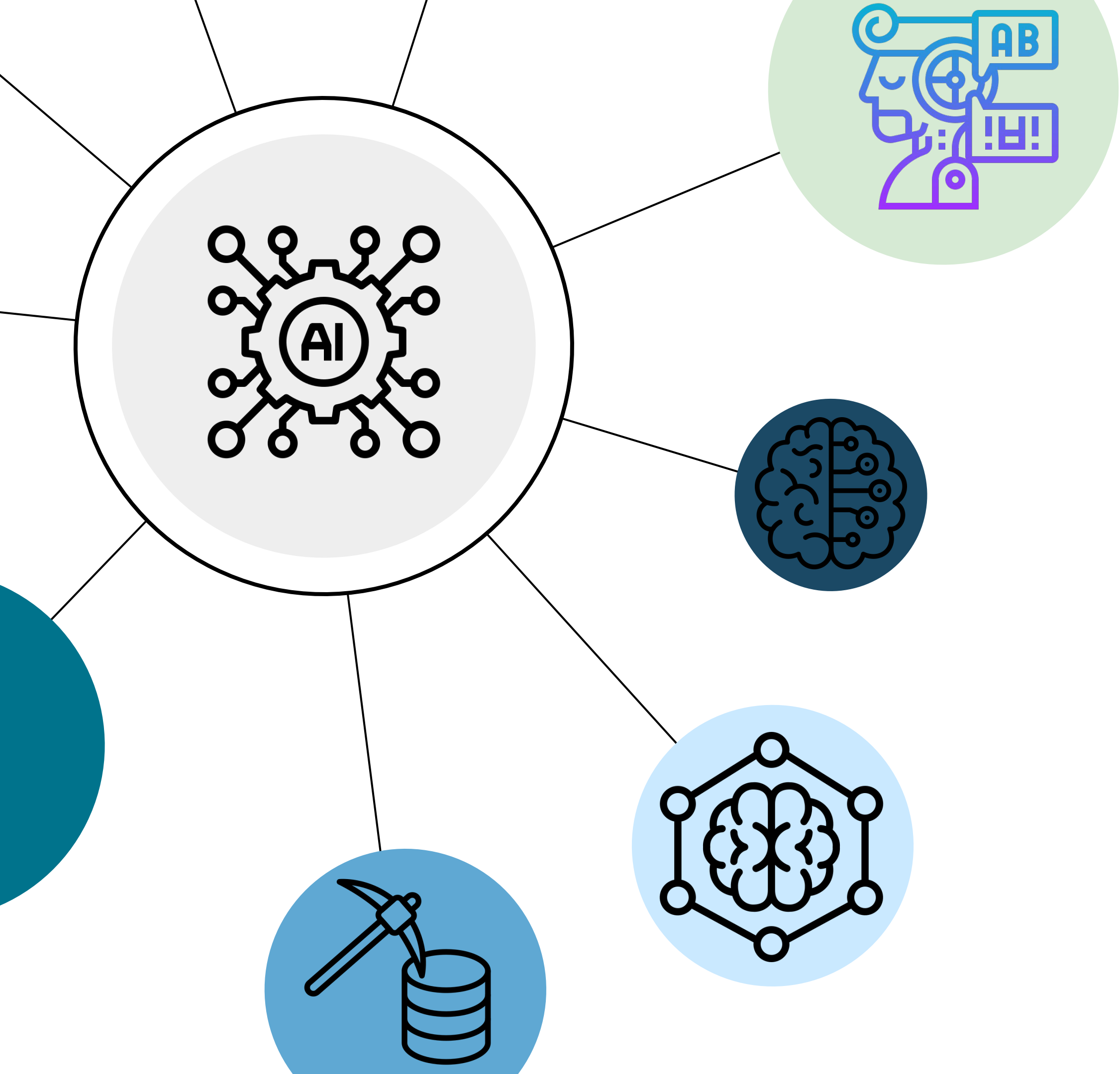


3iA Côte d’Azur
Institut interdisciplinaire d’intelligence artificielle

UNIVERSITÉ CÔTE D’AZUR

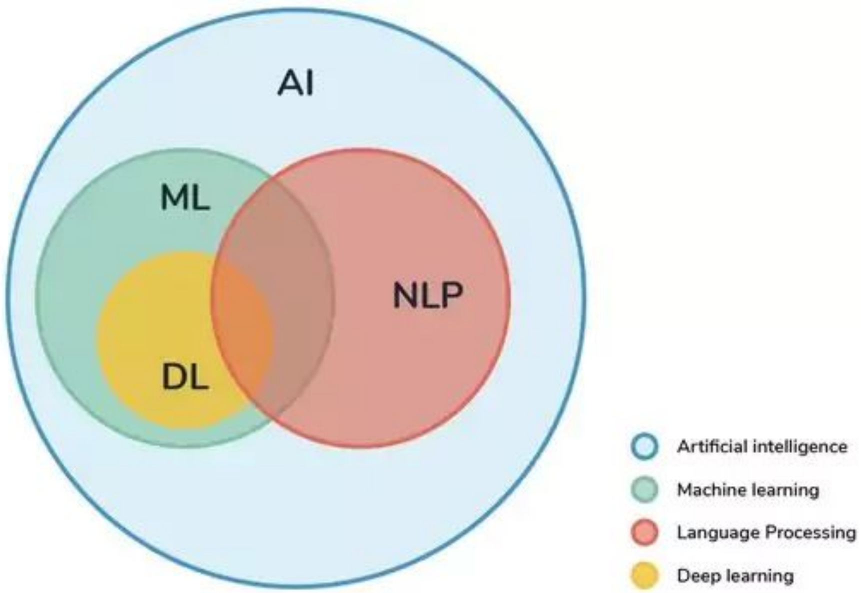
Introduction to NLP

Anaïs Ollagnier
Université Côte d’Azur, CNRS, Inria, I3S

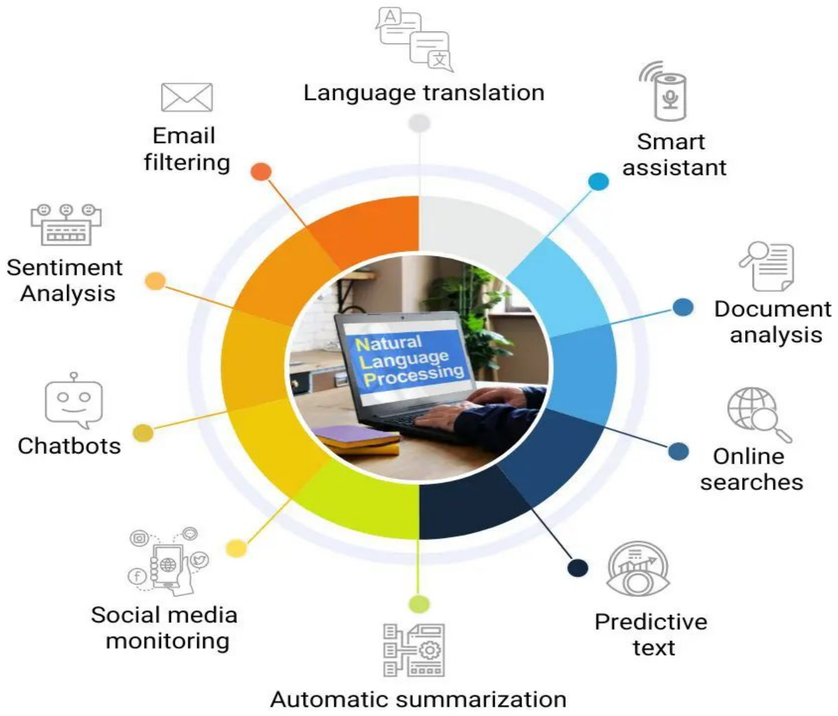
What's NLP ?

What's NLP ?

- NLP, which stands for Natural Language Processing or Traitement du Langage Naturel in French, is a discipline primarily concerned with machines understanding, manipulating, and generating natural language.



What issues does NLP address?



Commercial Applications

- Grammarly, Microsoft Word, Google Docs
- Search engines like DuckDuckGo, Google
- Voice assistants – Alexa, Siri
- News Feeds- Facebook, Google News
