

# Lecture 6 - Introduction to Natural Language Processing

## Intro2AI-class - ANITI

Chloé Braud, CNRS - IRIT, [chloe.braud@irit.fr](mailto:chloe.braud@irit.fr)

12 April 2021

# Unfortunately, still no HAL or Marvin



*2001, A Space Odyssey*



*The Hitchhiker's Guide to the Galaxy*

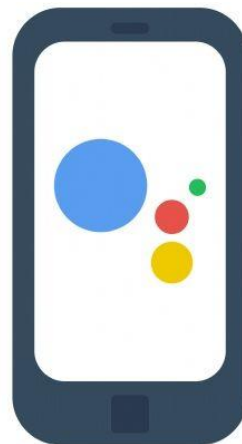
# Unfortunately, still no HAL or Marvin



2001, A Space Odyssey



The Hitchhiker's Guide to the Galaxy



Parler avec OUI.sncf



Bien sur. Voici OUI.sncf.



Bonjour, je suis l'assistant OUI.sncf. Dis-moi où et quand tu veux partir, je m'occupe du reste.



But we can "talk" with the SNCF...

# NLP is used for many applications: mining 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**

# What is Natural Language Processing?

At the **interface between:**

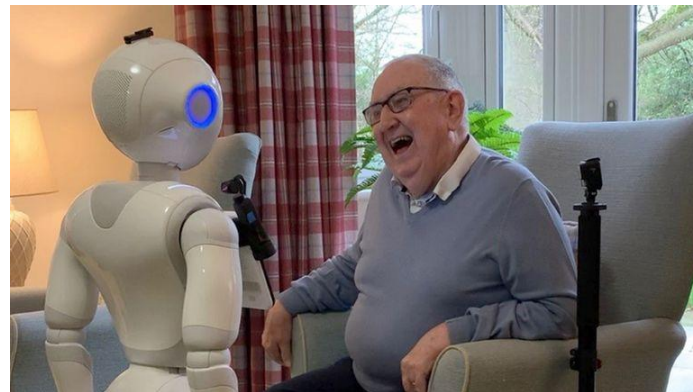
- Linguistics: dealing with natural languages
- Computational Science: algorithms
- Mathematics: statistics, probabilities
- Physics: speech processing

**Subdomain of Artificial Intelligence:**

- human machine interaction
- human human interaction

= access to information / extract information

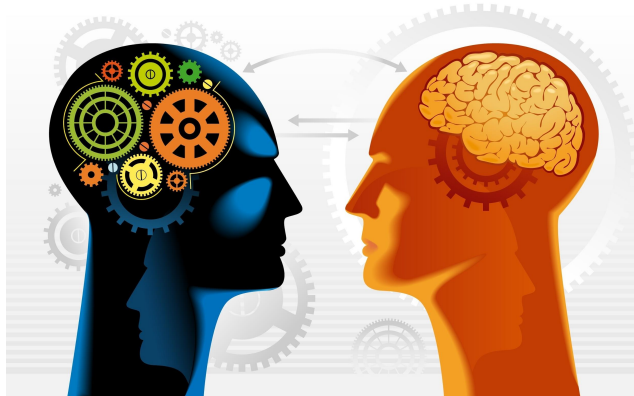
= understanding intelligence through communication



# 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?

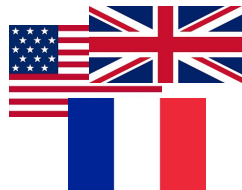


# What is Natural Language Processing?

Take data



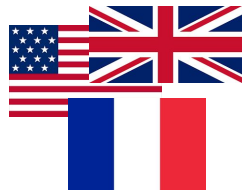
WIKIPÉDIA  
L'encyclopédie libre



# What is Natural Language Processing?

Take data (all)

PubMed



Doctissimo

WIKIPÉDIA  
L'encyclopédie libre





# What is Natural Language Processing?

Take data (all)

PubMed



Doctissimo

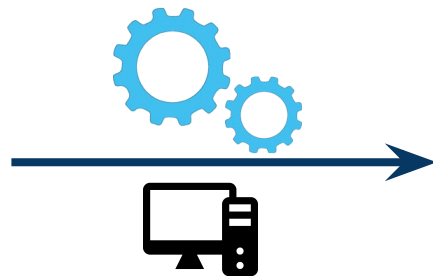
WIKIPÉDIA  
L'encyclopédie libre



# What is Natural Language Processing?

Take data (all)

Process



Doctissimo WIKIPÉDIA  
L'encyclopédie libre

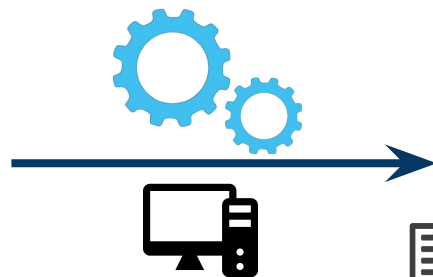


# What is Natural Language Processing?

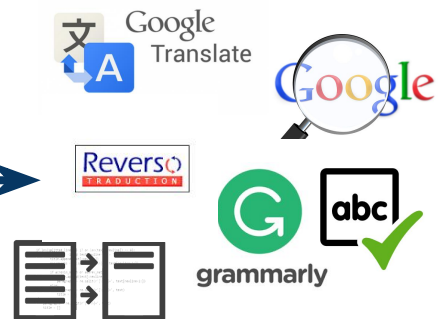
Take data (all)



Process



Develop applications

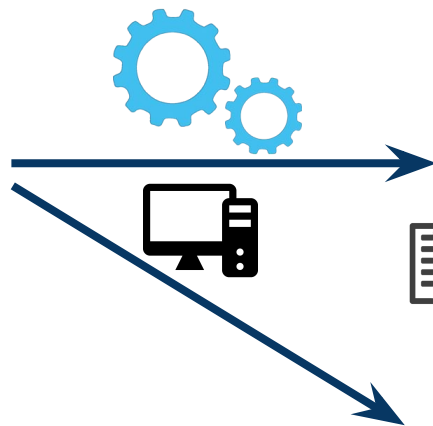


# What is Natural Language Processing?

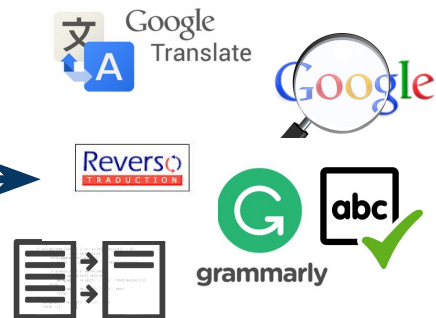
Take data (all)



Process



Develop applications



Understand language

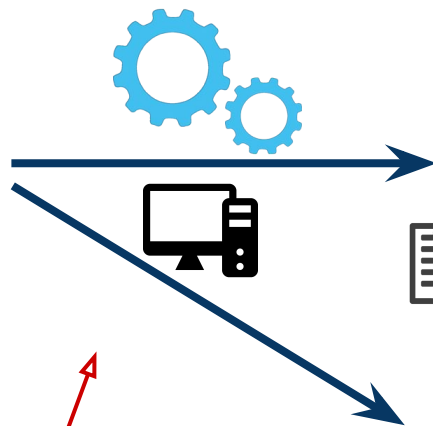


# What is Natural Language Processing?

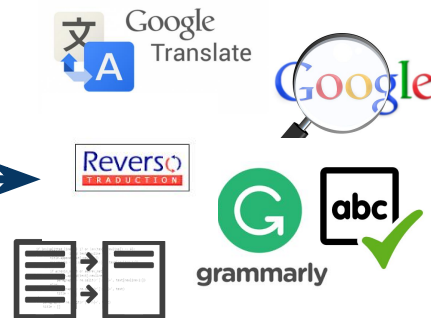
Take data (all)



Process



Develop applications



Understand language



**What does it mean?  
How we do that?**

# What is Natural Language Processing?

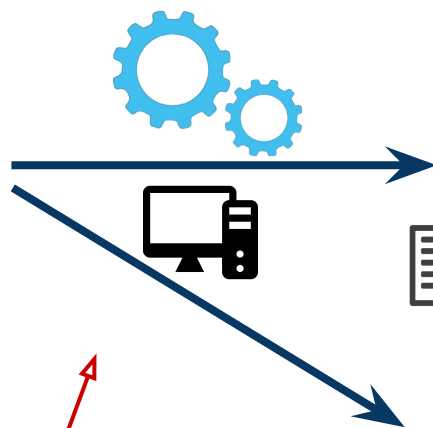
Take data (all)



Many challenges!

What does it mean?  
How we do that?

Process



Develop applications



Understand language



# What is Natural Language Processing?

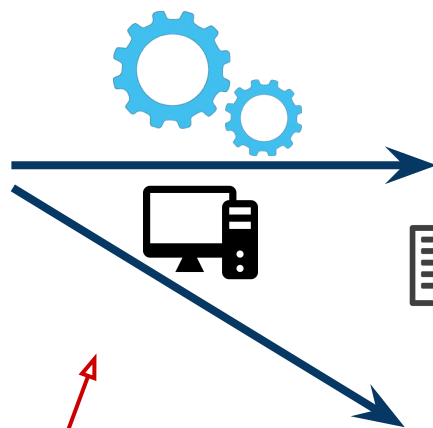
Take data (all)



Many challenges!

What does it mean?  
How we do that?

Process



Develop **applications**



Understand language



# Why NLP is hard?

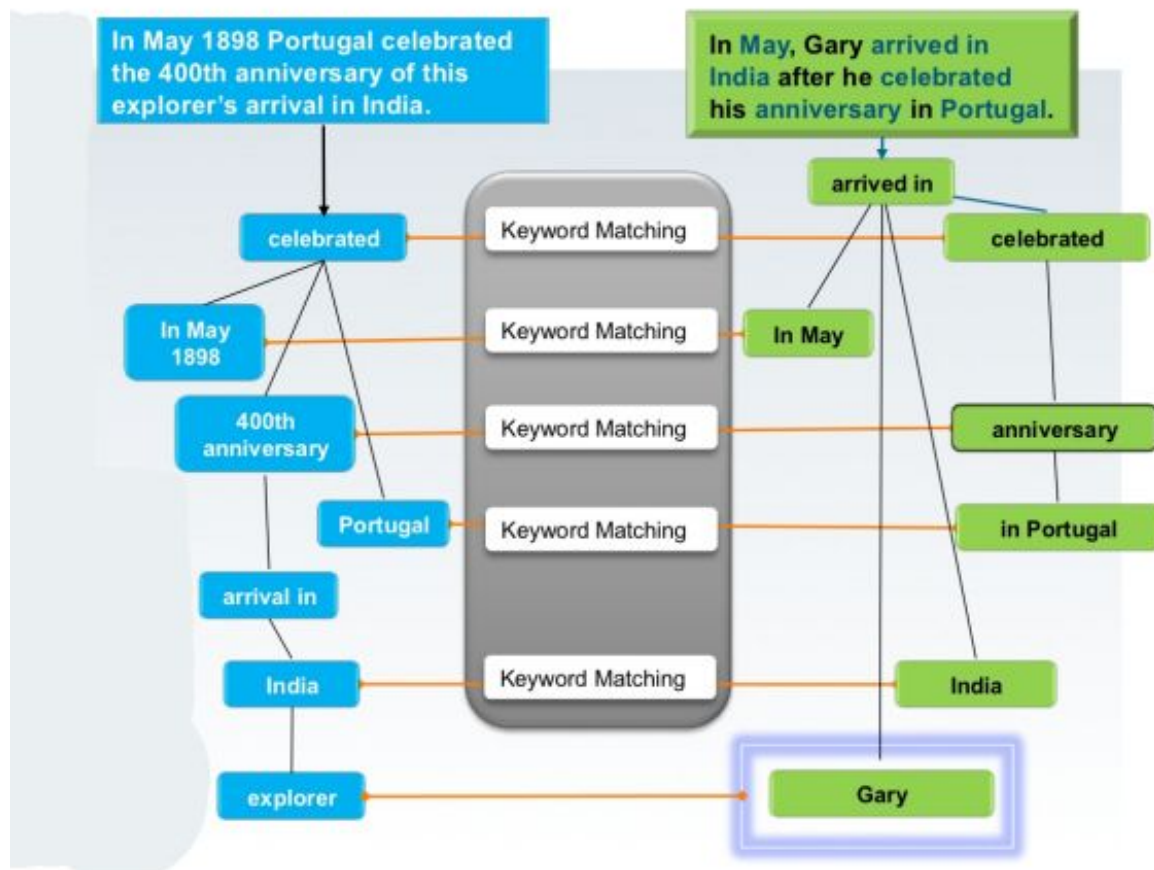
## Example: Question-Answering with Watson





# Why NLP is hard?

Keyword matching can provide good clues  
→ but here, misleading

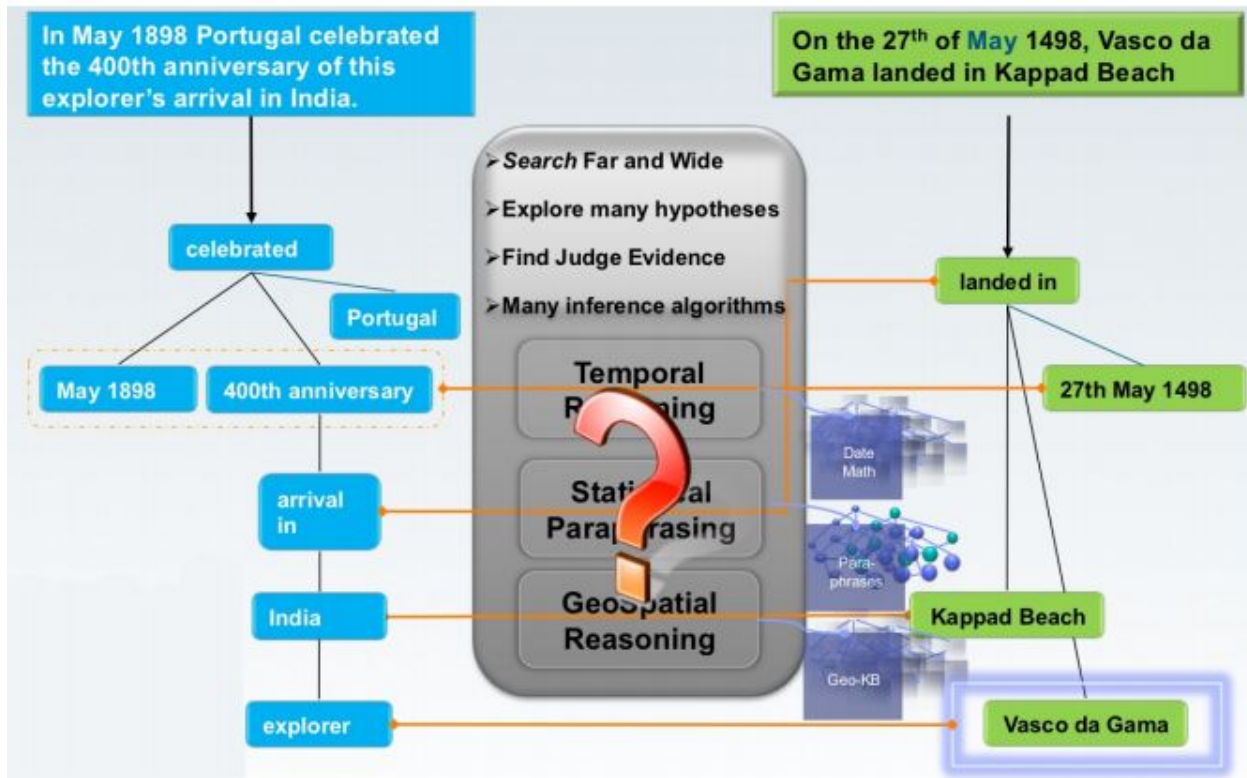


# Why NLP is hard?

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)

