal Justice—Do They? ([The Brink](#))

Championed as dispassionate, computer-driven calculations about risk, crime, and recidivism, their deployment in everything from policing to bail and sentencing to parole was meant to smooth out what are often unequal decisions made by **fallible, biased humans**.

→ algorithms built upon incomplete or biased data can [replicate or even amplify that bias](#): ProPublica found that one particular system used by courts across the country guessed wrong about two times as often for Black people than for white people (→ "Correctional Offender Management Profiling for Alternative Sanctions": COMPAS system is used in parts of the US to predict whether defendants will commit crime again)

→ because these scores *feel* impartial, they can carry a lot of weight with the judges who use them

[Rapport du Conseil de l'Europe](#), [[Barocas and Selbst, 2016](#)]

→ AI systems are often "black boxes": opaqueness of (...) decisions [makes it] difficult for people to assess whether they were discriminated against on the basis of, for instance, racial origin.

→ AI-driven decision-making can lead to discrimination (...) The problems relate to (i) how the "target variable" and the "class labels" are defined; (ii) labelling the training data; (iii) collecting the training data; (iv) feature selection; and (v) proxies. In addition, (vi), AI systems can be used, on purpose, for discriminatory ends

Bias and fairness: IA used in legal domain

Algorithms Were Supposed to Reduce Bias in Criminal Justice—Do They? ([The Brink](#))

Championed as dispassionate, computer-driven calculations about risk, crime, and recidivism, their deployment in everything from policing to bail and sentencing to parole was meant to smooth out what are often unequal decisions made by **fallible, biased humans**.

→ algorithms built upon **incomplete or biased data** can [replicate or even amplify that bias](#): ProPublica found that one particular system used by courts across the country guessed wrong about two times as often for Black people than for white people (→ "Correctional Offender Management Profiling for Alternative Sanctions": COMPAS system is used in parts of the US to predict whether defendants will commit crime again)

→ because these scores *feel* impartial, they can carry a lot of weight with the judges who use them

[Rapport du Conseil de l'Europe](#), [[Barocas and Selbst, 2016](#)]

→ AI systems are often "black boxes": opaqueness of (...) decisions [makes it] difficult for people to assess whether they were discriminated against on the basis of, for instance, racial origin.

→ AI-driven decision-making can **lead to discrimination** (...) The problems relate to (i) how the **"target variable" and the "class labels"** are defined; (ii) **labelling the training data**; (iii) **collecting the training data**; (iv) **feature selection**; and (v) proxies. In addition, (vi), AI systems can be used, **on purpose**, for discriminatory ends

Bias and fairness: IA used in legal domain

Algorithms Were Supposed to Reduce Bias in Criminal Justice—Do They? ([The Brink](#))

Championed as dispassionate, computer-driven calculations about risk, crime, and recidivism, their deployment in everything from policing to bail and sentencing to parole was meant to smooth out what are often unequal decisions made by fallible, biased humans.

→ algorithms built upon incomplete or biased data can [replicate or even amplify that bias](#): ProPublica found that one particular system used by courts across the country guessed wrong about two times as often for Black people than for white people (→ "Correctional Offender Management Profiling for Alternative Sanctions": COMPAS system is used in parts of the US to predict whether defendants will commit crime again)

→ **because these scores *feel* impartial, they can carry a lot of weight with the judges who use them**

[Rapport du Conseil de l'Europe](#), [[Barocas and Selbst, 2016](#)]

→ AI systems are often "black boxes": opaqueness of (...) decisions [makes it] difficult for people to assess whether they were discriminated against on the basis of, for instance, racial origin.

→ AI-driven decision-making can lead to discrimination in several ways. (...) The problems relate to (i) how the "target variable" and the "class labels" are defined; (ii) labelling the training data; (iii) collecting the training data; (iv) feature selection; and (v) proxies. In addition, (vi), AI systems can be used, on purpose, for discriminatory ends

Bias and fairness: IA used in legal domain

Algorithms Were Supposed to Reduce Bias in Criminal Justice—Do They? ([The Brink](#))

Championed as dispassionate, computer-driven calculations about risk, crime, and recidivism, their deployment in everything from policing to bail and sentencing to parole was meant to smooth out what are often unequal decisions made by fallible, biased humans.