- Translation systems – Google translate

NLP objectives

Create a **computer model** that represents how language is used by humans:

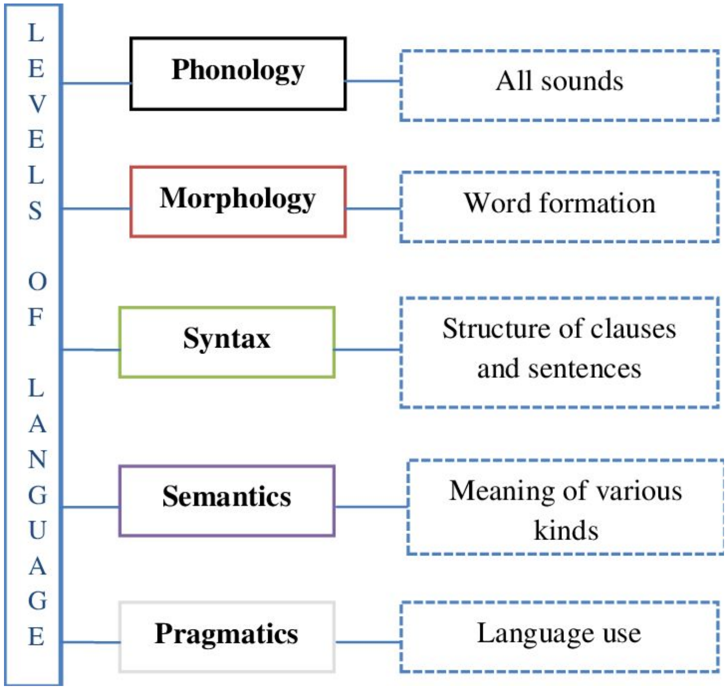
- Study how to formalize our knowledge about a language = make it unambiguous by using an "artificial" language.

- Ex :
    - “"The bank is on the river."” (lexical ambiguity)
    - “"I saw the man with the telescope."” (syntactic ambiguity)

- Better **understand the functioning of language structures**, the mechanisms through which humans learn, produce language, and comprehend it.

Levels of Language Understanding

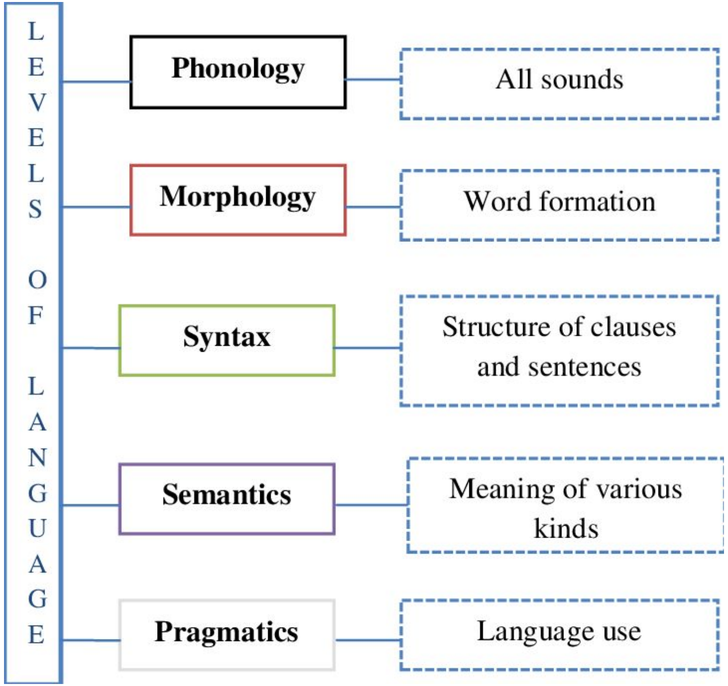
- Handling natural language requires the analysis of different levels of understanding/competence:



two, to, too, …

Levels of Language Understanding

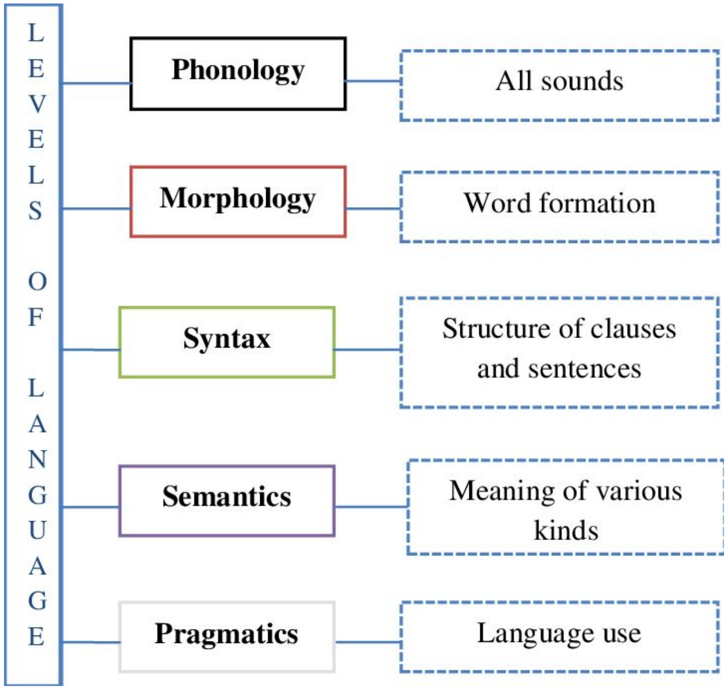
- Handling natural language requires the analysis of different levels of understanding/competence:



Joué, jouais, jouaient …

Levels of Language Understanding

- Handling natural language requires the analysis of different levels of understanding/competence:

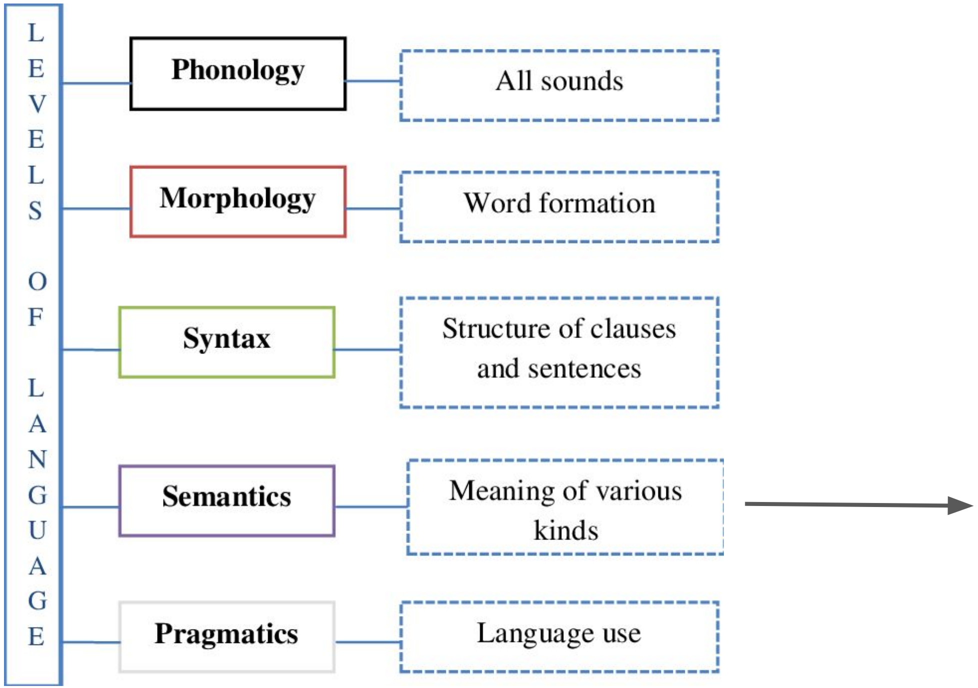


Le chien a mordu l'enfant,
l'enfant a mordu le chien

** Mordu a enfant le chien*

Levels of Language Understanding

- Handling natural language requires the analysis of different levels of understanding/competence:

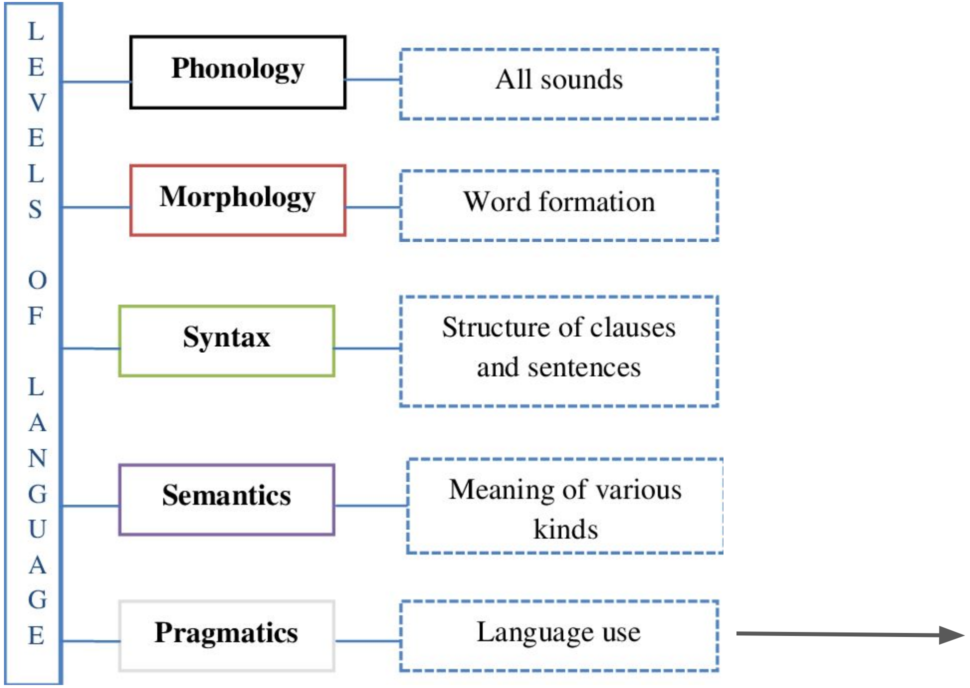


La gorge brûle, la maison brûle, la soupe brûle

L'avocat est mûr

Levels of Language Understanding

- Handling natural language requires the analysis of different levels of understanding/competence:

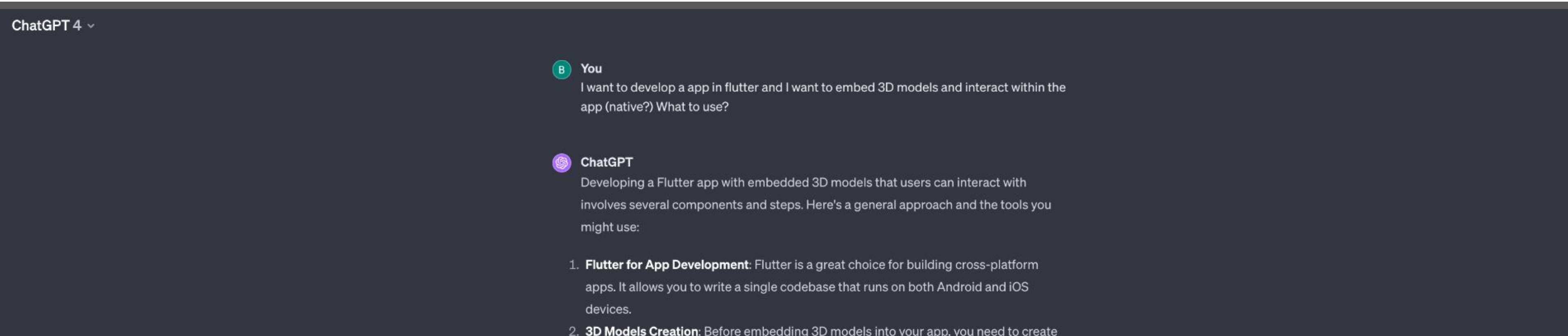


Passe moi la ! [Qui ou quoi ?]
Jean a fini son livre. [de le lire ou de l'écrire ?]

The application tasks

- Analysis
- Interaction
- Generation
- Transformation

Analysis	Generation	Transformation	Interaction
Spelling correction	“Data to Text”	Translation	Question-aswering
Hate speech detection	Image captioning	Summarization	Specialized Chatbots
Information Extraction	Video summarization	Text simplification	General-purpose Chatbots



The fundamental tasks

1/ Analyse morpho-syntaxique

Morphologie en linguistique

- Domaine qui traite de la structure interne des mots
- **Linguistique structurale** :
  - Notion de morphème = unité linguistique minimale (i.e. non décomposable) porteuse de sens
    - Unités abstraites
  - Notion de morphe = (une) forme graphique d'un morphème
    - Allomorphes : variantes d'un même morphème
      - Libres : assois/assieds
      - Contextuelles : j'/je