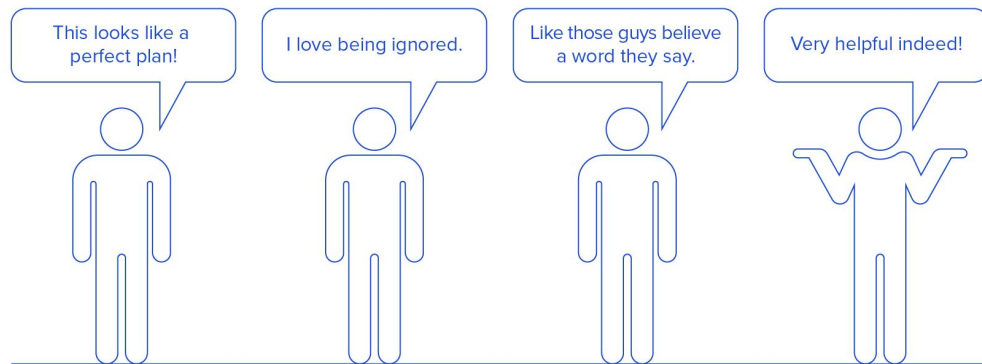


# Why NLP is hard?

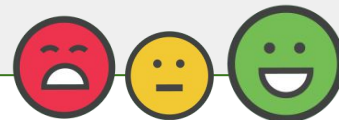
Many issues:

- 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.



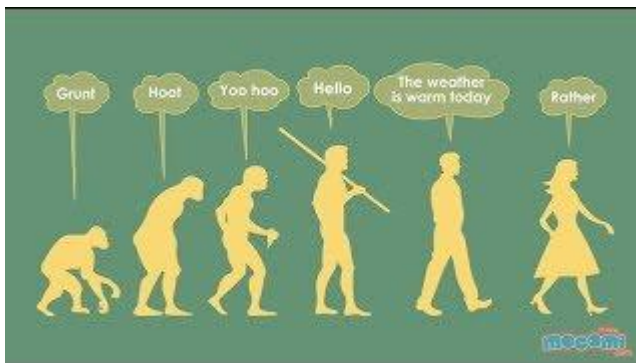
This mouse **is not good** looking, but it works **perfect** and I **like** it.



# 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



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

## 1. Fundamentals of linguistics

- linguistic levels of analysis
- ambiguity!

## 2. Fundamentals of NLP

- Pre-processing
- Linguistic analysis

## 3. Hands-on NLP

- Finding data
- Preparing data
- Learning from data

## 4. More about learning scenario and applications

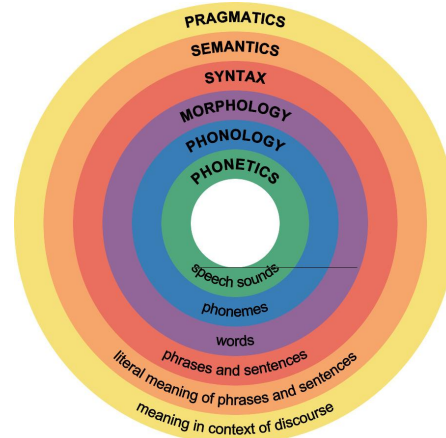
- Classification
- Sequence labelling
- Sequence to Sequence

## 5. To go further

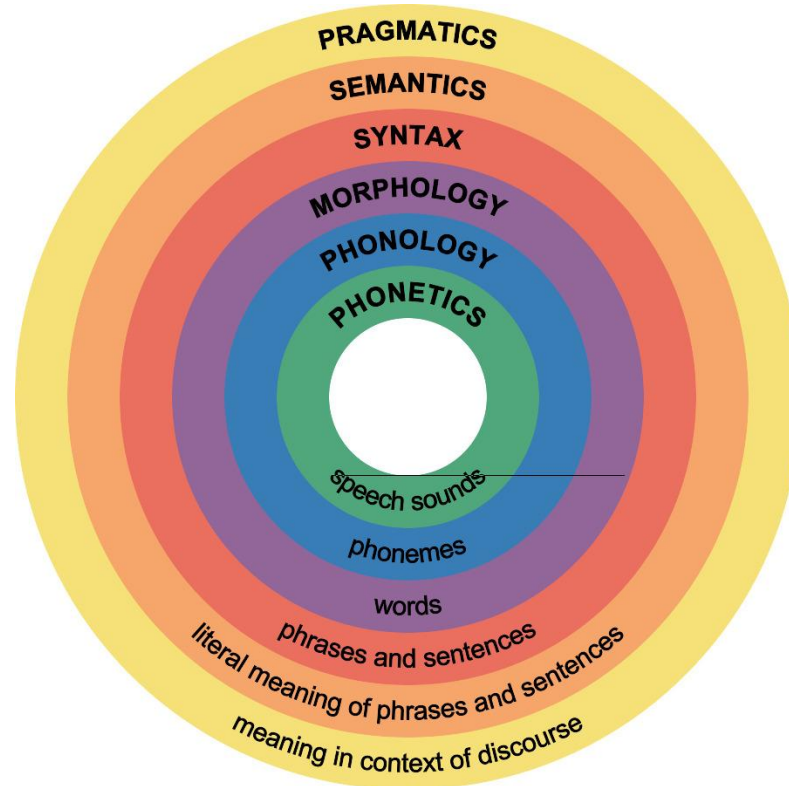
- Limits of classical statistical methods
- Limits of supervised learning
- Current challenges in NLP

**Practical session:** Sentiment analysis

# Fundamentals of Linguistics



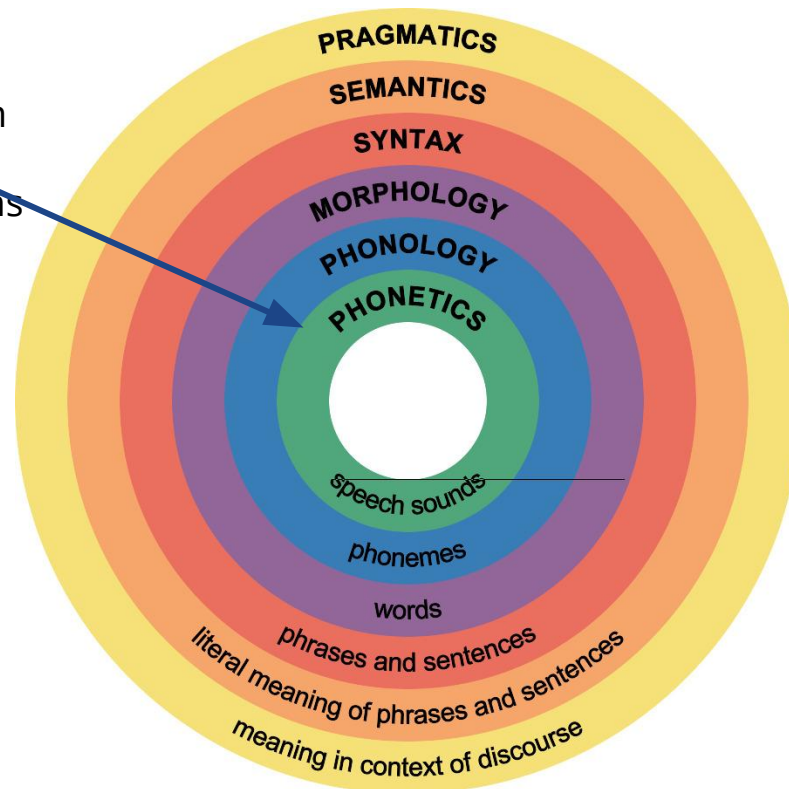
# Main levels of linguistics structure



# Main levels of linguistics structure

## Phonetics-Phonology:

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





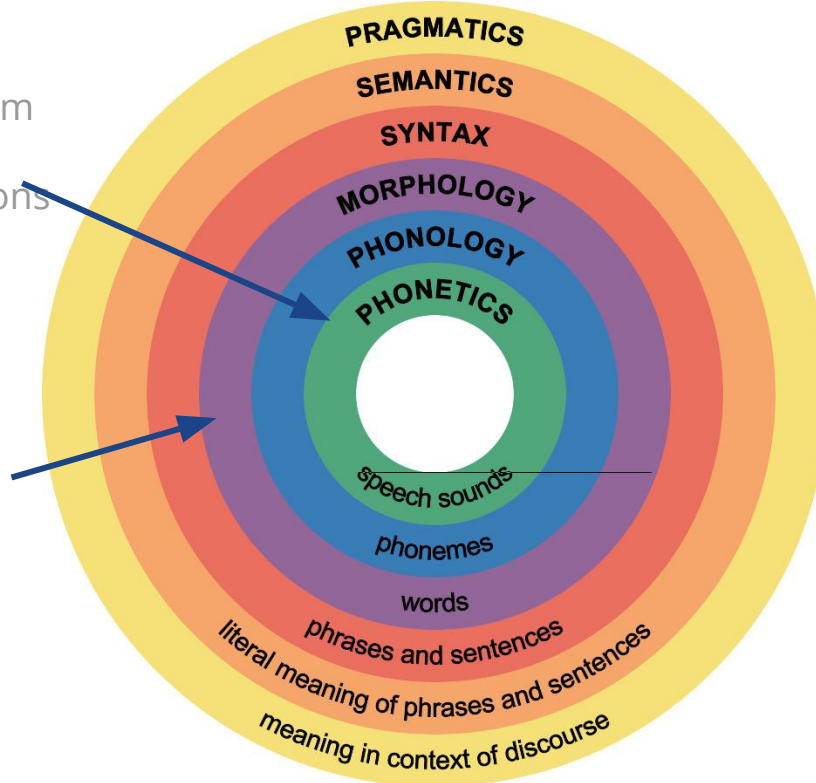
# Main levels of linguistics structure

## 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:

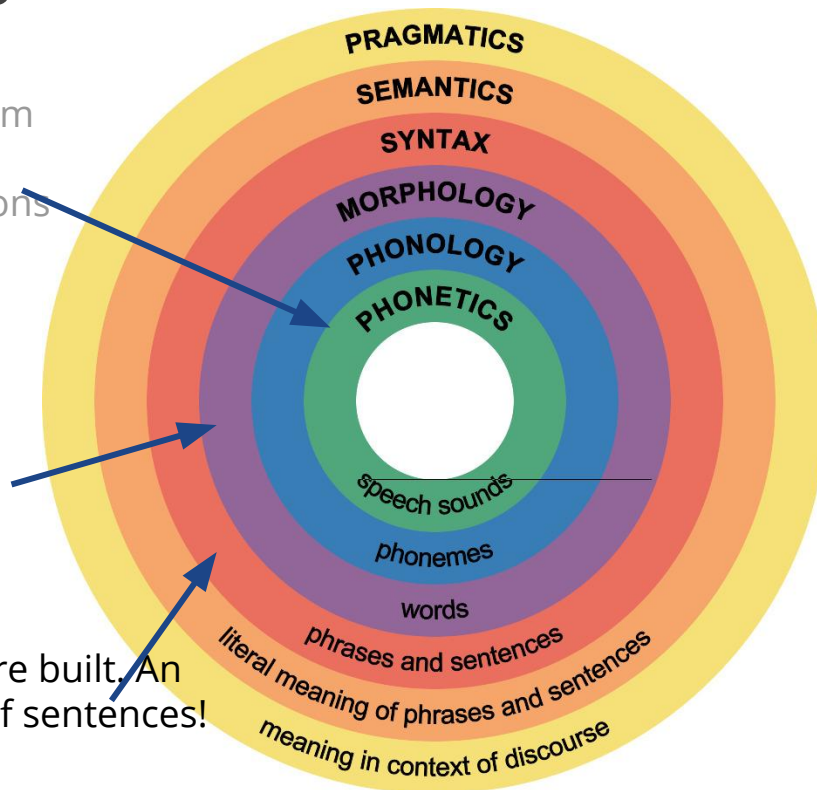
Speech processing is distinct from NLP = focus on text. But NLP researchers work on transcriptions from speech.

## 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:

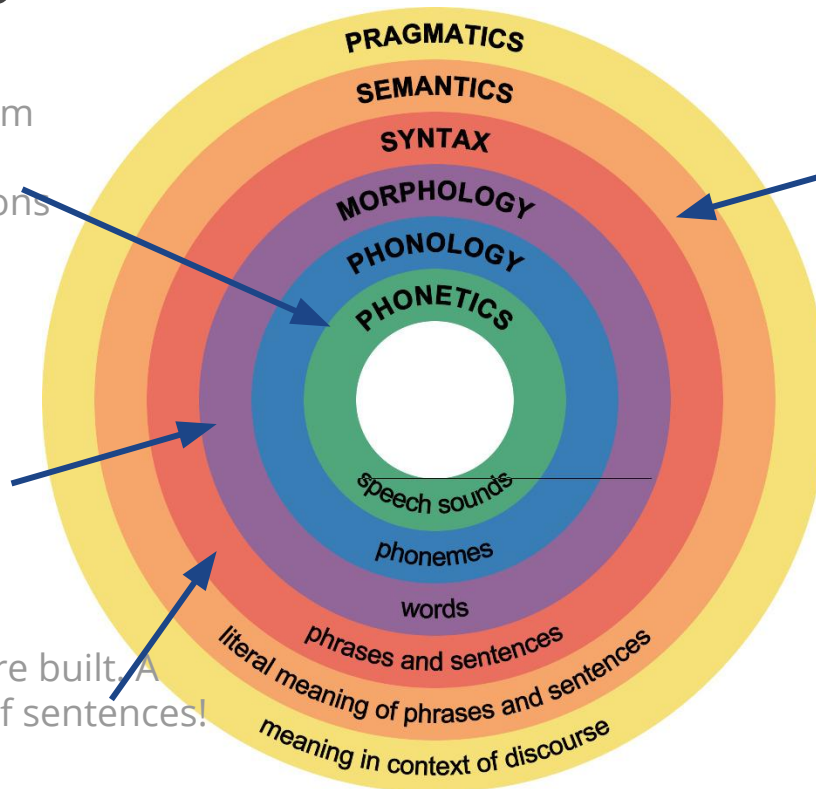
How words are built. Studies on new words creation: people are very creative!