- algorithms built upon incomplete or biased data can [replicate or even amplify that bias](#): ProPublica found that one particular system used by courts across the country guessed wrong about two times as often for Black people than for white people (→ "Correctional Offender Management Profiling for Alternative Sanctions": COMPAS system is used in parts of the US to predict whether defendants will commit crime again)
- because these scores *feel* impartial, they can carry a lot of weight with the judges who use them

[Rapport du Conseil de l'Europe](#), [[Barocas and Selbst, 2016](#)]

- **AI systems are often "black boxes"**: opaqueness of (...) decisions [makes it] difficult for people to assess whether they were discriminated against on the basis of, for instance, racial origin.
- AI-driven decision-making can lead to discrimination in several ways. (...) The problems relate to (i) how the "target variable" and the "class labels" are defined; (ii) labelling the training data; (iii) collecting the training data; (iv) feature selection; and (v) proxies. In addition, (vi), AI systems can be used, on purpose, for discriminatory ends

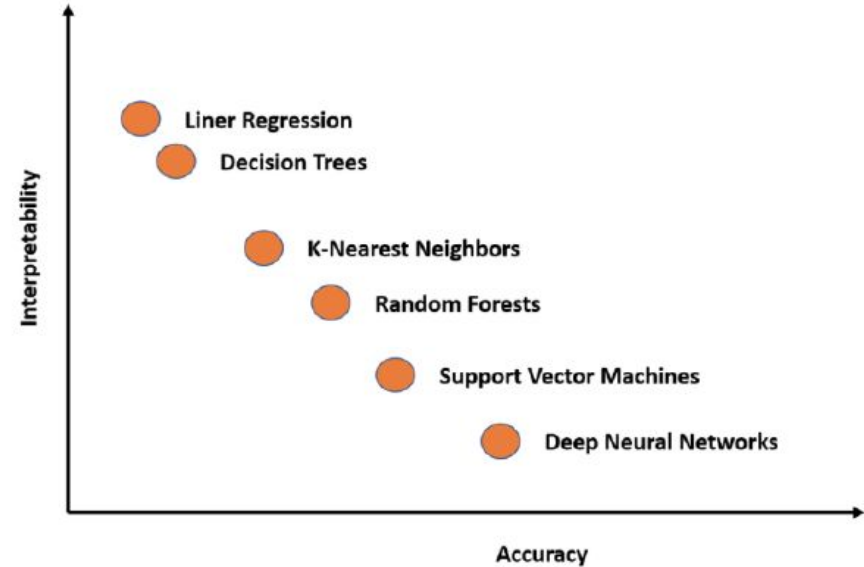
Interpretability

To accept automatic systems:

- we need to be able to justify / understand / interpret their decisions
- but neural models are hard to analyze

→ very important research area

→ crucial for 'critical' domains: medicine, legal domain, some technical domains (e.g. airplanes) ...



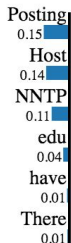
LIME

Explaining an example based on its inputs: LIME [Ribeiro et al 2017] <https://github.com/marcotcr/lime>

Prediction probabilities



atheism



christian

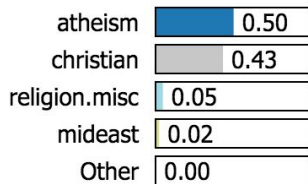
Text with highlighted words

From: johnchad@triton.unm.edu (jchadwic)
Subject: Another request for Darwin Fish
Organization: University of New Mexico, Albuquerque
Lines: 11
NNTP-Posting-Host: triton.unm.edu

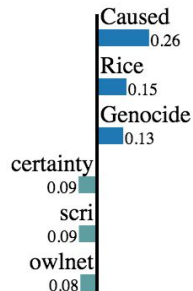
Hello Gang,

There have been some notes recently asking where to obtain the DARWIN fish.
This is the same question I have and I have not seen an answer on the net. If anyone has a contact please post on the net or email me.