1/ Analyse morpho-syntaxique

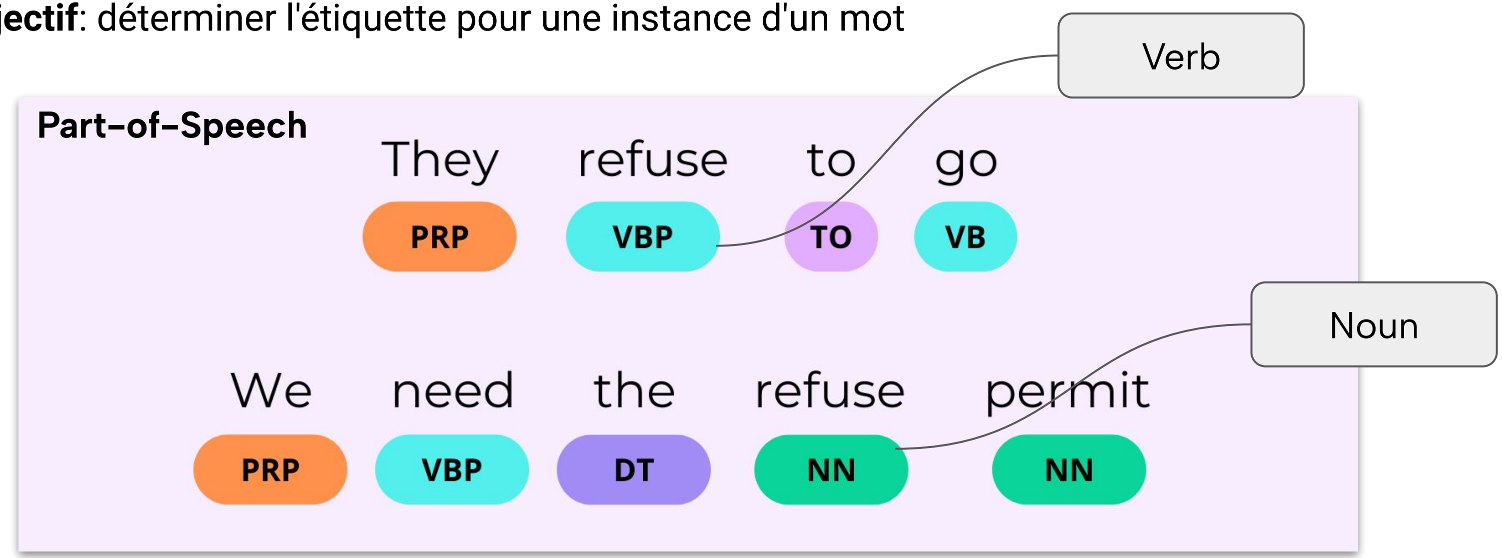
- **Procédés morphologiques** :
    - *Flexion* : déclinaison, conjugaison
        - Grand/grands/grande, cours/courir
    - *Dérivation* : formation de nouveaux mots notamment par adjonction d'affixes au radical
        - Anti-constitu-tionn-elle-ment
    - *Composition* : combinaison de plusieurs bases pour former un nouveau mot
        - Tournevis

1/ Analyse morphologique

- **Racinisation (stemming)**
  - But : supprimer la terminaison des mots
    - Conjugaison/conjuguer → conjug
  - Très utilisé en recherche d'information
- **Lemmatisation**
  - But : ramener les variantes flexionnelles d'un même mot à sa forme canonique, le lemme
    - Conjugue/conjuger/conjugué → conjuguer
- **Décomposition**
  - But : segmenter un mot contenant plusieurs autres mots afin de retrouver ses composants
    - Surtout utilisé dans des langues comme l'allemand

2/ Etiquetage morpho-syntaxiques

- Les mots ont généralement plus d'une étiquette possible
    - Le bois vient de France. → le=det, bois=nom
    - Je le bois. → le = pronom, bois = verbe
- **Objectif**: déterminer l'étiquette pour une instance d'un mot



2/ Etiquetage morpho-syntaxiques

Exemple de difficultés

Entrée: “Le débat est relancé.”

- Ambiguités: le=det/pro débat=verbe/nom est=verbe/nom

Sortie: Le/DET débat/NOM est/VER relancé/VER

Applications:

- Synthèse vocale: comment prononcer **est** ?
- Recherche dans un corpus: **est** en tant que *nom*
- Entrée d'un analyseur syntaxique

3/ Analyse syntaxique

Définition:

- L'analyse syntaxique, ou parsing, est l'étape du NLP qui consiste à analyser la structure grammaticale des phrases.
- Elle permet de déterminer la fonction grammaticale de chaque mot dans une phrase et de comprendre la structure hiérarchique de celle-ci.

Importance en NLP:

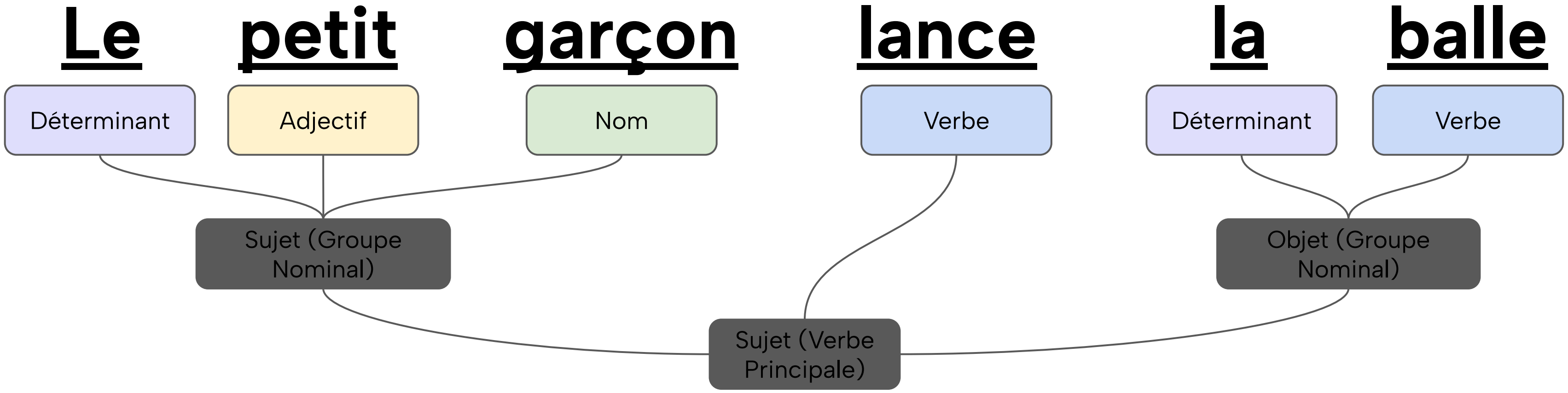
- Cruciale pour la compréhension du sens des phrases.
- Permet des applications comme la *traduction automatique*, la *reconnaissance vocale*, et la *génération de texte*.