## 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:

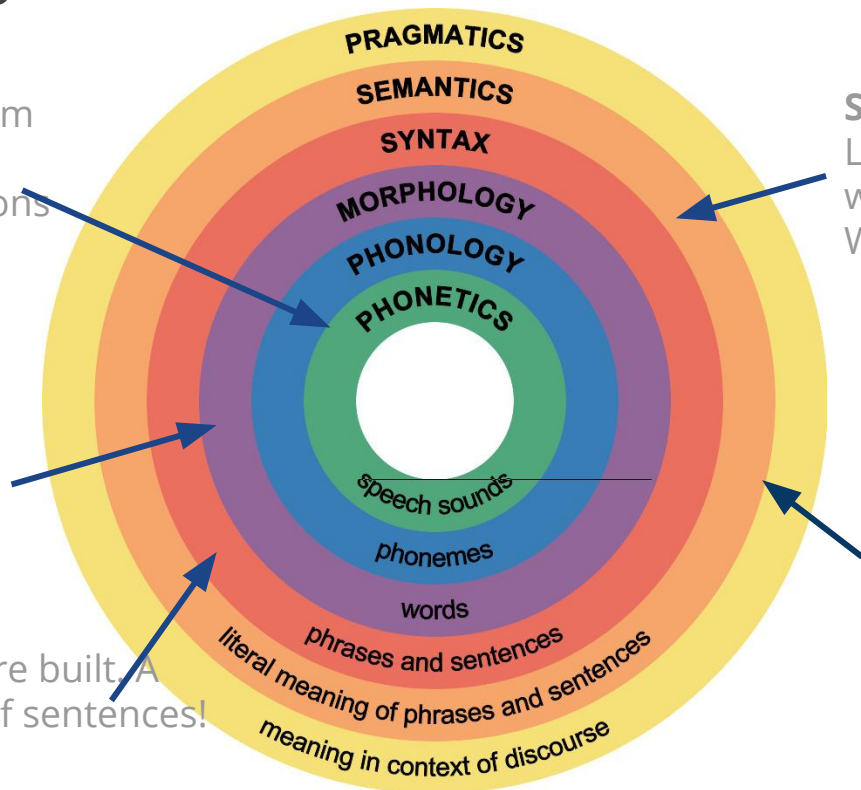
Speech processing is distinct from NLP = focus on text. But NLP researchers work on transcriptions from speech.

## 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?

# Fundamentals of NLP



# Pre-processing

Basic operations / pre-processing needed for almost any applications:

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



# 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?

Sentence: 这是一篇有趣的文章

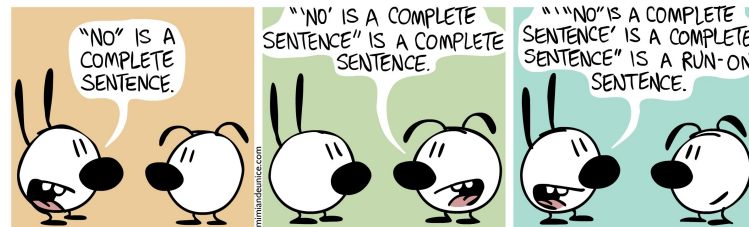
Words: 这是 一篇 有趣 的 文章

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

- 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)



# Sentences



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

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

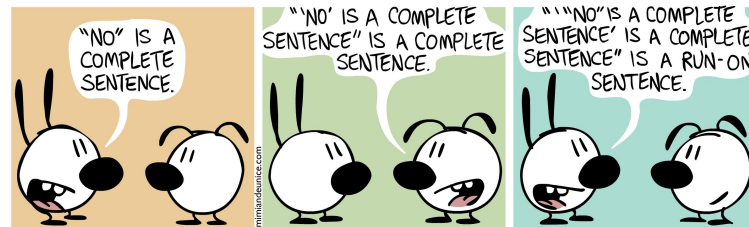
→ Crucial: Sentences are the input of many systems; Syntactic analysis is based on sentences

- ' 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

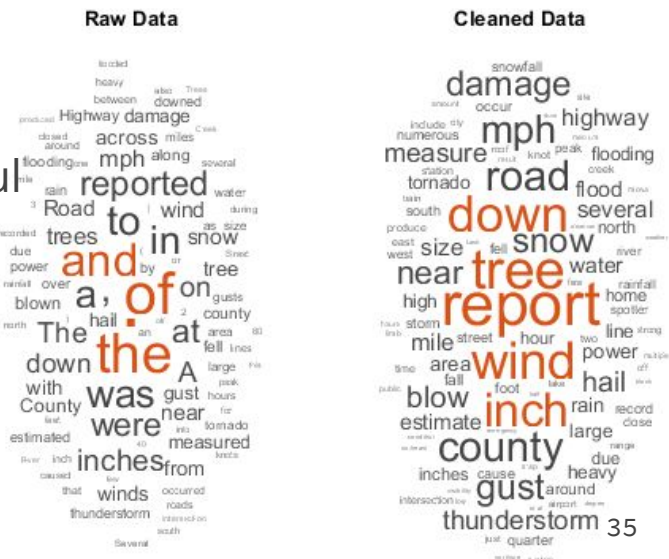
# Normalization

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>



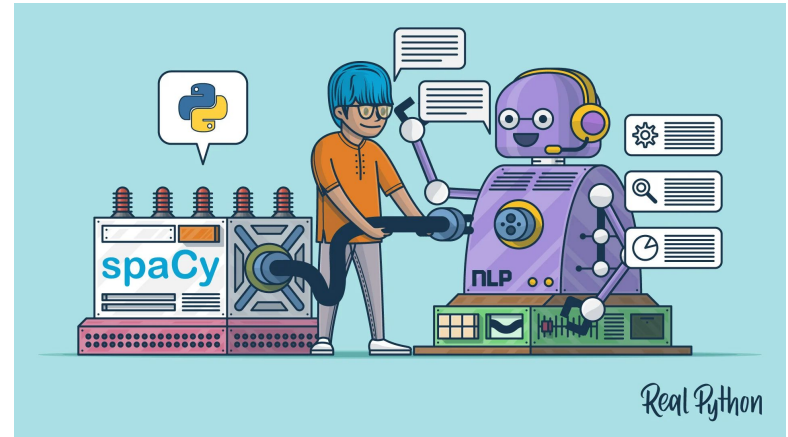
# Linguistic Analysis

Additional processing that can provide rich information:

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

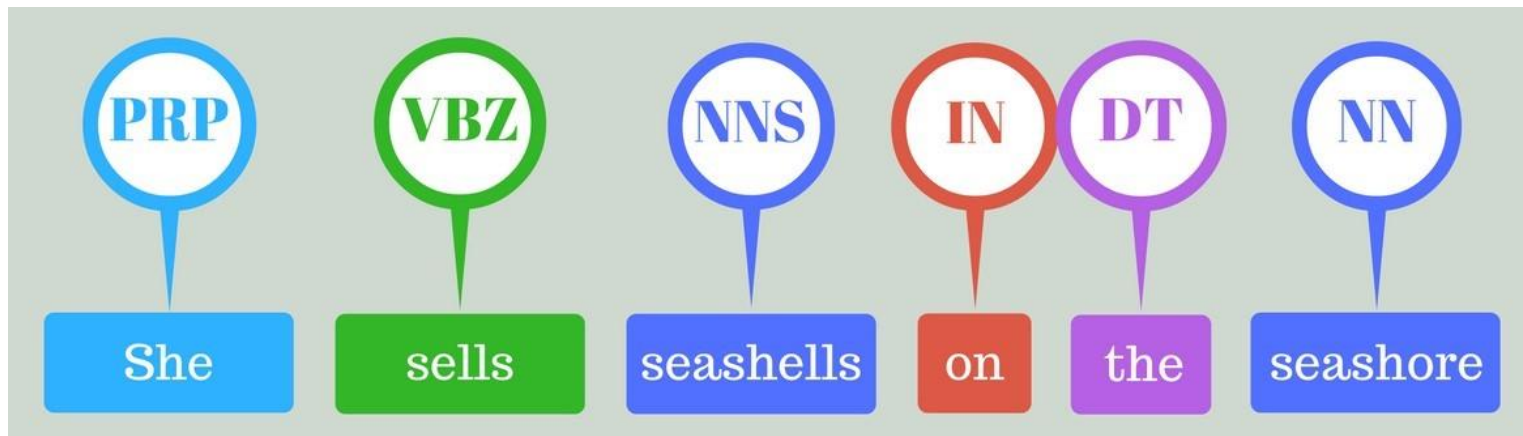
→ statistical models

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



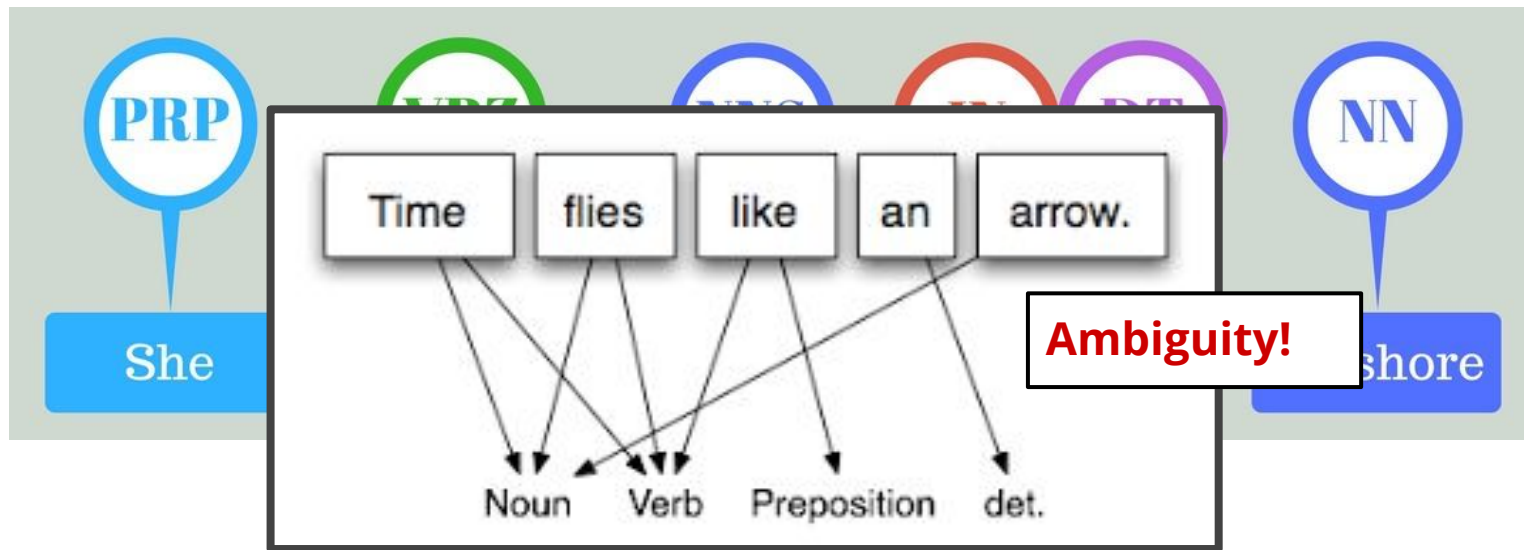
# Part-Of-Speech Tagging

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



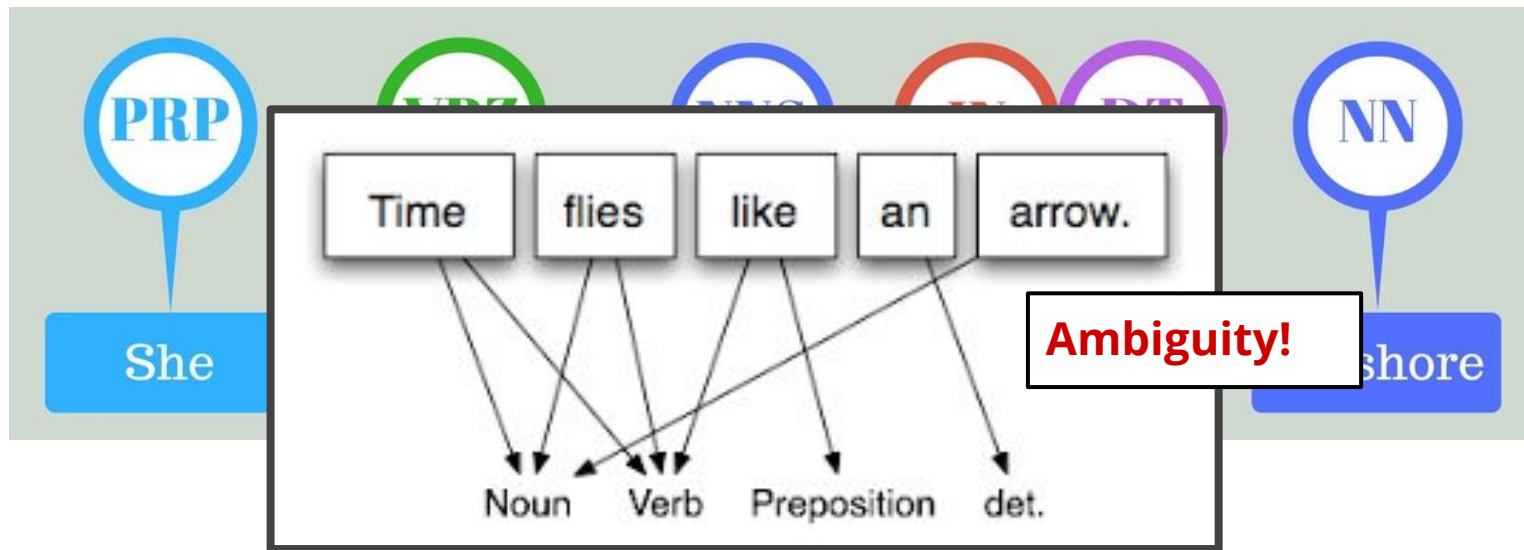
# Part-Of-Speech Tagging

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



# Part-Of-Speech Tagging

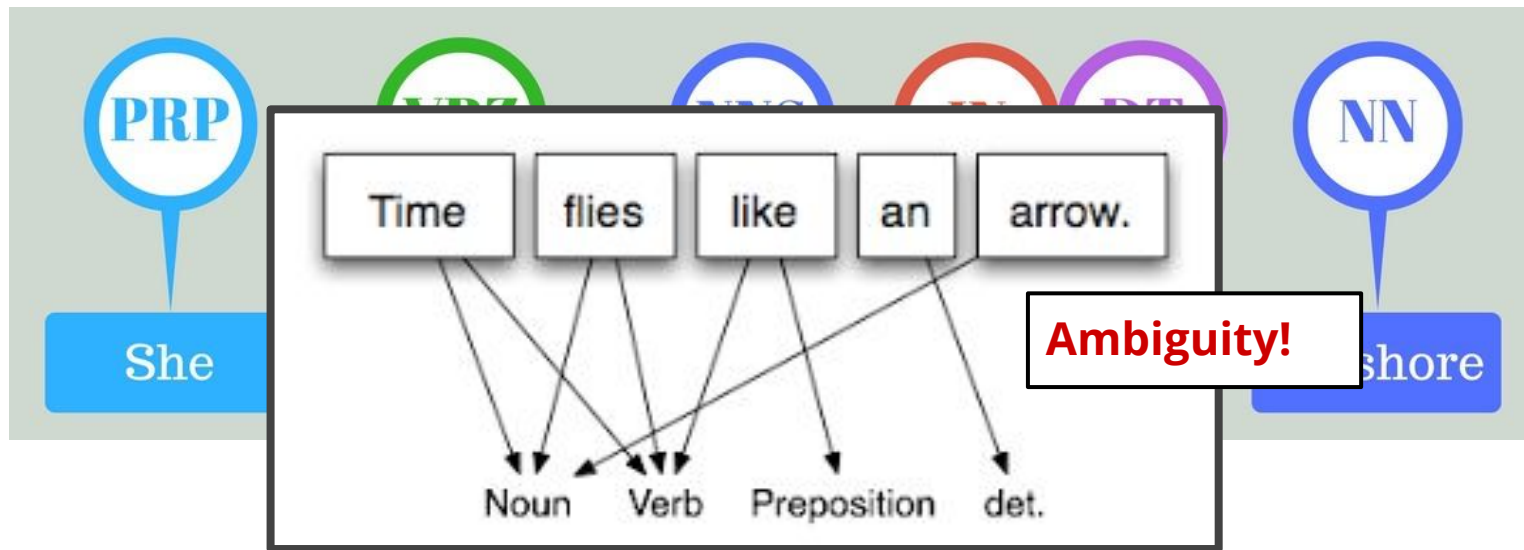
= Associating a morpho-syntactic category to each word  
→ Noun, Verb, Pronoun, Adjective, Adverb ...



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

# 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%

Track progress in NLP:

<https://github.com/sebastianruder/NLP-progress>



# 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 .

# Syntactic analysis

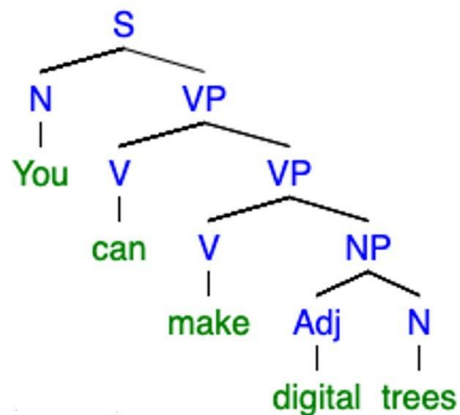
= Grouping words into phrases, and phrases into sentences

- *You can make (**constituent trees**) / (**beautiful drawings**).* → Noun Phrase
- *You (**make something**) / (**fly**).* → Verb Phrase
- *You make (**very interesting and well designed**) / (**boring**) presentations*  
→ Adjective Phrase

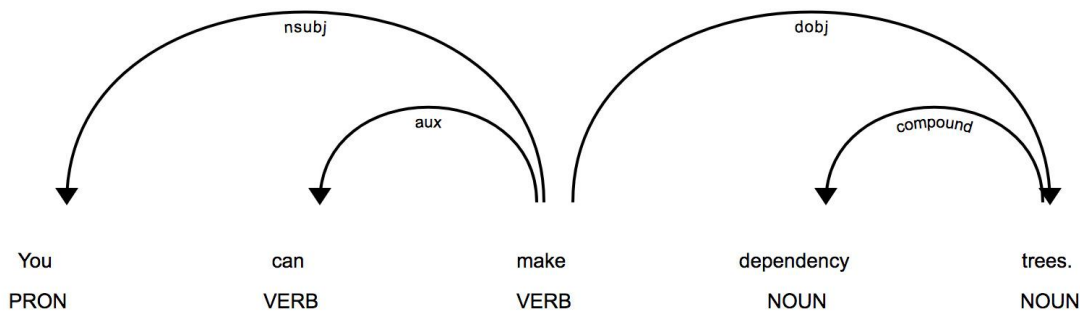
→ Hierarchical organization

# Syntactic analysis

Two main paradigms: constituency and dependency



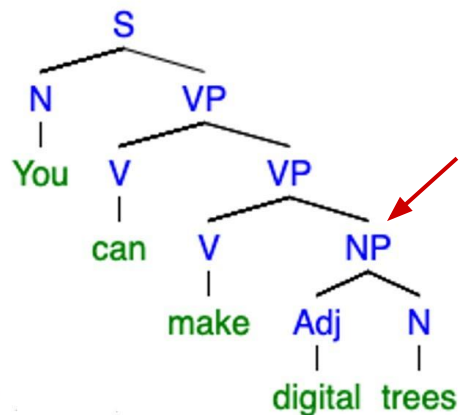
Constituency trees



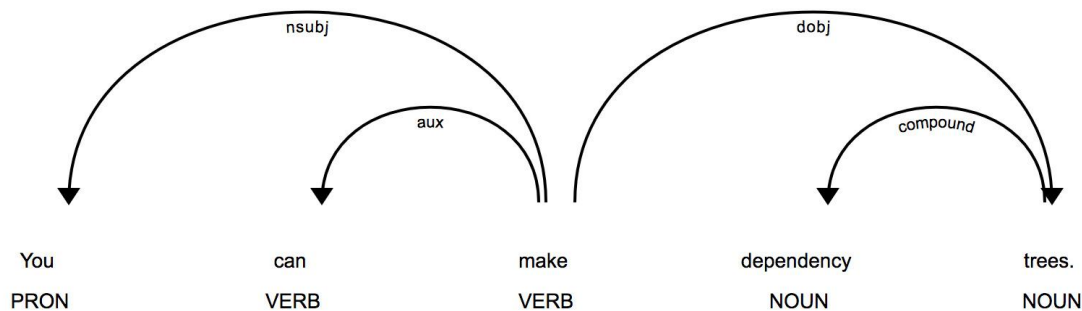
Dependency trees

# Syntactic analysis

Two main paradigms: constituency and dependency



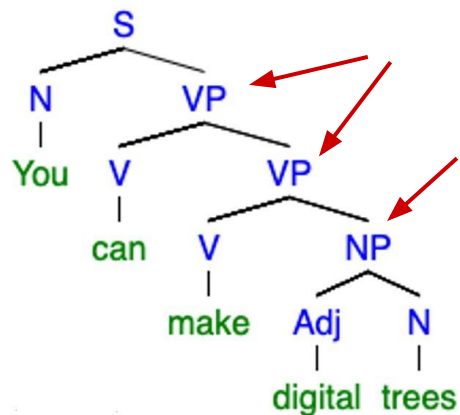
Constituency trees



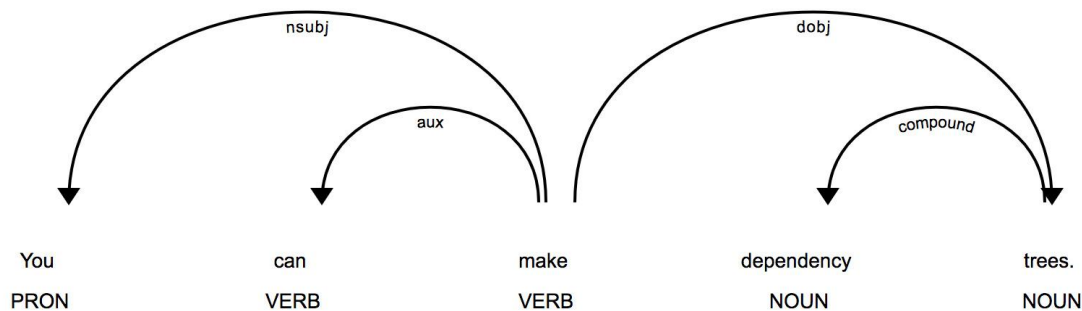
Dependency trees

# Syntactic analysis

Two main paradigms: constituency and dependency



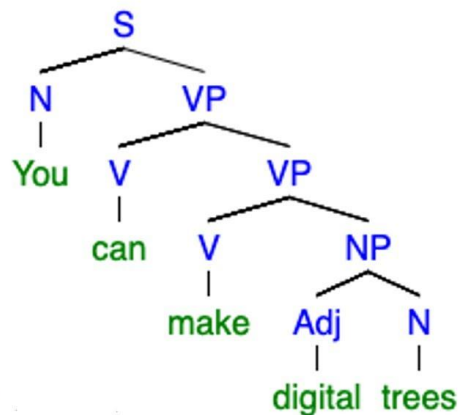
Constituency trees



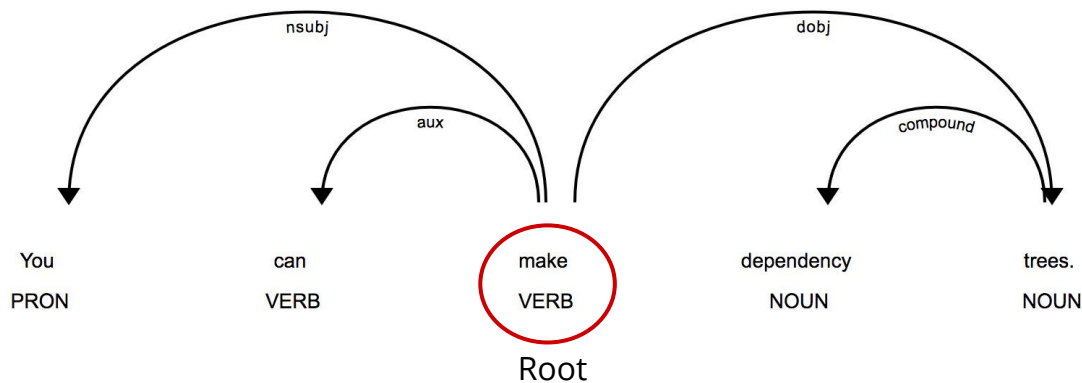
Dependency trees

# Syntactic analysis

Two main paradigms: constituency and dependency



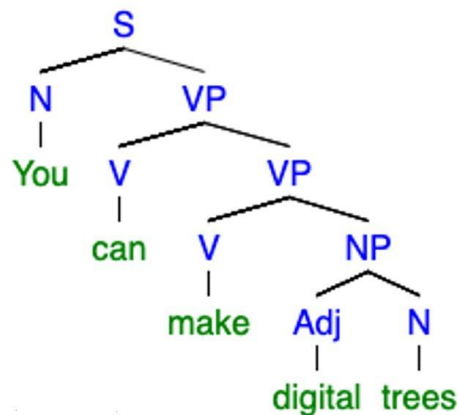
Constituency trees



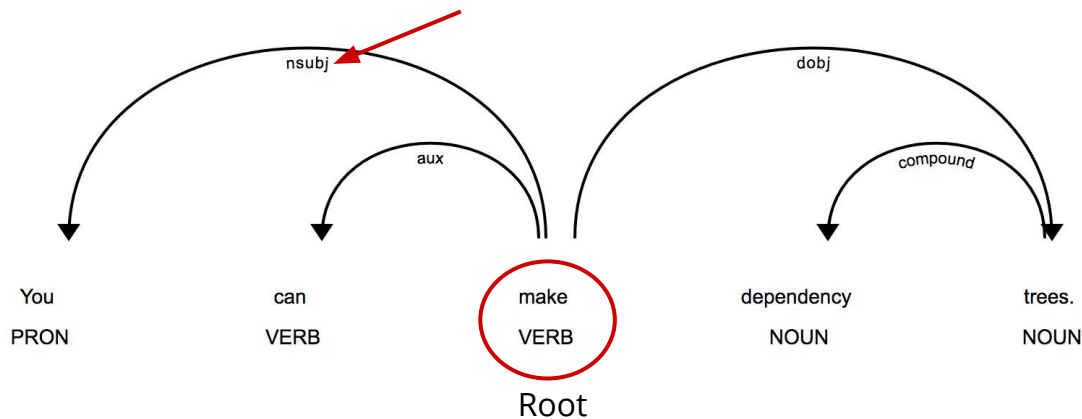
Dependency trees

# Syntactic analysis

Two main paradigms: constituency and dependency



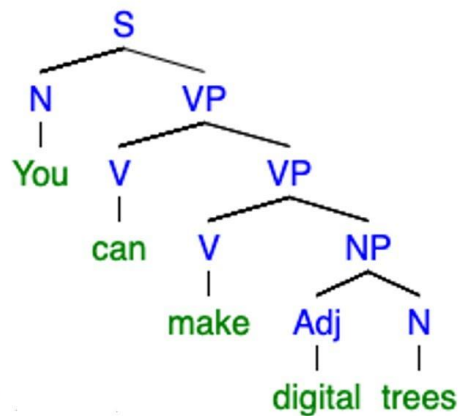
Constituency trees



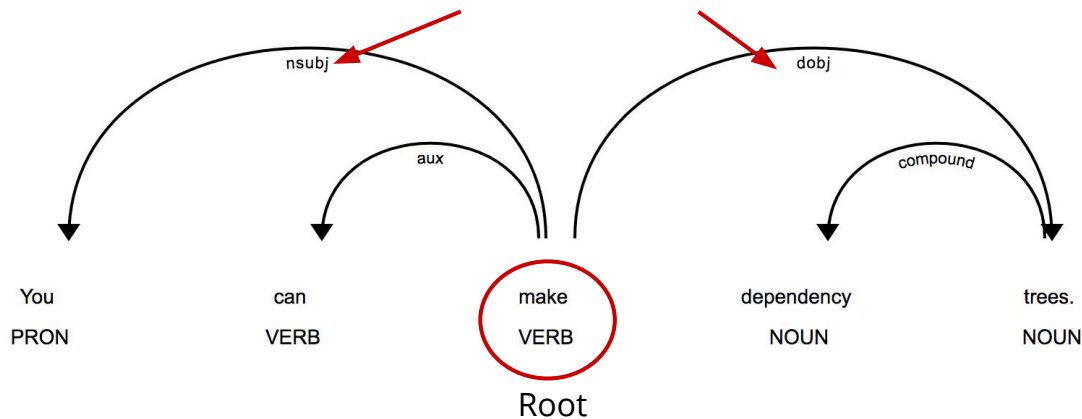
Dependency trees

# Syntactic analysis

Two main paradigms: constituency and dependency



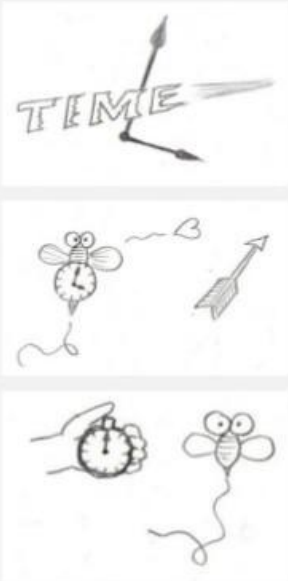
Constituency trees



Dependency trees



# Syntactic analysis



Time flies like an arrow.