Prediction probabilities

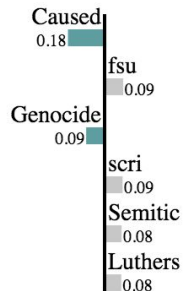


NOT atheism



atheism

NOT christian



christian

Saliency map

Explaining an example based on its inputs

- Saliency Map / Allen Interpret (Wallace et al., 2019) <https://allennlp.org/interpre>

Simple Gradients Visualization

See saliency map interpretations generated by [visualizing the gradient](#).

Saliency Map:

[CLS] The [MASK] rushed to the emergency room to see her patient . [SEP]

Mask 1 Predictions:

47.1% nurse

16.4% woman

10.0% doctor

3.4% mother

3.0% girl

Saliency Map / Allen Interpret (Wallace et al., 2019)

Environmental and financial cost

- training of BERT – Large 16 Cloud TPUs (64 TPU chips total) → 16 (devices) * 4 (days) * 24 (hours) * 4.5 (US\$ per hour) = US\$6,912
- GPT-2 model used 256 Google Cloud TPU v3 cores → costs \$256 per hour
- XLNet: about \$245,000... ([estimated costs, no communication](#))
- [How to shrink AI's ballooning carbon footprint](#)

Year	Model	# of Parameters	Dataset Size
2019	BERT [39]	3.4E+08	16GB
2019	DistilBERT [113]	6.60E+07	16GB
2019	ALBERT [70]	2.23E+08	16GB
2019	XLNet (Large) [150]	3.40E+08	126GB
2020	ERNIE-GEN (Large) [145]	3.40E+08	16GB
2019	RoBERTa (Large) [74]	3.55E+08	161GB
2019	MegatronLM [122]	8.30E+09	174GB
2020	T5-11B [107]	1.10E+10	745GB
2020	T-NLG [112]	1.70E+10	174GB
2020	GPT-3 [25]	1.75E+11	570GB
2020	GShard [73]	6.00E+11	–
2021	Switch-C [43]	1.57E+12	745GB

Table 1: Overview of recent large language models

Air travel, 1 person, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experiments	78,468
Transformer (big)	192
w/ neural arch. search	626,155

NLP: an empirical field with high societal impact

Empirical field:

- Models are built upon data 'observed' or 'created'
 - representativeness?
 - label annotation is necessary → cost, representativeness, quality, ethical issues
- Evaluation: metrics are often debatable and qualitative analysis is often missing
- Language diversity = no 'universal' approach: most work for majority / dominating languages

High societal impact:

- carbon footprint
- ethical stakes
 - excluding part of the population (low-resourced languages, minorities ...)
 - increasing societal biases
 - malicious use: surveillance, personal data, advertising, political targeting...
 - The Social Impact of Natural Language Processing <https://aclanthology.org/P16-2096.pdf>
 - Cartography of Natural Language Processing for Social Good <https://aclanthology.org/2021.nlp4posimpact-1.3.pdf>

Progress in NLP

Fast evolving field!

- increasing number of articles, researchers, conferences are growing
- many computer scientists but a need for interdisciplinarity, linguistics, sociology, and experts of targeted domains

Find papers:

- ACL anthology
- Track NLP progress, nice initiative, e.g. sentiment analysis on IMDB
<https://nlpprogress.com/>

Model	Accuracy	Paper / Source
XLNet (Yang et al., 2019)	96.21	XLNet: Generalized Autoregressive Pretraining for Language Understanding
BERT_large+ITPT (Sun et al., 2019)	95.79	How to Fine-Tune BERT for Text Classification?
BERT_base+ITPT (Sun et al., 2019)	95.63	How to Fine-Tune BERT for Text Classification?
ULMFiT (Howard and Ruder, 2018)	95.4	Universal Language Model Fine-tuning for Text Classification
Block-sparse LSTM (Gray et al., 2017)	94.99	GPU Kernels for Block-Sparse Weights
oh-LSTM (Johnson and Zhang, 2016)	94.1	Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings
Virtual adversarial training (Miyato et al., 2016)	94.1	Adversarial Training Methods for Semi-Supervised Text Classification
BCN+Char+CoVe (McCann et al., 2017)	91.8	Learned in Translation: Contextualized Word Vectors