Exemple de processus

3/ Analyse syntaxique

Phrase: “Le petit garçon lance la balle”

1. Segmentation en mots (Tokenization)
2. Identification des parties du discours (Part-of-Speech Tagging)
3. Construction d'un arbre syntaxique



3/ Analyse syntaxique

1. Analyse Syntaxique Ascendante (Bottom-up Parsing) :

- **Principe** : Cette méthode démarre par les *symboles d'entrée* (les mots de la phrase) et construit progressivement l'arbre de parsing en remontant vers le symbole de départ (souvent la racine de la phrase).
- **Fonctionnement** : Elle *identifie les unités les plus petites* (comme les mots ou les tokens) et les *regroupe en unités plus grandes* (comme les phrases ou les clauses) selon les règles grammaticales.
- **Exemple** : Le "Shift-reduce Parser" est un exemple classique. Il fonctionne en déplaçant les symboles d'entrée (shift) dans une structure de données temporaire et en les réduisant (reduce) en fonction des règles grammaticales pour former des unités plus grandes.

3/ Analyse syntaxique

2. Analyse Syntaxique Descendante (Top-down Parsing) :

- **Principe** : Cette approche commence par le *symbole de départ de la grammaire* (généralement la racine de l'arbre de parsing) et tente de *décomposer ce symbole en symboles d'entrée* en suivant les règles grammaticales.
- **Fonctionnement** : Elle utilise une *approche récursive* pour décomposer les unités grammaticales plus grandes en leurs composants plus petits jusqu'à ce qu'elle atteigne les symboles d'entrée.
- **Exemple** : Le "Recursive Descent Parser" est un exemple typique. Il utilise une série de fonctions récursives pour décomposer la structure grammaticale d'une phrase, chaque fonction traitant un élément spécifique de la grammaire.

3/ Analyse syntaxique

3. Analyse Dépendancielle vs Constituante :

Analyse Dépendancielle :

- **Fonctionnement** : Se concentre sur les relations de dépendance entre les mots dans une phrase. Chaque mot est considéré comme dépendant d'un autre, formant une structure en réseau.
- **Utilisation** : Permet de comprendre comment les mots se rapportent les uns aux autres, indépendamment de leur structure hiérarchique.

Analyse Constituante :

- **Fonctionnement** : Construit une structure hiérarchique de la phrase, généralement sous forme d'arbres, où chaque unité (constituant) est une combinaison de mots ou de sous-unités plus petites.
- **Utilisation** : Utile pour analyser la structure grammaticale globale d'une phrase et pour comprendre comment les groupes de mots fonctionnent ensemble comme des unités.

Les tâches ok, mais comment?

Symbolic Approaches

(?i)nlp?

Regular expression

A formal language to **specify text strings**

How can we search for any of these words?

- marmotte
- marmottes
- Marmotte
- Marmottes

Regular expressions allow searching for a pattern in a corpus,

meaning finding occurrences of words with certain characteristics

in textual documents.

Developed in 1956 by Steve Kleene as an algebraic notation for representing strings of characters.

Applications:

- Performing text searches that are more flexible than normal word searches
- Substituting words or character strings in text
- Formalizing morphological rules
- Formalizing syntactic rules

Regular expression: Terminology

Language: A set of character strings. A regular expression denotes a language.

String: Concatenation of zero or more symbols.

Regular expressions also include **operators** that apply to character classes (e.g., intersection, union, difference) or strings (concatenation).

Regular expression: Syntaxe

/chat/

Find all the occurrences of the string “chat”

Ex. “Le **chat** dort” as well as “Teo est un beau **chat**on”

If we do not specify any additional parameters, regular expressions are "**case-sensitive**".

We would not find "**Chat** noir" because it starts with an uppercase letter.

Regular expression: Disjunctions

Lettres entre crochets[]

Patterns	Matches
[mM]armotte	marmotte, Marmotte
[1234567890]	N'importe quel chiffre

Classes [A-Z]

Patterns	Matches	Exemple
[a–z]	Une lettre minuscule	<u>m</u>on chat
[A–Z]	Une lettre majuscule	le <u>C</u>hat
[0–9]	Un seul chiffre	Chapitre <u>1</u>:Intro

Regular expression

Negation

in regular expressions is denoted by the caret symbol **"^"** inside square brackets. It specifies that the pattern should not match the characters listed within the brackets. For example:

- /[^Ll]ait/ matches "fait" but not "lait".
- /[^A-Z]/ matches any character that is not an uppercase letter.
- /[^,] / matches any character that is not a comma followed by a space.

Option

"?" indicates that the preceding character or group is optional, meaning it may appear once or not at all. For instance:

- /chats?/ matches both "chat" and "chats".
- /colou?r/ matches both "color" and "colour".

Regular expression: Multiplicators

Character Multipliers: **Kleene *** This multiplier specifies zero or more occurrences of the immediately preceding character (or regular expression).

Examples:

- **/demain*/**: Finds "demai", "demainn", "demainnn", and "demainnnn".
- **/[abc]*defg/**: Finds "defg", "adefg", "bdefg", "abacdefg", etc.

The symbol **+** This multiplier specifies one or more occurrences of the immediately preceding character (or regular expression).

Examples:

- - /demain+/ : Finds "demain", "demainn", "demainnn".
- - /[0-9]+/ : Looks for at least one digit between 0 and 9.
- - /bu+/ : Recognizes "bu", "buu", "buuu" but not "b".