Time flies like an arrow.

Time flies like an arrow.

The image shows three different visual interpretations of the sentence "Time flies like an arrow". Each interpretation is accompanied by a syntactic tree diagram. The first interpretation shows the word "TIME" written on an arrow, with a tree where the subject is NP (Time), the verb is V (flies), and the prepositional phrase is PP (like an arrow). The second interpretation shows a clock with wings flying towards an arrow, with a tree where the subject is NP (Time), the verb is V (flies), and the object is NP (an arrow). The third interpretation shows a clock and a fly, with a tree where the verb is V (flies), the subject is NP (Time), and the prepositional phrase is PP (like an arrow).

**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  $\approx 96\%$  ; Dependency  $\approx 97\%$

# Semantics: Words

Words disambiguation: homophones and homonyms

- Homophones: mostly a problem for speech processing
- Homonyms:
  - makes syntactic analysis harder: Noun or Verb?
  - makes semantic analysis harder: what sense?
- State-of-the-art: English  $\approx$  73-80% depending on granularity



# Semantico-pragmatics: Discourse

Going beyond sentence boundaries

= connecting spans of text into a coherent, meaningful document

= understanding interaction in conversations

- coreference and anaphora
- temporal links
- topics
- discourse / rhetorical relations



# Semantico-pragmatics: Discourse

Going beyond sentence boundaries

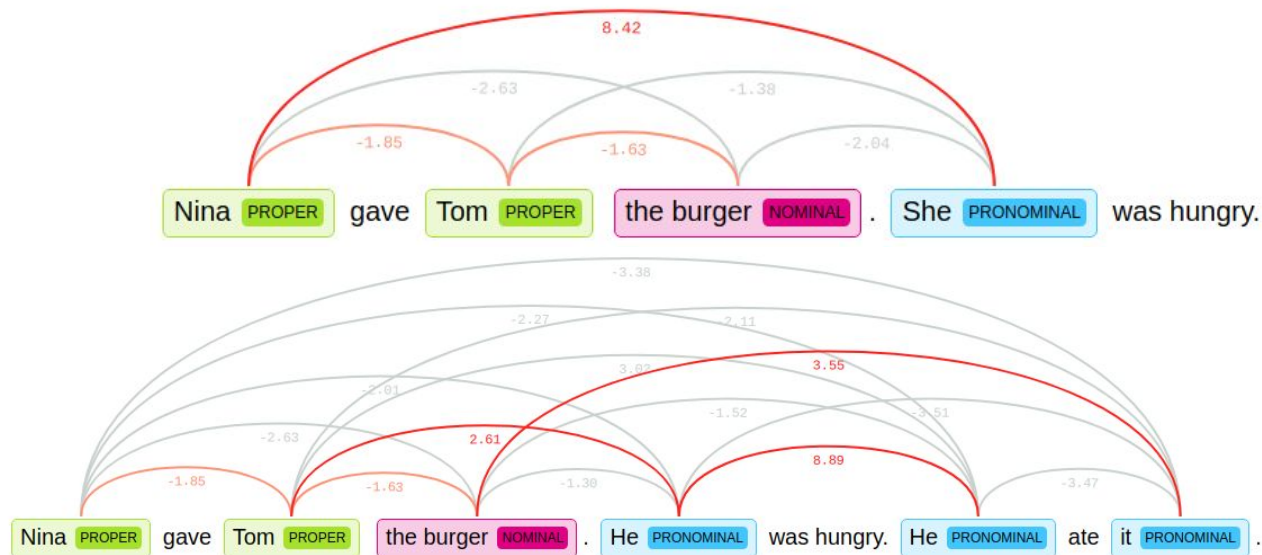
= connecting spans of text into a coherent, meaningful document

= understanding interactions in conversations

- **coreference and anaphora**
- temporal links
- topics
- **discourse / rhetorical relations**



# Coreference



Finding expressions / entities (person, animals, things...) that are linked together:

- nouns to other nouns
- nouns to pronouns
- pronouns to other pronouns

State-of-the-art: English  $\approx$  80%

Examples from Neuralcoref (by HuggingFace), try it: <https://huggingface.co/coref/>



# Discourse parsing

= Linking sentences or clauses with semantico-pragmatic relations such as *Explanation, Result, Contrast...*

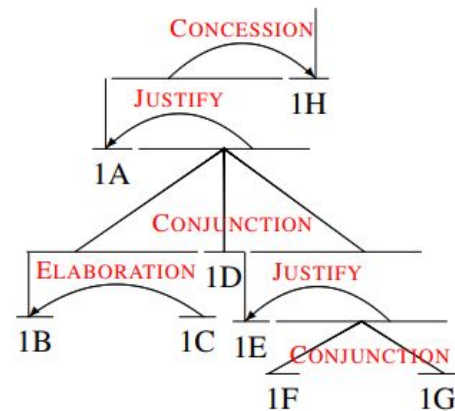
- (1) The train derailed. (Result) There were four wounded people. (Explanation) The driver was going too fast.

→ Could help a QA system to answer complex questions:

e.g., why did the train derailed?

Very hard task: State-of-the-art: English ≈ 57.4%

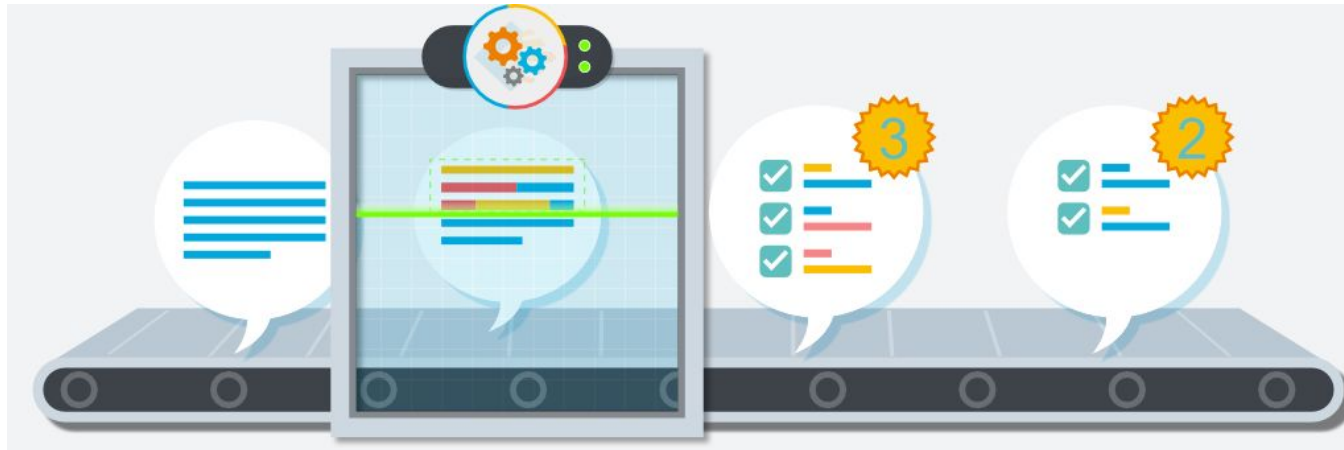
- (2) Tina gave Tom the burger. (because / Explanation) **He** was hungry.
- (3) Tina gave Tom the burger. (although / Contrast) **She** was hungry.



[It could have been a great movie]<sup>1A</sup> [It does have beautiful scenery,]<sup>1B</sup> [some of the best since Lord of the Rings.]<sup>1C</sup> [The acting is well done,]<sup>1D</sup> [and I really liked the son of the leader of the Samurai.]<sup>1E</sup> [He was a likable chap,]<sup>1F</sup> [and I hated to see him die.]<sup>1G</sup> [But, other than all that, this movie is nothing more than hidden rip-offs.]<sup>1H</sup>

[Bahtia et al., 2015]

# Hands-on NLP



# Finding data



Many datasets available:

- NLTK datasets and lexical resources: <https://www.nltk.org/book/ch02.html>
- Scikit-learn datasets: <https://scikit-learn.org/stable/datasets.html>
- Kaggle: <https://www.kaggle.com/datasets?tags=13204-NLP>
- LDC (charged): <https://www ldc.upenn.edu/>
- A list here: <https://github.com/niderhoff/nlp-datasets>
- Research teams websites
  - <https://nlp.stanford.edu/links/statnlp.html#Corpora>
  - <http://www.llf.cnrs.fr/en/resources>



# Example: Sentiment analysis with movies reviews

- Reviews are texts of varied size
- Written by anyone
- Describing a sentiment about something (here movies)
- The task: classify each review as positive or negative
  - sometimes, a neutral category is added
  - more fine-grained: retrieve the number of stars / likes



most of the **problems** with the film don't derive from the screenplay , but rather the **mediocre** performances by most of the actors involved



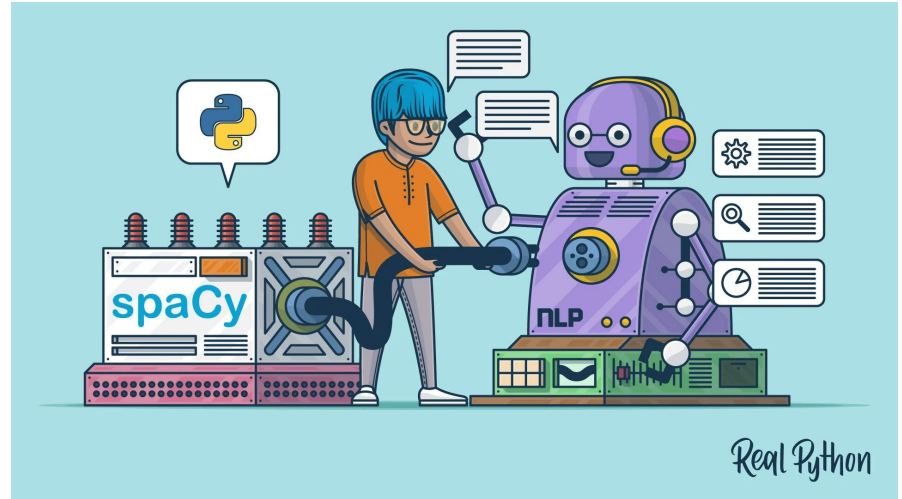
the film provides some great **insight** into the neurotic mindset of all comics -- even those who have reached the **absolute top** of the game .

# Preprocessing with NLTK or SpaCy



Natural Language Analysis  
with Python NLTK

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



<https://spacy.io/>

# Example: Tokenisation and sentence segmentation using SpaCy

```
[14] import spacy
      nlp = spacy.load('en_core_web_sm')

      # Tokenization
      sentence = 'Nina gave Tom the burger.'

      # Preprocess using spacy's pipeline
      doc = nlp(sentence)

      for token in doc:
          print( token.text )
```

```
Nina
gave
Tom
the
burger
.
```

```
▶ # Sentence segmentation
  text = 'Nina gave Tom the burger. He was hungry. She ate it.'

  # Preprocess using spacy's pipeline
  doc = nlp(text)

  # Print the sentences
  for i, sent in enumerate( doc.sents ):
      print( i, sent.text.strip() )
```

```
0 Nina gave Tom the burger.
1 He was hungry.
2 She ate it.
```

# Example: Part-Of-Speech Tagging with SpaCy

```
[16] import spacy
import pandas as pd
nlp = spacy.load('en')

sentence = "Time flies like an arrow."
nlp_sentence = nlp(sentence)
spacy_pos_tagged = [(w, w.tag_, w.pos_) for w in nlp_sentence]
print(spacy_pos_tagged )

pd.DataFrame(spacy_pos_tagged,
             columns=['Word', 'POS tag', 'Tag type'])

[(Time, 'NNP', 'PROPN'), (flies, 'VBZ', 'VERB'), (like, 'IN', 'SCONJ'), (an, 'DT', 'DET'), (arrow, 'NN', 'NOUN'), (., '.', 'PUNCT')]
```

# Example: using NLTK

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

text = "This ice-cream is from the U.K.. This isn't a cake. "
for sentence in nltk.sent_tokenize(text):
    tokens = nltk.word_tokenize(sentence)
    tagged_tokens = nltk.pos_tag(tokens) # POS tagging
    print(tagged_tokens)

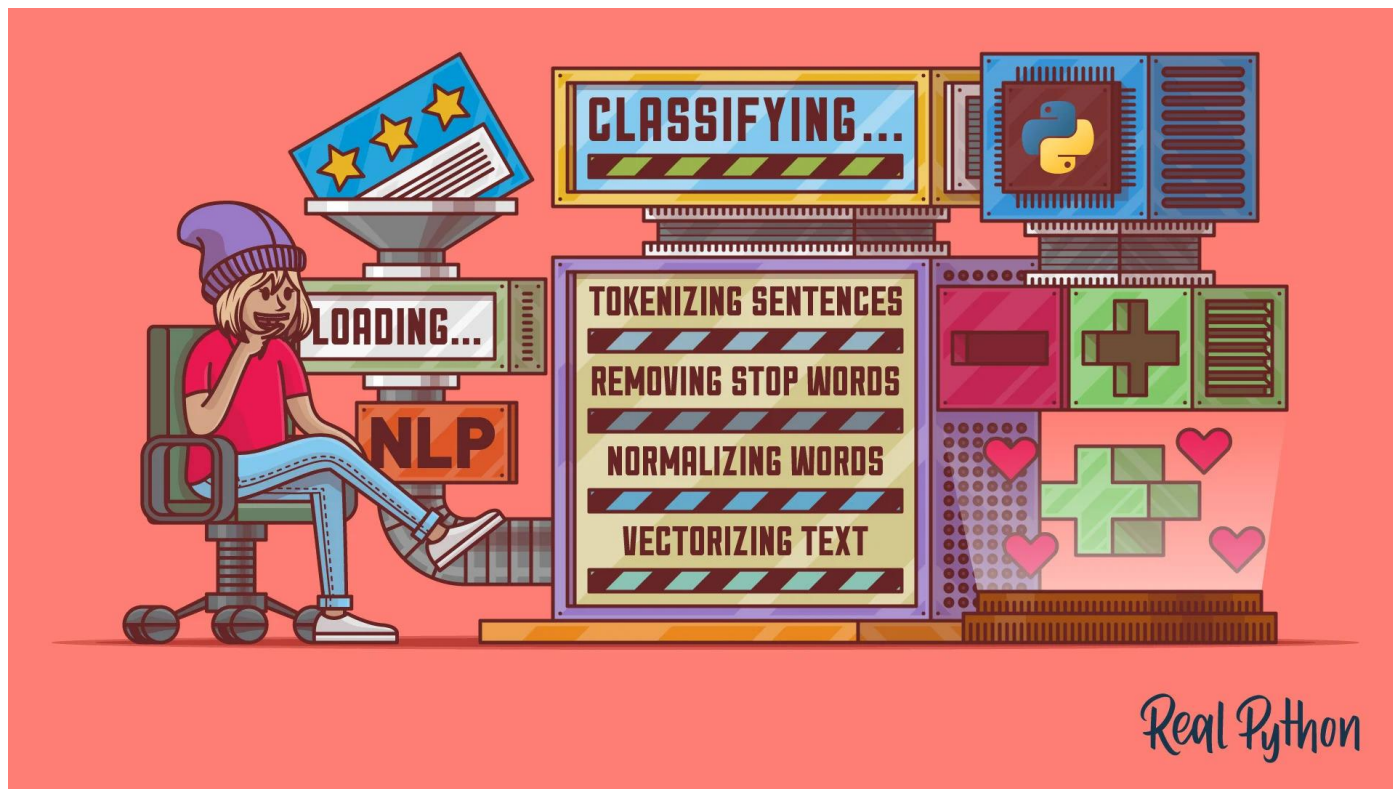
[('This', 'DT'), ('ice-cream', 'NN'), ('is', 'VBZ'), ('from', 'IN'), ('the', 'DT'), ('U.K..', 'NNP')]
[('This', 'DT'), ('is', 'VBZ'), ("n't", 'RB'), ('a', 'DT'), ('cake', 'NN'), ('.', '.')]