Practical session

1. Spacy: pre-processing and analysing text data
2. Biases in Languages Models
3. Classification using HuggingFace library
4. Explainability

→ upload the Python notebook and make a copy:

<https://colab.research.google.com/drive/1veOjpSz5EDYLSpmnUQaxi662Ki2UbOyW?usp=sharing>



Machine Learning: Summary (and missing details)

In this course we talked about:

- learning functions / architectures
 - linear functions
 - neural architectures combining linear and non linear functions
 - specific neural architectures : RNN, Transformers
- textual data representations
 - one-hot encoding
 - distributional representation: static and contextual embeddings

Some details on machine learning settings are missing.

Machine Learning: Summary (and missing details)

1. What does the function f looks like ?
 - a. we need to assign a score to all possibles outputs
 - b. and decide the predicted output based on these scores
2. How to learn the function f ?
 - a. we see examples of pairs of input, output
 - b. build a (first) approximation of f
 - c. compare its prediction with the gold truth
 - d. modify f if the models is wrong
3. How to evaluate our function?
 - a. perform predictions on an unseen dataset
 - b. compute some performance metrics
 - c. compare to baselines, other architectures and state-of-the-art

Machine Learning: Summary (and missing details)

1. What does the function f look like? → linear / neural archi.
 - a. we need to assign a score to all possible outputs
 - b. and decide the predicted output based on these scores
2. How to learn the function f ?
 - a. we see examples of pairs of input, output
 - b. build a (first) approximation of f
 - c. compare its prediction with the gold truth
 - d. modify f if the model is wrong
3. How to evaluate our function?
 - a. perform predictions on an unseen dataset
 - b. compute some performance metrics
 - c. compare to baselines, other architectures and state-of-the-art

Machine Learning: Summary (and missing details)

1. What does the function f look like? → linear / neural archi.
 - a. we need to assign a score to all possible outputs
 - b. and decide the predicted output based on these scores
2. How to learn the function f ?
 - a. we see examples of pairs of input, output
 - b. build a (first) approximation of f
 - c. compare its prediction with the gold truth
 - d. modify f if the model is wrong
3. How to evaluate our function?
 - a. perform predictions on an unseen dataset
 - b. compute some performance metrics
 - c. compare to baselines, other architectures and state-of-the-art

Machine Learning: Summary (and missing details)

1. What does the function f look like? → linear / neural archi.
 - a. we need to assign a score to all possible outputs
 - b. and decide the predicted output based on these scores
2. How to learn the function f ? = training + tuning / optimizing
 - a. we see examples of pairs of input, output
 - b. build a (first) approximation of f
 - c. compare its prediction with the gold truth
 - d. modify f if the model is wrong
3. How to evaluate our function?
 - a. perform predictions on an unseen dataset
 - b. compute some performance metrics
 - c. compare to baselines, other architectures and state-of-the-art

Machine Learning: Summary (and missing details)

1. What does the function f look like? → linear / neural archi.
 - a. we need to assign a score to all possible outputs
 - b. and decide the predicted output based on these scores
2. How to learn the function f ? = training + tuning / optimizing
 - a. we see examples of pairs of input, output
 - b. build a (first) approximation of f
 - c. compare its prediction with the gold truth
 - d. modify f if the model is wrong
3. How to evaluate our function?
 - a. perform predictions on an unseen dataset
 - b. compute some performance metrics, e.g. accuracy = fraction of correctly predicted samples
 - c. compare to baselines, other architectures and state-of-the-art