Regular expression: Joker

The "joker": **. (point)**

/./ = finds characters except newline

(if we want to find the period instead, /\./)

/cha./ = chat, chat, chats, char, ..

/.+/ = at least one occurrence of any character

Regular expression: Anchors

Special characters signaling a specific position in the searched document:

^ = Start of line

- /^ The '[a-z]+/
- /^ 1900/

$ = End of line

- /Route$ /

The word boundary delimiter **\b**

For example, /\ble\b/ = finds "le" as an article but not in "pilule"

The symbol **\B** indicates all characters that do not delimit a word

Ex. /\Ble\B/ = matches in "pilules" but not in "le"

Regular Expression: Operators

| **(pipe)**: OR operator, indicates an alternative between strings

/chat|chien/ = chat or chien

Using **parentheses**, you can define alternative string portions

/chat(s|on|ons)/ = chat, chaton, chatons

To multiply strings /(la)+/ = la or lalalala …

What would produce the same regular expression without parentheses? /la+/

To group disjointed strings

/chats|ons/ = chats and ons

/chat(s|ons)/ = chats and chatons

Regular Expression: Sum up

Regular expressions play a very important role.

Sophisticated sequences of regular expressions are **often the first model for any text processing**.

For many challenging tasks, we use machine learning classifiers.

But regular expressions are used in classifiers.

Very useful for **capturing generalizations**.

```
Welcome to
                    EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                    EE      LL       II       ZZ   AA   AA
                    EEEEE   LL       II     ZZZ    AAAAAAA
                    EE      LL       II    ZZ      AA   AA
                    EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

https://en.wikipedia.org/wiki/ELIZA

Response Patterns: ELIZA

```
User1: Men are all alike.
ELIZA1: IN WHAT WAY

User2: They're always bugging us about something or other.
ELIZA2: CAN YOU THINK OF A SPECIFIC EXAMPLE

User3: Well, my boyfriend made me come here.
ELIZA3: YOUR BOYFRIEND MADE YOU COME HERE

User4: He says I'm depressed much of the time.
```

```
s/.* I'M (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/.* I AM (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
```

Response Patterns: ELIZA

```
User1: Men are all alike.
ELIZA1: IN WHAT WAY

User2: They're always bugging us about something or other.
ELIZA2: s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/

User3: Well, my boyfriend made me come here.
ELIZA3: YOUR BOYFRIEND MADE YOU COME HERE