sentence = "Time flies like an arrow."
tokens = nltk.word_tokenize(sentence)
tagged_tokens = nltk.pos_tag(tokens) # POS tagging
print(tagged_tokens)

[('Time', 'NNP'), ('flies', 'VNS'), ('like', 'IN'), ('an', 'DT'), ('arrow', 'NN'), ('.', '.')]

```

# Playing with data: let's learn something



# 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**
  - combine linear and non-linear functions
  - improved performance (in general)
  - harder to interpret ("black-box")

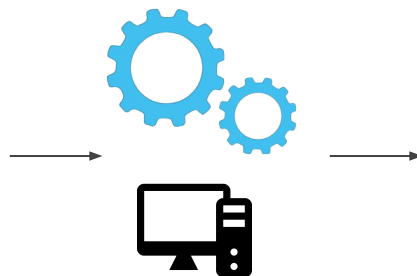


# Learning from data (reminder, supervised setting)

Input Data / Training set

Data points $x$	Labels $y$
	Cat
	Sandwich

Training



building some  
function  $f(x) = y$

Test set



Use  $f$  to  
predict a label

Compute score

CAT ?

20/20  
Très bien !

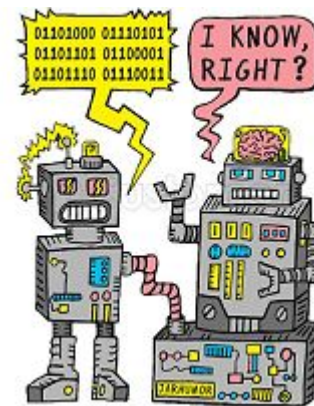


# Learning from text data

Main issue:

- how to represent text?

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

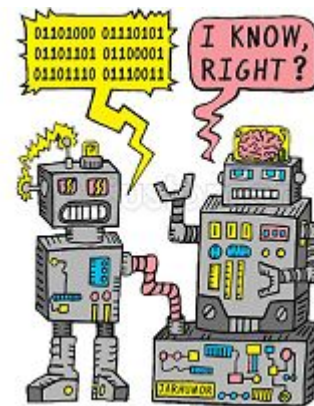


# Learning from text data

Main issue:

- how to represent text?

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



## Bag-of-Words (BOW):

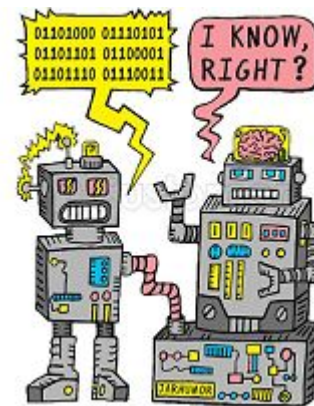
- one vector where each dimension is a word / feature in our vocabulary
- if the word / feature is present in the document, associate a specific value

# Learning from text data

Main issue:

- how to represent text?

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



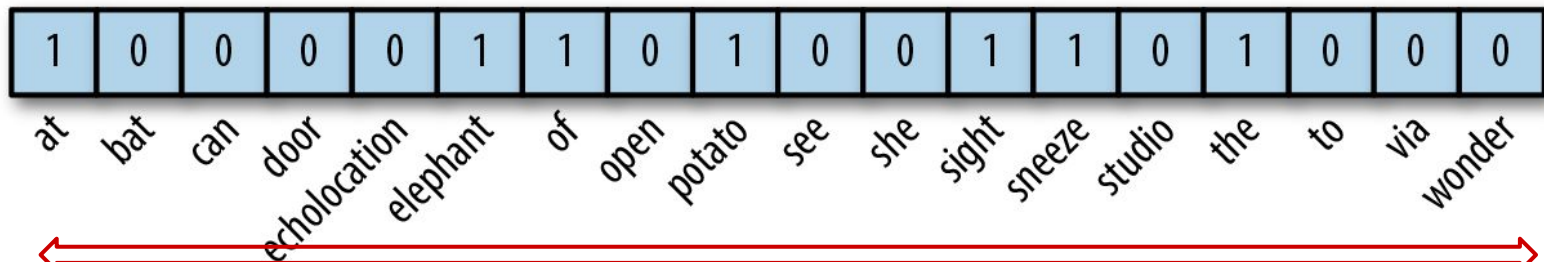
## Bag-of-Words (BOW):

- 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

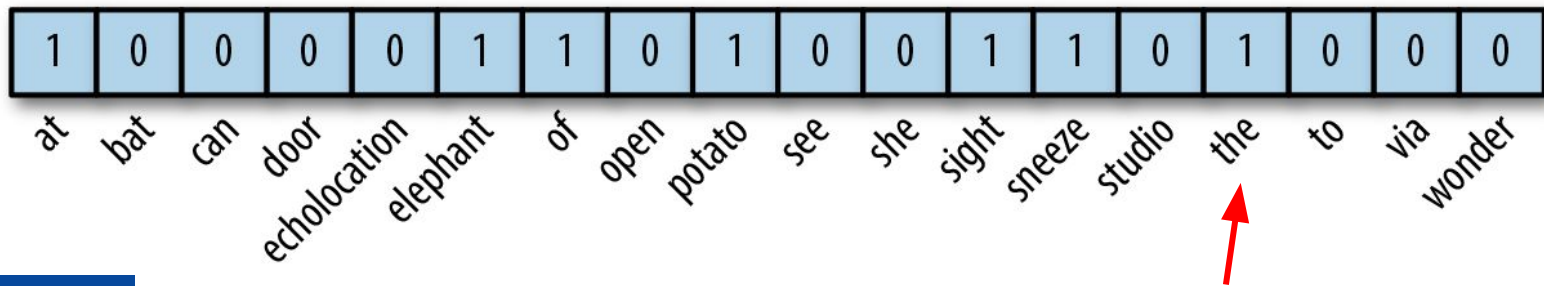


18 words / dimensions

# 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

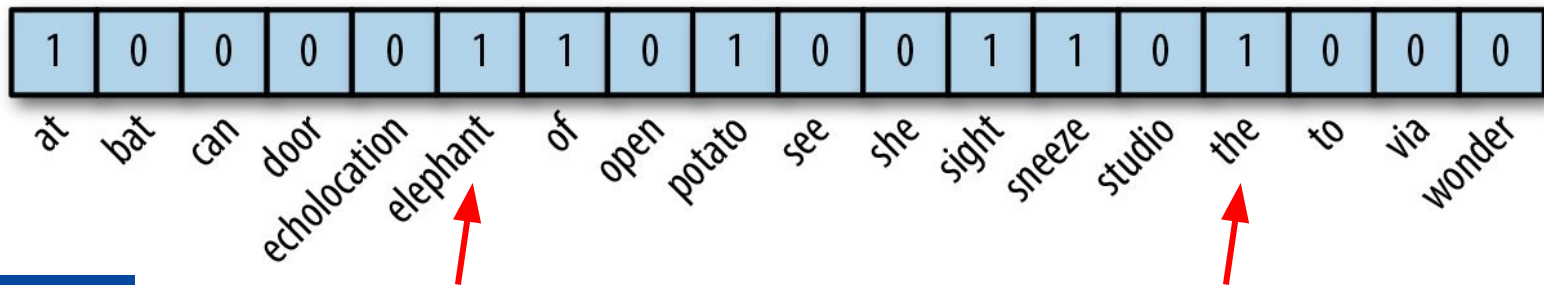




# BOW: One-hot encoding

The elephant sneezed  
at the sight of potatoes.

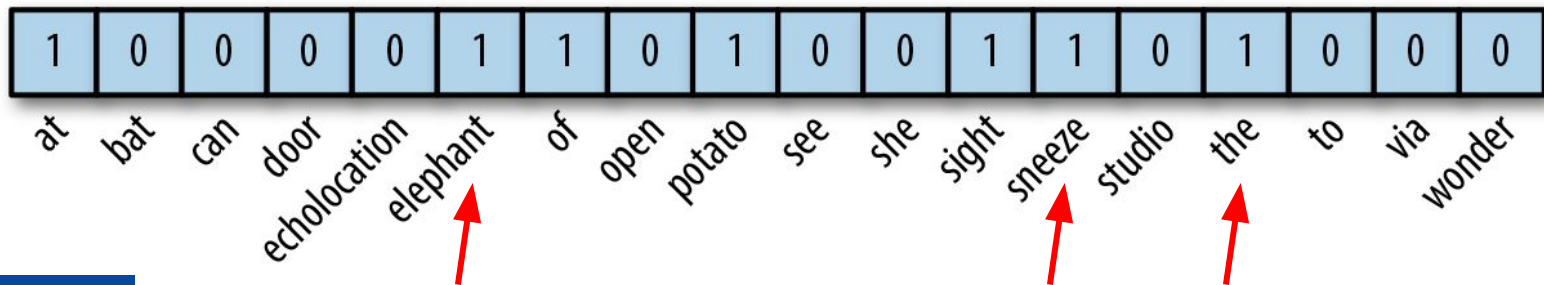
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



# BOW: One-hot encoding

The elephant sneezed  
at the sight of potatoes.

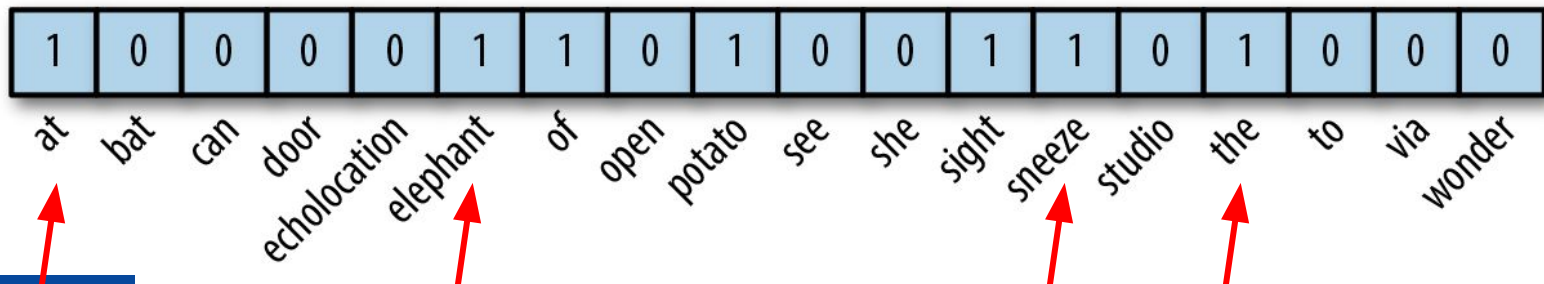
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



# BOW: One-hot encoding

The elephant sneezed  
at the sight of potatoes.

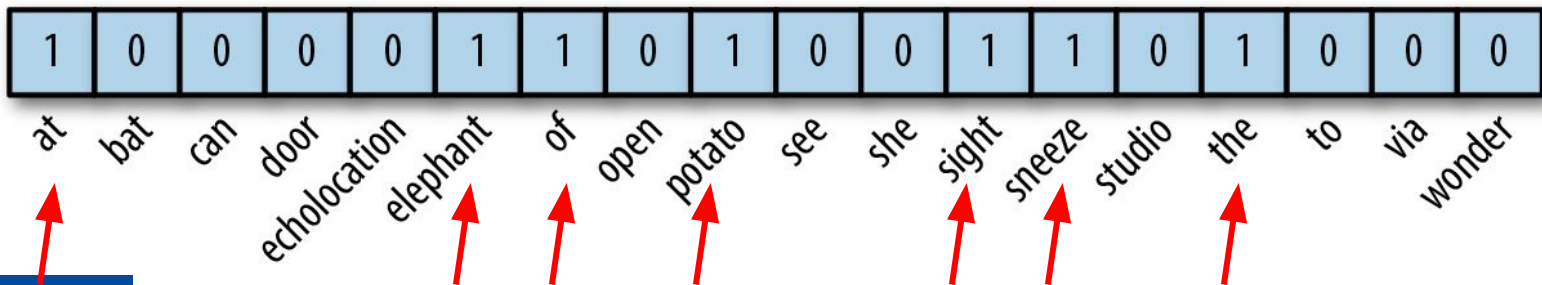
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



# BOW: One-hot encoding

The elephant sneezed  
at the sight of potatoes.

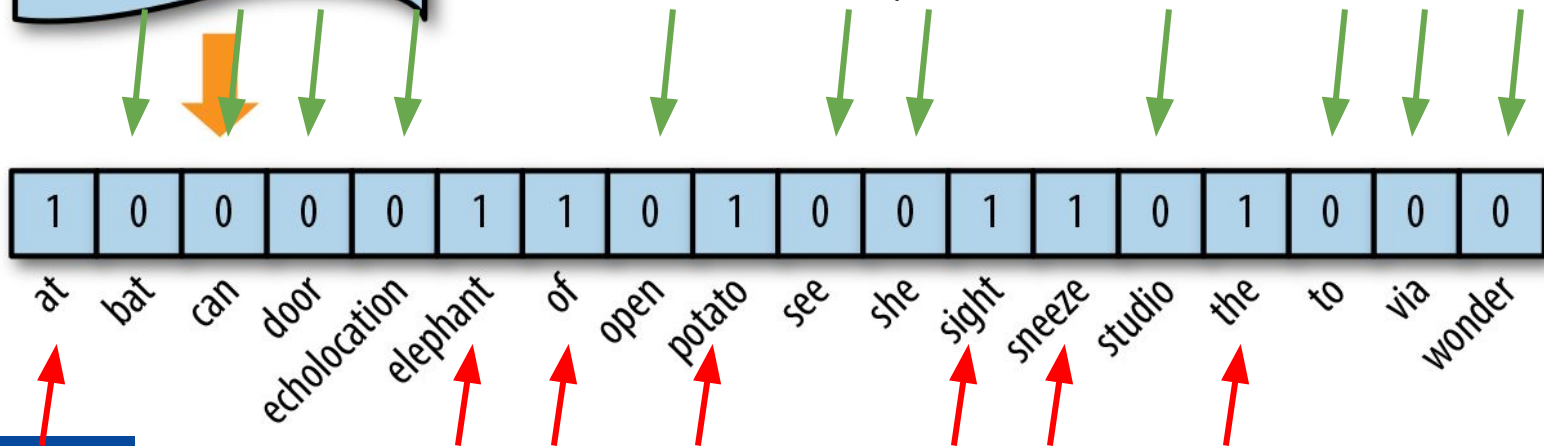
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1



# BOW: One-hot encoding

The elephant sneezed  
at the sight of potatoes.

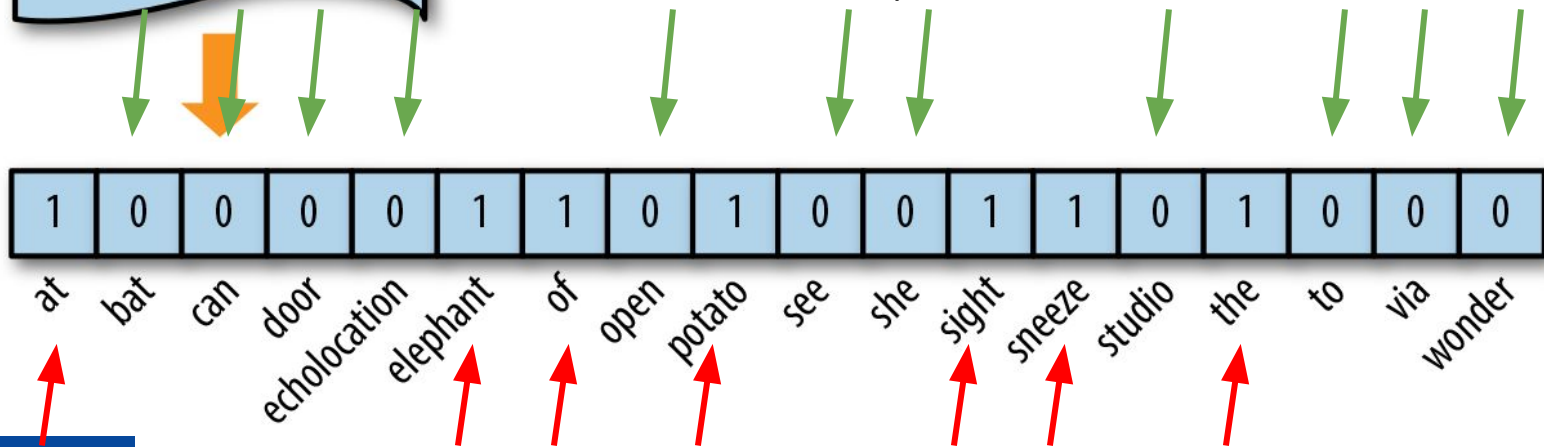
- First, build a vocabulary: identify all the word in your data
- If the word is present in the sentence / document, value = 1
- The other words: value = 0 → present in the training data, but not in this specific sentence / document



# 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:

- 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

‘the’  




# 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

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

# 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

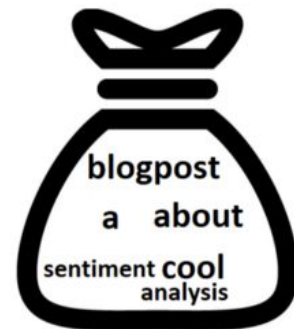
We can encode any information:

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

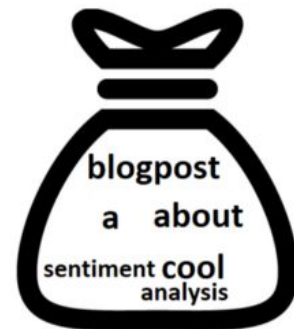
# Problems and extensions

## 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!

# 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



# 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 ) )
```