The different tasks

- **Classification:** predict a categorical label for each item
 - single label: each instance is assigned a single label
 - binary: 2 labels, e.g. an email is either a spam or not
 - multi-class: > 2 labels, e.g. sentiment is either positive, negative or neutral
 - multi-label: each instance is assigned multiple labels, e.g. *The Lord of the Ring* is classified as: Adventure, Fantasy, Drama
- **Sequence labeling / structured prediction:** predict a categorical label for each member of a sequence
 - e.g. POS tagging, NER...
 - can be seen as performing independent classification tasks on each item
 - but performance are improved when taking into account the dependence between the elements
- **Regression:** Predict a **real value** for each item
 - e.g.: prediction of stock values, variations of economic variables, house prices..
 - rarer for NLP, but e.g. data with depression "scores" (DAIC)
- **Clustering:** group (similar) data without knowing the classes, e.g. social media relations

The learning scenarios

Depend on the **annotations** you have:

Supervised learning:

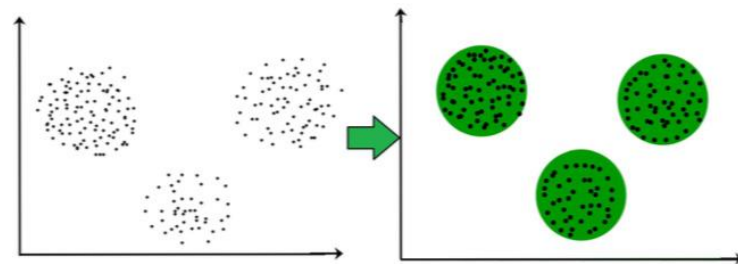
- we have a set of **labeled** examples as training data
- most common for classification and regression

Unsupervised learning:

- we only have **unlabeled** training data
 - e.g.: Clustering and dimensionality reduction
 - often hard to evaluate

Semi-supervised learning: in-between

- use unlabeled data and annotated data together, and propagate labels
- use knowledge, e.g. from the domain to automatically annotate (noisy) data :) = happy ;:(= sad



Classification with Scikit-Learn

```
▶ from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

vectorizer = CountVectorizer( )
train_data_features = vectorizer.fit_transform( train )
test_data_features = vectorizer.transform( test )

classifier = LogisticRegression(max_iter=200)
classifier.fit( train_data_features, train_labels )

preds = classifier.predict( test_data_features )

print( classification_report( test_labels, preds ) )
```

Vectorize



Classification with Scikit-Learn

```
▶ from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

vectorizer = CountVectorizer( )
train_data_features = vectorizer.fit_transform( train )
test_data_features = vectorizer.transform( test )

classifier = LogisticRegression(max_iter=200)
classifier.fit( train_data_features, train_labels )

preds = classifier.predict( test_data_features )

print( classification_report( test_labels, preds ) )
```

Vectorize

Train

Classification with Scikit-Learn

```
▶ from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

vectorizer = CountVectorizer( )
train_data_features = vectorizer.fit_transform( train )
test_data_features = vectorizer.transform( test )

classifier = LogisticRegression(max_iter=200)
classifier.fit( train_data_features, train_labels )

preds = classifier.predict( test_data_features )

print( classification_report( test_labels, preds ) )
```

Vectorize

Train

Predict

Classification with Scikit-Learn

```
▶ from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

vectorizer = CountVectorizer( )
train_data_features = vectorizer.fit_transform( train )
test_data_features = vectorizer.transform( test )

classifier = LogisticRegression(max_iter=200)
classifier.fit( train_data_features, train_labels )

preds = classifier.predict( test_data_features )

print( classification_report( test_labels, preds ) )
```

Vectorize

Train

Predict

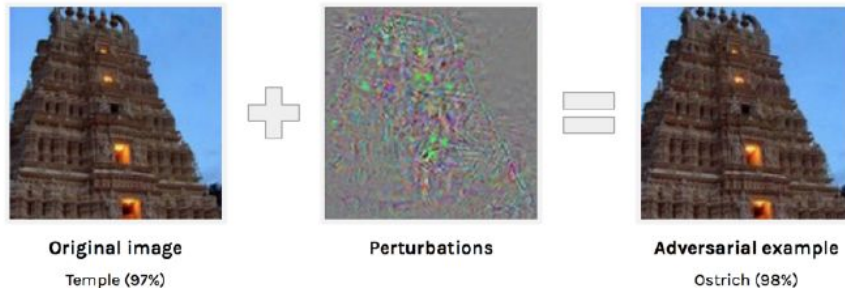
Evaluate

Robustness

Adversarial examples

- Find examples correctly predicted by the system
- try to generate similar examples where the system makes a mistake