User4: He says I'm depressed much of the time.
ELIZA4: I AM SORRY TO HEAR YOU ARE DEPRESSED
```

Words, tokens, forms, lemmas...

Words, tokens, forms, lemmas...

“*Logical*” units for text processing:

Document ⊃ paragraph ⊃ sentence ⊃ "word" ⊃ "character"

But a “*word*” is not a well-defined unit:

Examples: airplane, eaten, very, Robert, SNCF, 42...

Words, tokens, forms, lemmas...

Form: graphical representation of a word (Igor Mel'čuk)

Lemma: intersection between a form (graphical) and a meaning, sometimes by morpheme composition (find, a, cat)

Morpheme: smallest unit carrying meaning (e.g., "re")

Token: minimal unit of information detected during lexical analysis or tokenization – Often referred to as "lexeme" in French

Tokenization

Token – the smallest unit of information detected during "lexical analysis"

Tokenization – segmenting a text into "minimal units" for processing

Example:

Les étudiants, ceux du M2, n'ont-ils pas tous 15,3 de moyenne ?

Les | étudiants | , | ceux | du | M2 | , | n' | ont | -ils | pas | tous | 15,3 | de | moyenne | ?

Set of automata that recognize tokens by accepting strings, possibly by typing them

Lexeme: -?[A-Z]?[a-z]*

Punctuation: .|...|,|!|?

Number: -?[0-9]*(,|.)[0-9]*

Tokenization: issues

- French: ***L'ensemble*** ➢ one token or two? L?L'?Le?

  We want ***l'ensemble*** to match with ***un ensemble***

- Compound expressions in german : ***Lebensversicherungsgesellschaft Angestellter*** ➢ 'employee of the life insurance'

- Chinese and Japanese, no space between words: 莎拉波娃现在居住在美国东南部的佛罗里达莎拉波娃现在 居住在 美国 东南部 的 佛罗里达 ➢ Sharapova now lives in the southeastern United States, in Florida.

Lemmatization

Lemma - An autonomous unit (composed of morphemes) used to build the lexicon of a language.

- **Morphemes**: the "parts" of the lemma.

- **Autonomous**: can be used as is in a sentence.

Lemmatization - Finding the lemmas for each token within a sentence.

Example:

I carry potatoes

I | carry | a | potato

Stemming

Stemming - reducing terms to their roots during information retrieval

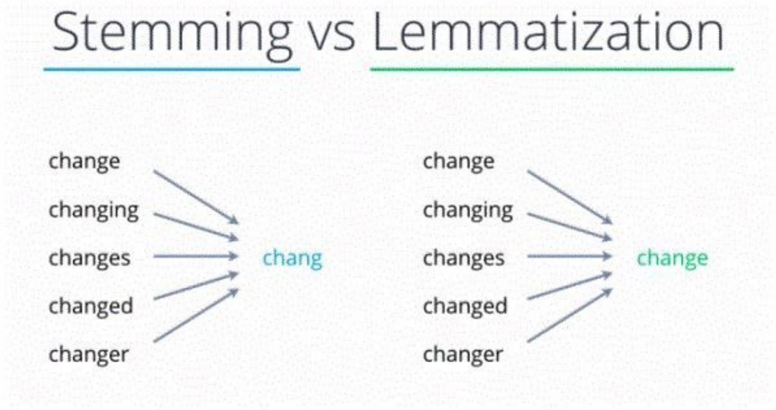
Root - the base/stem of a word carrying the meaning

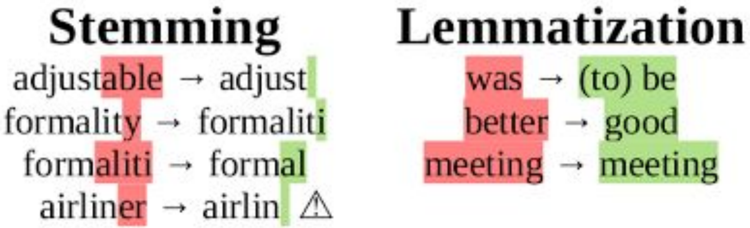
- The root of "déplaira" and "déplaisent" is "déplai"
- If "déplu" is added, the root becomes "dépl"
- With "plaira," the root is now "pl"

Conversely, mechanisms that construct possible forms from roots are inflection, derivation, composition

For example, "place" can yield: "places", "placerions", "déplacer", "déplacements", "remplaceras", etc.

Stemming / Lemmatization





Normalization

Need to "normalize" terms: for example, indexed text and query terms should have the same form (e.g., PV and P.V.)

We implicitly define equivalence classes of terms: for example, by removing periods in a term (e.g., M.)

Alternative: asymmetrical expansion:

• Input: window ➢ Search: window, windows

• Input: windows ➢ Search: Windows, windows, window

• Input: Windows ➢ Search: Windows

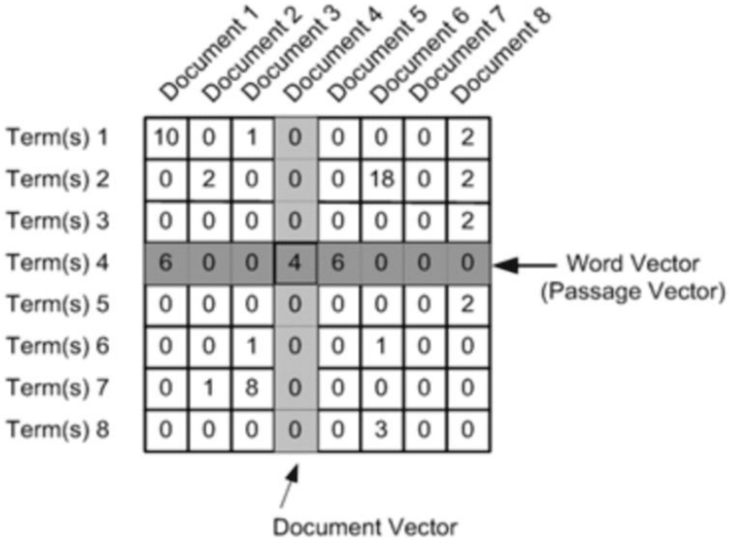
Change of case: convert all letters to lowercase

Automatic Processing

Vectorization

Convert Text Data into Numerical Values

Term-Frequency (TF): This method involves counting the number of occurrences of tokens present in the corpus for each text. Each text is then represented by a vector of occurrences. This is commonly referred to as Bag-of-Words, or "sac de mots" in French.



Convert Text Data into Numerical Values

- *Term Frequency-Inverse Document Frequency (TF-IDF)* : This method involves counting the number of occurrences of tokens present in the corpus for each text, which is then divided by the total occurrences of these same tokens in the entire corpus.

$$\underbrace{TFIDF_{t,d,D}}_{\substack{\text{Importance d'un terme} \\ \text{t dans un document d}}} = \underbrace{TF_{t,d}}_{\substack{\text{Fréquence d'un terme} \\ \text{t dans un document d}}} \times \underbrace{IDF_{t,D}}_{\substack{\text{Importance du terme} \\ \text{t dans l'ensemble des} \\ \text{documents D}}}$$

Convert Text Data into Numerical Values

```
corpus = [
   'This is the first document.',
   'This is the second second document.',
   'And the third one.',
   'Is this the first document?',
]
```

and	document	first	is	one	second	the	third	this
0	1	1	1	0	0	1	0	1
0	1	0	1	0	2	1	0	1
1	0	0	0	1	0	1	1	0
0	1	1	1	0	0	1	0	1

Convert Text Data into Numerical Values

	and	document	first	is	one	second	the	third	this
df(t)	1	3	2	3	1	1	4	1	3
idf(t)	2.39	1.29	1.69	1.29	2.39	2.39	1.00	2.39	1.29

and	document	first	is	one	second	the	third	this
0.00	1.29	1.69	1.29	0.00	0.00	1.00	0.00	1.29
0.00	1.29	0.00	1.29	0.00	4.77	1.00	0.00	1.29
2.39	0.00	0.00	0.00	2.39	0.00	1.00	2.39	0.00
0.00	1.29	1.69	1.29	0.00	0.00	1.00	0.00	1.29

Limitations

The effectiveness of these methods varies depending on the application. However, they have two main limitations:

- The richer the vocabulary of the corpus, the **larger the size of the vectors**, which can be a **problem for learning models**.

- **Word occurrence** counting does not account for the arrangement of words and thus the **meaning of sentences**.

Your turn !

Rdv on :

https://gist.github.com/rspeer/ef750e7e407e04894cb3b78a82d66aed

Exercises:

- Expression: "Chocolat Chaud" capture only the letter 'c' at the beginning of each word.
- Find the regular expression to capture an email address: xxxxxxxx@xxxxxxxx.xxx

(An email address will be deemed correct if it contains an "@" character and a "." character.)