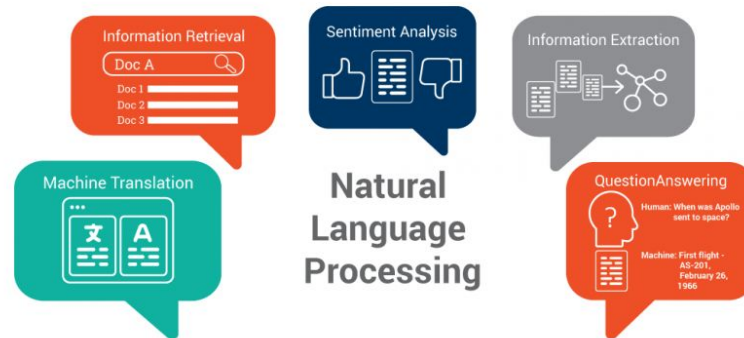
Vectorize

Train

Predict

Evaluate

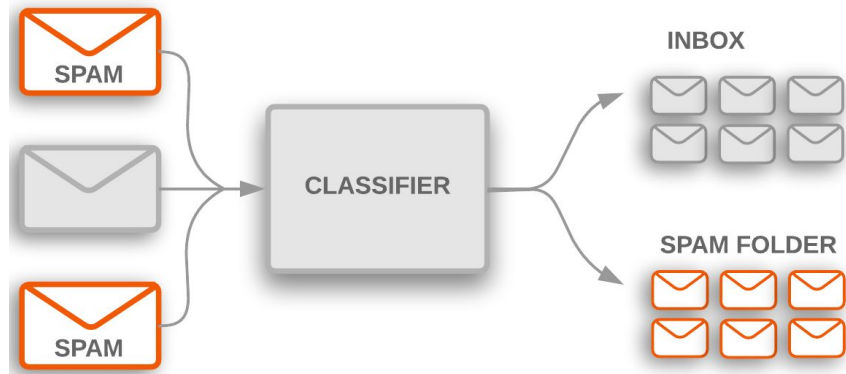
# NLP: tasks and applications



# 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
- Psychological trouble detection
- ....



# Sequence Labelling

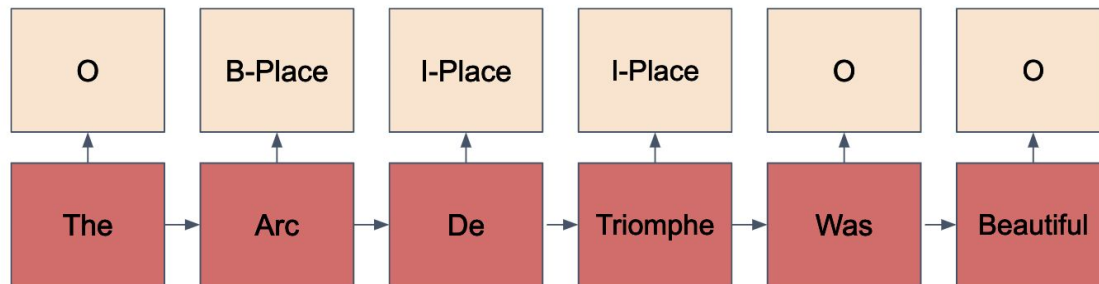
Idea: we want to take into account context, e.g. surrounding words

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

- POS tagging
- Named Entity Recognition

→ Without neurons: = CRF, HMMs ...

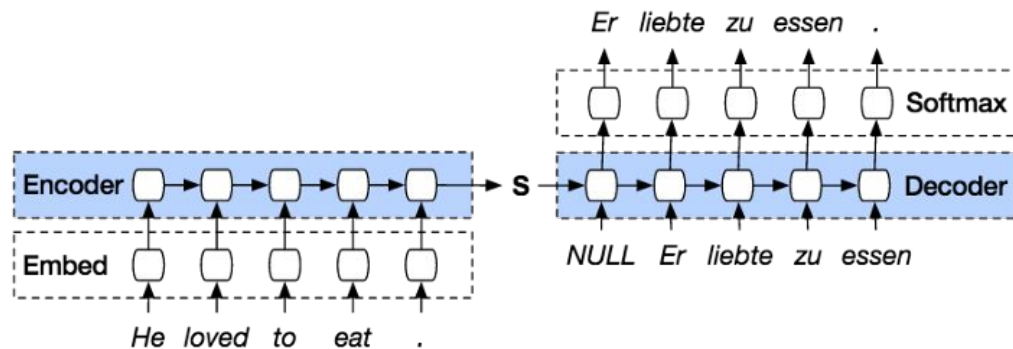
→ Now: e.g. with Recurrent Neural Network + CRF



# Sequence to Sequence

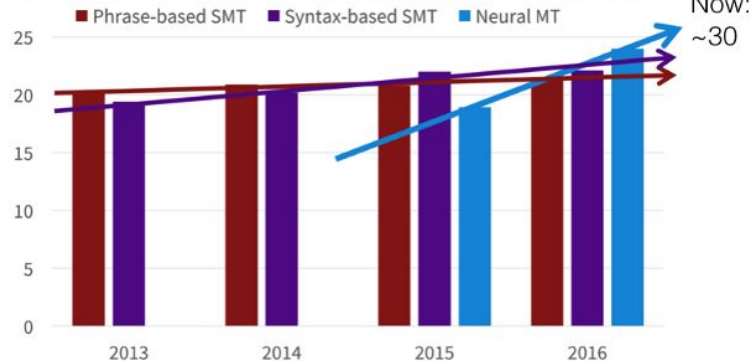
Idea: we want to take a sequence as input, and to output another sequence

- Machine translation
- Question-answering
- Summarization
- Generation, e.g. generating subtitles, captions, poetry...



## Progress in Machine Translation

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

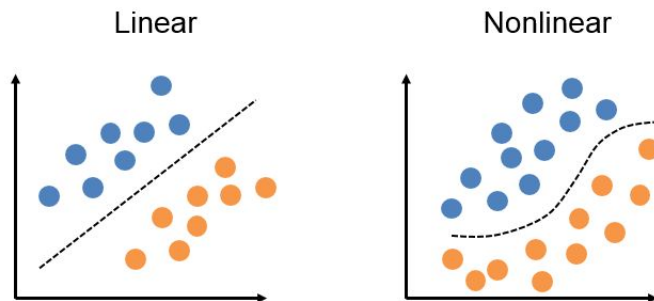


# To go further...



# Limits of classical statistical methods

1. Linearity → Neural methods, see next lectures!
2. Data representation: “Feature engineering”
3. Word similarity



# Data representation

- 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 ]



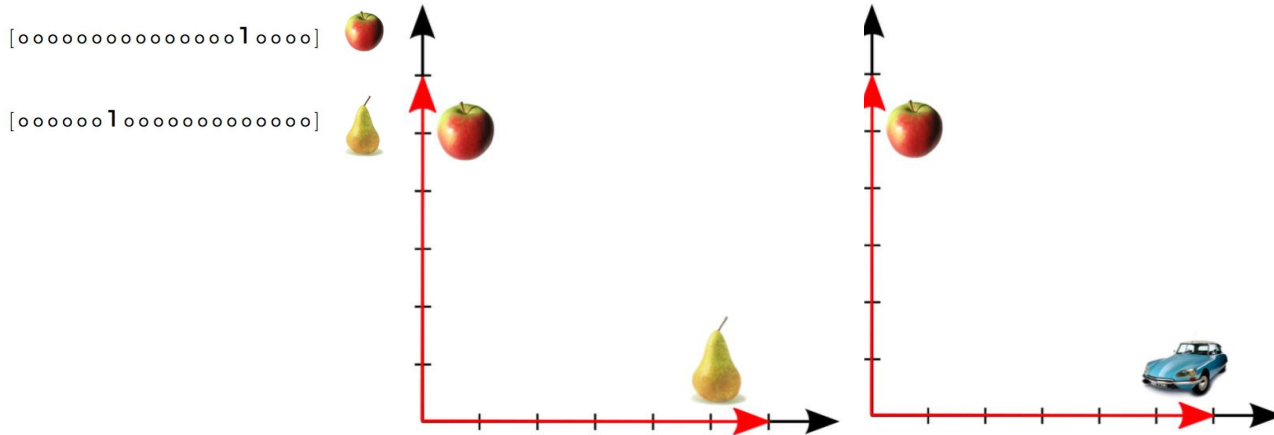
[ o o o o o o 1 o o o o o o o o o o o o o o o o ]





# Data representation

- Defining features has to be done manually: require expertise and tests
- A word is represented with a one-hot vector: easy to implement
- But representing 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

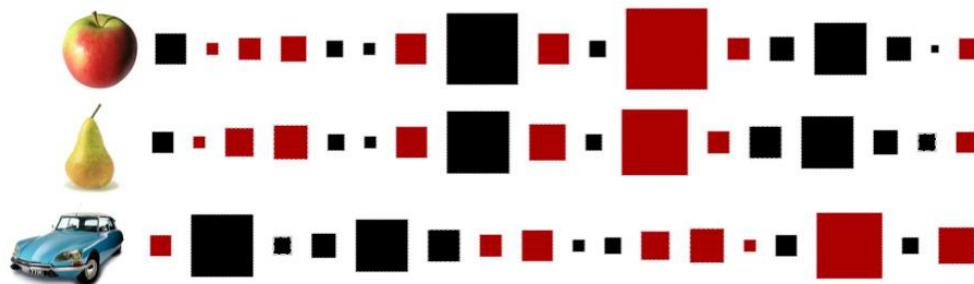
Many studies on representing the meaning of words using context

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.

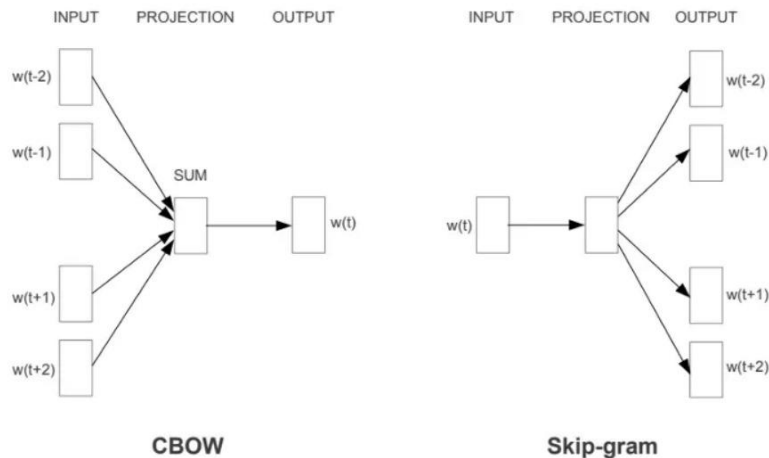
The quick brown **fox** jumps over the lazy dog.