→ test system robustness (or to hack it)



Adversarial examples

Original

Perfect performance by the actor → **Positive (99%)**

Adversarial

Spotless performance by the actor → **Negative (100%)**

TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP
<https://www.aclweb.org/anthology/2020.emnlp-demos.16.pdf>

Original Input	Connoisseurs of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: <u>Positive (77%)</u>
Adversarial example [Visually similar]	<u>Aonnoisseurs</u> of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: <u>Negative (52%)</u>
Adversarial example [Semantically similar]	Connoisseurs of Chinese <u>footage</u> will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: <u>Negative (54%)</u>

NLP courses, slides, interesting websites (and sources)

- https://dair.ai/notebooks/nlp/2020/03/19/nlp_basics_tokenization_segmentation.html
 - <https://www.slideshare.net/dinel/new-trends-in-nlp-applications>
 - <https://www.infoq.com/presentations/nlp-practitioners/>
 - <https://github.com/sebastianruder/NLP-progress>
- + Thanks to: Tim Van Der Cruys and Philippe Muller who shared their materials

Other sources

- Morphology:
 - <https://www.slideserve.com/flynn/natural-language-processing-morphology>
 - <https://theweek.com/articles/463500/8-favorite-ridiculously-long-german-words>
- Tokenisation:
 - <https://www.thoughtvector.io/blog/subword-tokenization/>
 - <http://tm-town-nlp-resources.s3.amazonaws.com/ch2.pdf>
 - <https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4>
 - <https://pythonprogramming.net/tokenizing-words-sentences-nltk-tutorial/>
 - <https://blog.ekbana.com/pre-processing-text-in-python-ad13ea544dae>
- Sentence segmentation:
 - <https://www.tm-town.com/blog/is-segmentation-solved-problem>
- NER:
 - http://nlpprogress.com/english/named_entity_recognition.html
- Spacy:
 - <https://pypi.org/project/visualise-spacy-tree/>
 - <https://towardsdatascience.com/getting-to-grips-with-parse-trees-6e19e7cd3c3c>
- Vectorization:
 - <https://towardsdatascience.com/representing-text-in-natural-language-processing-1eead30e57d8>
 - <https://towardsdatascience.com/the-magic-behind-embedding-models-part-1-974d539f21fd>
- NLP challenges:
 - <https://towardsdatascience.com/bias-in-natural-language-processing-nlp-a-dangerous-but-fixable-problem-7d01a12cf0f7>
 - <https://www.aclweb.org/anthology/P19-1334/>
 - <https://www.aclweb.org/anthology/S18-2023/>
 - http://web.cs.ucla.edu/~kwchang/publications_area/#FEAT
- Sentiment Analysis:
 - Bhatia, P., Ji, Y., & Eisenstein, J. (2015, September). Better Document-level Sentiment Analysis from RST Discourse Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 2212-2218).
- <https://iq.opengenus.org/advanced-nlp-models/>

Sources of pictures

- <https://news.sky.com/story/talking-robots-could-be-used-in-uk-care-homes-to-ease-loneliness-and-improve-mental-health-12066295>
- <https://blog.lecko.fr/machine-learning/ordinateur-a-t-il-vraiment-lesoin-de-nous/>
- https://fr.m.wikipedia.org/wiki/Fichier:Major_levels_of_linguistic_structure.svg
- <https://realpython.com/sentiment-analysis-python/>
- <https://realpython.com/natural-language-processing-spacy-python/>
- <https://www.dw.com/en/linguist-theres-a-difference-between-learning-words-and-learning-a-language/a-18602460>
- <https://numberdyslexia.com/100-orton-gillingham-red-words-list/>
- <https://sites.google.com/a/sheffield.ac.uk/aal2013/branches/syntax/why-is-syntax-studied>
- <https://www.bangkokpost.com/life/social-and-lifestyle/1757224/word-wise>
- <https://blog.aaroncwong.com/2019/building-a-bigram-hidden-markov-model-for-part-of-speech-tagging/>
- <https://diyagodayal.github.io/2018/06/08/An-introduction-to-part-of-speech-tagging-and-the-Hidden-Markov-Model-953d45538f24/>
- <https://www.youtube.com/watch?v=CeuhQ3s-Iss>
- <https://analytic.sindilamag.com/10-most-used-databases-by-developers-in-2020/>
- <https://www.erdlil.fr/blog/2017/09/26/traitement-automatique-langues-erdlil/>
- <https://www.thoughtco.com/ambiguity-language-1692388>
- <http://katbailev.github.io/ml/#/title>
- <https://towardsdatascience.com/representing-text-in-natural-language-processing-1eead30e57d8>
- <https://towardsdatascience.com/representing-text-in-natural-language-processing-1eead30e57d8>
- <https://www.wolfram.com/language/12/natural-language-processing/>
- <https://medium.com/swlh/text-cleaning-43fe4062952b>
- <https://twitter.com/yoavo/status/1318576449985662977/photo/1>
- <http://jetpaper.web.fc2.com/reviews/page-5462805.html>
- <https://www.sjl.com/7detailStory=can-we-talk-librarians-lead-new-push-civics-education-focusing-discourse>
- <https://www.toptal.com/deep-learning/4-sentiment-analysis-accuracy-traps>
- <https://nativeenglishshain.blogspot.com/2013/11/16/wise-wednesday-grammar-homonym.html>
- <https://www.altcourse.com/training/insearce/lexicogram.html>
- <https://medium.com/@umaguntur789/everything-you-need-to-know-about-named-entity-recognition-2a136f38c08f>
- <https://activevizards.com/blog/top-nlp-algorithms-and-concepts/>
- <https://medium.com/@laura.mondoloni/ironback-sketch-exercise-kata-rocket-9ee72879d0d9>
- <https://www.youtube.com/watch?v=hTfghKdHhttps://www.thepoke.co.uk/2014/05/21/the-25-worst-best-spelling-mistakes-on-twitter/>
- <http://www.igfasouza.com/blog/check-tweets-spelling/>
- <https://www.sergent-tobogo.com/dessins/marvin-droopy/>
- <https://www.relationclientmag.fr/Thematique/strategies-1255/Breves/chatbot-oui-sncf-devient-premier-gagnant-Best-Robot-Experience-380102.htm>
- <https://developers.google.com/machine-learning/guides/text-classification>
- <https://medium.com/mosaix/deep-text-representation-for-sequence-labeling-2f2e605ed9d>
- <https://medium.com/swlh/we-need-to-talk-about-sentiment-analysis-9d1f20f2ebfb>
- https://medium.com/@pierre_guillou/nlp-fastai-sequence-to-sequence-model-seq2seq-38d9984cf271
- <http://www.bigdatalab.ac.cn/~junix/publications/KG4R2017-keynote.pdf>
- <https://datascientest.com/nlp-word-embedding-word2vec>
- <https://www.datacamp.com/community/tutorials/simplifying-sentiment-analysis-python>
- <https://medium.com/@suneelpatel.in/nlp-guide-101-nlp-evolution-past-present-and-future-fcc573629da3>
- <https://itsully.github.io/perceptron/>
- arabic tokenisation https://www.researchgate.net/publication/267096934_Arabic_Compound_Nouns_processing_inflection_and_tokenization
- <https://navigate360.com/blog-news/what-is-natural-language-processing/>
- <https://www.babelstreet.com/blog/how-named-entity-recognition-connects-the-dots-for-law-enforcement-and-intelligence>
- https://medium.com/@techforce_global/opportunities-and-challenges-of-ai-in-the-job-market-91d661643ca4
- <https://blogs.fainsttute.org/investor/2023/05/26/chatgpt-and-large-language-models-six-evolutionary-steps/>