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



# 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.

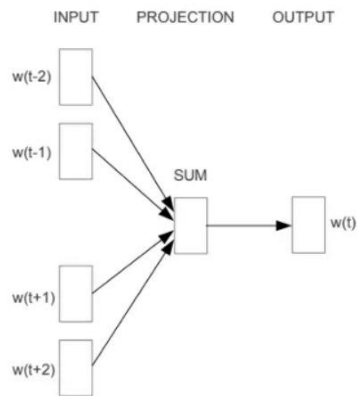
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

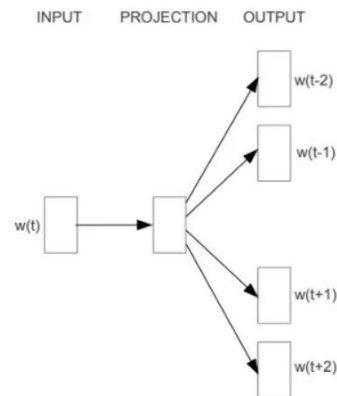
## Skip-gram:

- predict the context words given the target word

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



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.

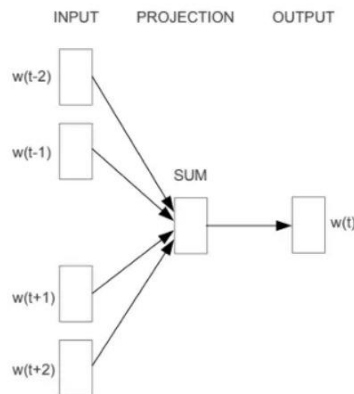
The quick brown **fox** jumps over the lazy dog.

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

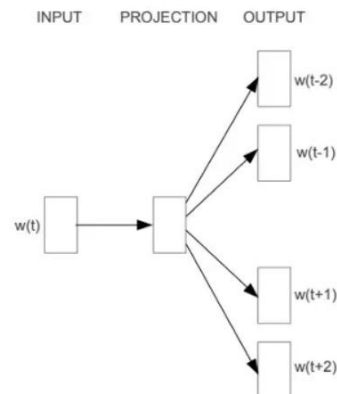
Skip-gram:

- predict the context words given the target word

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



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.

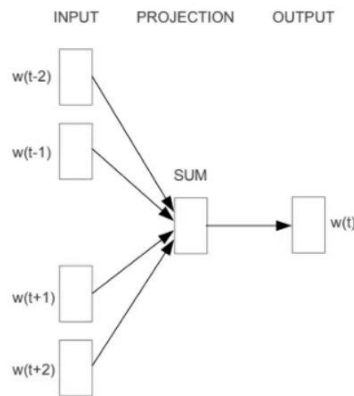
The quick brown **fox** jumps over the lazy dog.

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

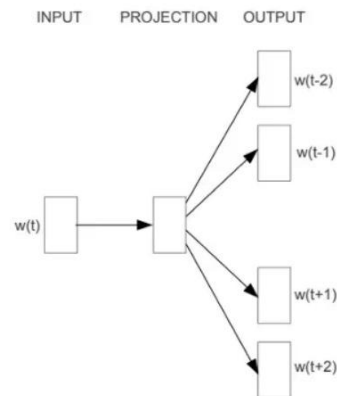
Skip-gram:

- predict the context words given the target word

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

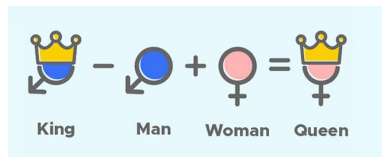


CBOW

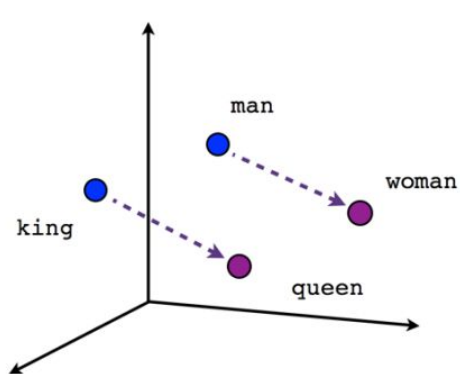


Skip-gram

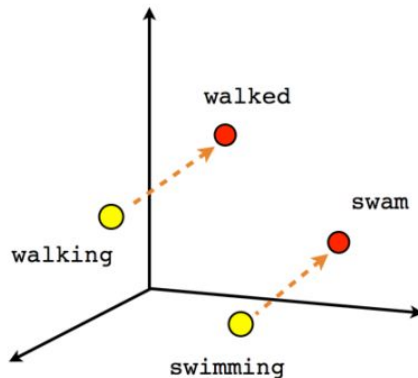
# Word2vec



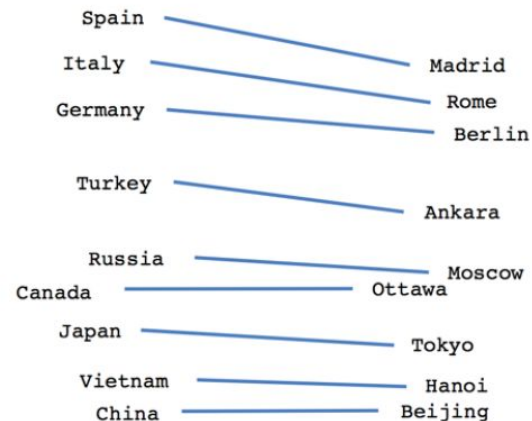
Allow inference:



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.txt

UMAP T-SNE PCA CUSTOM

Dimension: 2D 3D

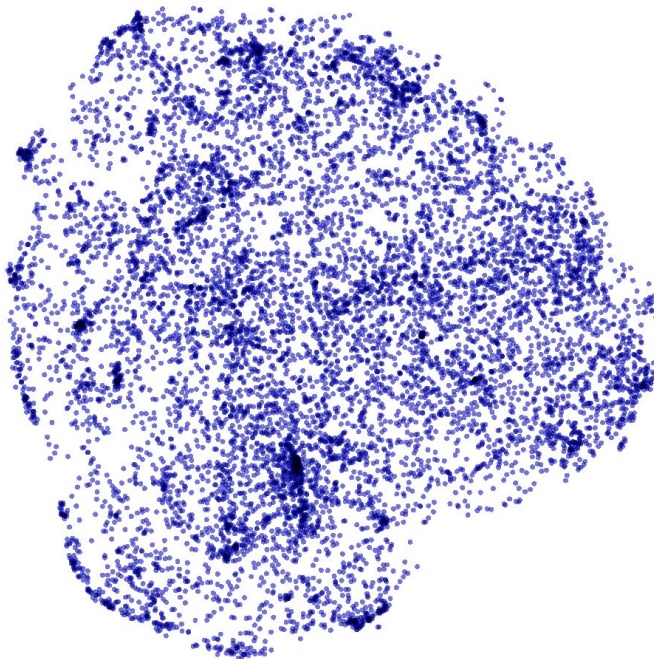
Perplexity: 25

Learning rate: 10

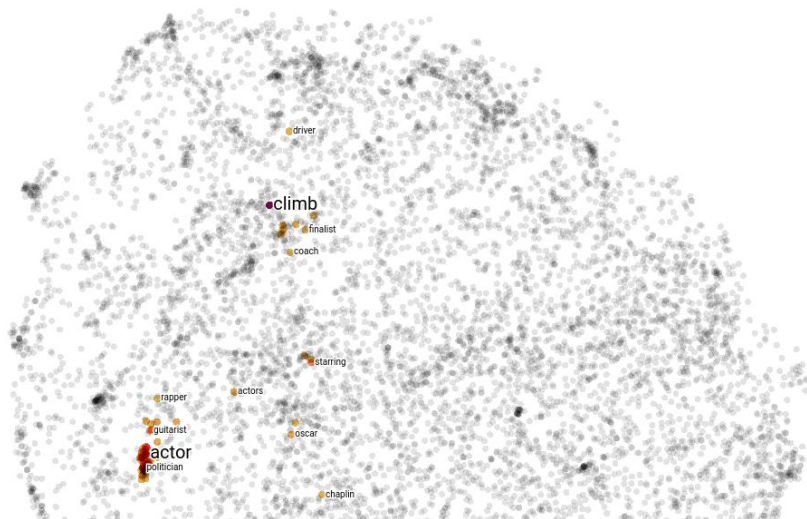
Supervise: 0

Re-run Pause Perturb

Iteration: 262



# Visualizing embeddings



actor

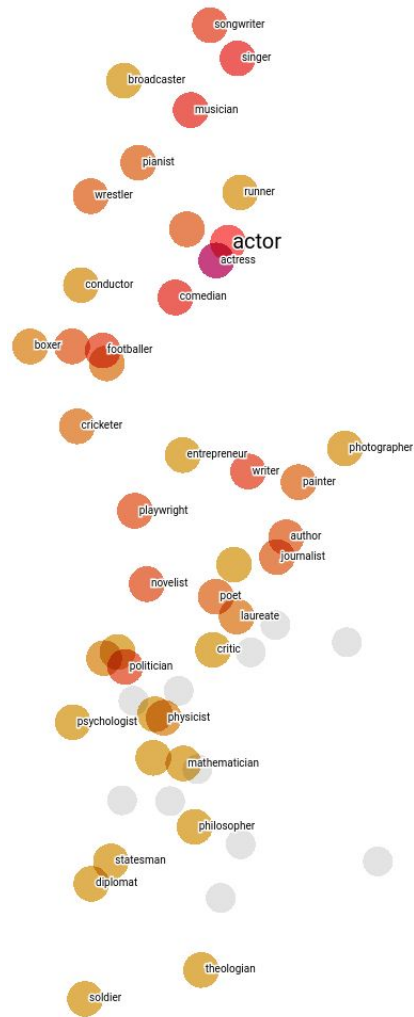
Search  by

neighbors

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

# Limits of supervised learning



Annotating data is hard, expensive, time-consuming (but necessary!)

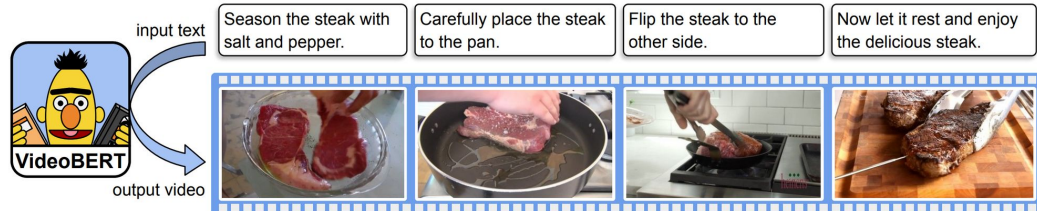
→ Limited amount of annotated data, many proposed solutions e.g.:

- distant learning: for example use emojis as annotation of sentiments in Tweets = noisy labels, but free annotation
- transfer learning: for example, with multitask learning, transfer knowledge from varied related tasks
- domain adaptation:
  - build a model on a large set of available data in one domain
  - optimize on a small set of annotated data for a target domain
- semi-supervised learning: for example, use word embeddings built on a massive amount of unannotated data to feed your neural network

# Current challenges in NLP

- Low-resourced languages / domains
- Multi-modality:
  - mostly work on monologues, but increasing work on dialogues / spoken language
  - 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

- Interpretability / explainability → very active domain
  - Saliency Map / Allen Interpret (Wallace et al., 2019) <https://allennlp.org/interpre>

# Current challenges in NLP

- Reproducibility, data sharing
- Cost of training very large models
  - [Energy and Policy Considerations for Deep Learning in NLP](#)
  - [On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?](#)
- 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](#)

Consumption	CO <sub>2</sub> e (lbs)
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

Finnish ↔ English

Hän sijoittaa. Hän pesee pyykkiä. Hän urheilee. Hän hoitaa lapsia. Hän tekee töitä. Hän tanssii. Hän ajaa autoa. × He invests. She washes the laundry. He's playing sports. She takes care of the children. He works. She dances. He drives a car.





# Practical session

1. Spacy: pre-processing and analysing text data
2. Scikit: sentiment analysis, classifying movie reviews
3. Word2vec: generating word embeddings
4. (Advanced) PyTorch: sentiment analysis using NN

# NLP courses, slides, interesting websites (and sources)

- [https://dair.ai/notebooks/nlp/2020/03/19/nlp\\_basics\\_tokenization\\_segmentation.html](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 (Courses in M2 Info, UPS and M1 Droit, UT3)

# 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](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](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).

# Sources of pictures

- <https://news.sky.com/story/talking-robots-could-be-used-in-uk-care-homes-to-ease-loneliness-and-improve-mental-health-12066296>
- <https://blog.lecko.fr/machine-learning/ordinateur-a-t-il-vraiment-chesoin-de-nous/>
- [https://fr.m.wikipedia.org/wiki/Fichier:Major\\_levels\\_of\\_linguistic\\_structure.svg](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-953d45388f24/>
- <https://www.youtube.com/watch?v=CeuhQ3s-Iss>
- <https://analytic.sindlamag.com/10-most-used-databases-by-developers-in-2020/>
- <https://www.erdl.fr/blog/2017/09/26/traitement-automatique-langues-erdl/>
- <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/yoavgo/status/1318576449985662977/photo/1>
- <http://jetpaper.web.fc2.com/reviews/page-5462805.html>
- <https://www.sjl.com/?detailStory=can-we-talk-librarians-lead-new-push-civics-education-focusing-discourse>
- <https://www.toptal.com/deep-learning/4-sentiment-analysis-accuracy-traps>
- <https://nativeenglishspain.blogspot.com/2013/10/wise-wednesday-grammar-homonym.html>
- <https://www.altcourse.com/training/in-service/lexicogrammar/polysemy.html>
- <https://medium.com/@vimagunturi789/everything-you-need-to-know-about-named-entity-recognition-2a136f38c08f>
- <https://active wizards.com/blog/top-nlp-algorithms-and-concepts/>
- <https://medium.com/@laura.mondoloni/ironback-sketch-exercise-kata-rocket-9ee72879d0d9>
- <https://www.youtube.com/watch?v=hTlFghkLdHhttps://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-330102.htm>
- <https://developers.google.com/machine-learning/guides/text-classification>
- <https://medium.com/mosaix/deep-text-representation-for-sequence-labeling-272e605ed9d>
- <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](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://jsuliv.github.io/perceptron/>