



A large word cloud centered around the acronym NLP (Natural Language Processing). The words are arranged in a roughly circular pattern around the center, with larger words representing more frequent terms. The most prominent words include NLP, LANGUAGE, SYSTEMS, RULES, EVALUATION, and HUMAN.

The word cloud includes the following words:

- DESIGNERS MADE LEARNING BASED ACCURATE AUTOMATIC
- EVALUATED FEELINGS WORKED
- TASK HUMAN STANDARD INPUT DATA SYSTEM
- PROCESSING THIS CRITERIA
- RESULTS PREDICTIVE
- COMPLEXITY MODELS SIMPLY
- WITHOUT UNDERSTANDING ANSWERS
- RESEARCH GOLD
- EVALUATION OUTPUT
- COMPUTERS HAND-WRITTEN
- PROBLEM ALGORITHMS
- EVALUATION
- ONE MANY
- COMPUTER SOLUTIONS TAKE EFFECTIVE
- DIFFERENT
- APPLIED BEYOND
- DEFINITION
- REAL-WORLD
- CONSIDERABLE
- INCLUDED METRIC
- PRODUCING
- NUMBER LARGER
- ALTHOUGH SIGNIFICANT
- SOLVING COMPLEX
- AREA DEFINED
- APPLIED BEYOND
- DEFINITION
- MASSIVE
- APPLIED BEYOND
- ELEMENT
- APPLIED BEYOND
- ELEMENT

RÉSUMÉ THÉORIQUE – FILIÈRE INTELLIGENCE ARTIFICIELLE

M202 – Manipuler les moteurs NLP

SOMMAIRE



01 – Explorer les fondamentaux de NLP

S'initier à NLP

Pratiquer le prétraitement de NLP

02 – Maîtriser les méthodes quantitatives pour l'analyse textuelle

Pratiquer l'analyse syntaxique et sémantique

Pratiquer l'apprentissage approfondi

03 – Maîtriser le fonctionnement des transformateurs

Définir les fondements des transformateurs

Explorer les modèles de transformateurs classiques

Implémenter les modèles préentraînés

Explorer les LLMs

04 – Maîtriser le déploiement des transformateurs dans les moteurs NLP

Utiliser les plateformes de conception et d'intégration des modèles

Maîtriser les bases du cloud computing et de Dialogflow

Intégrer les modèles préentraînés

Optimiser les performances des modèles préentraînés

Approfondir les techniques de maintenance et de mise à jour des modèles



مدن المهن و الكفاءات
+٤٣٨٤٤١١٢٣٦٥٠
Cités des Métiers et des Compétences

Intro to Natural Language Processing (NLP)

01 – S'INITIER A NLP

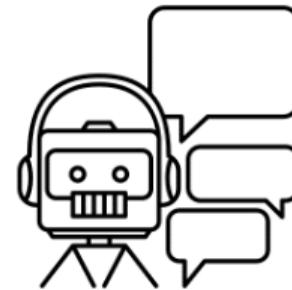
Généralités sur NLP

Définition de NLP

Le traitement automatique du langage naturel (NLP, pour *Natural Language Processing*) est une branche de l'intelligence artificielle et de la linguistique informatique . Il vise à permettre aux machines de comprendre, d'interpréter, de générer et d'interagir avec des données textuelles ou sonores en langage humain

Pour être précis, le NLP est un **domaine multidisciplinaire** qui repose sur l'intersection de plusieurs champs :

- **Linguistique** : Modélisation de la syntaxe, de la sémantique et de la pragmatique
- **Information** : Gestion et extraction des données textuelles, indexation, recherche d'informations
- **Intelligence artificielle** : Entraînement de modèles pour la reconnaissance et la génération de textes
- **Sciences sociales** : Compréhension des contextes culturels, étude des biais de langage, et analyse de comportements via les textes



Sooooooooooooo!!!!

By “natural language” we mean a language that is used for everyday communication by humans.

NLP is an Intersection of several fields

- Computer Science
- Artificial Intelligence
- Linguistics

It is basically teaching computers to process human language

Two main components:

- Natural Language Understanding (NLU)
- Natural Language Generation (NLG)

NLP is AI Complete

- Requires all types of knowledge humans possess → It's hard!

01 – S'INITIER A NLP

Généralités sur NLP

Le langage au cœur de la communication humaine

Le langage sert à :

- **Communiquer** : Le langage est le principal outil d'échange
- **Penser et structurer les idées** : Une grande partie des processus cognitifs passe par le langage
- **Raconter des histoires** : Le langage permet de transmettre des expériences et des récits
- **Construire des théories scientifiques** : Les concepts scientifiques sont formulés et diffusés à travers le langage
- **Créer du lien social** : Le langage est essentiel pour nouer des amitiés et entretenir des relations humaines



Natural Language Understanding (NLU)

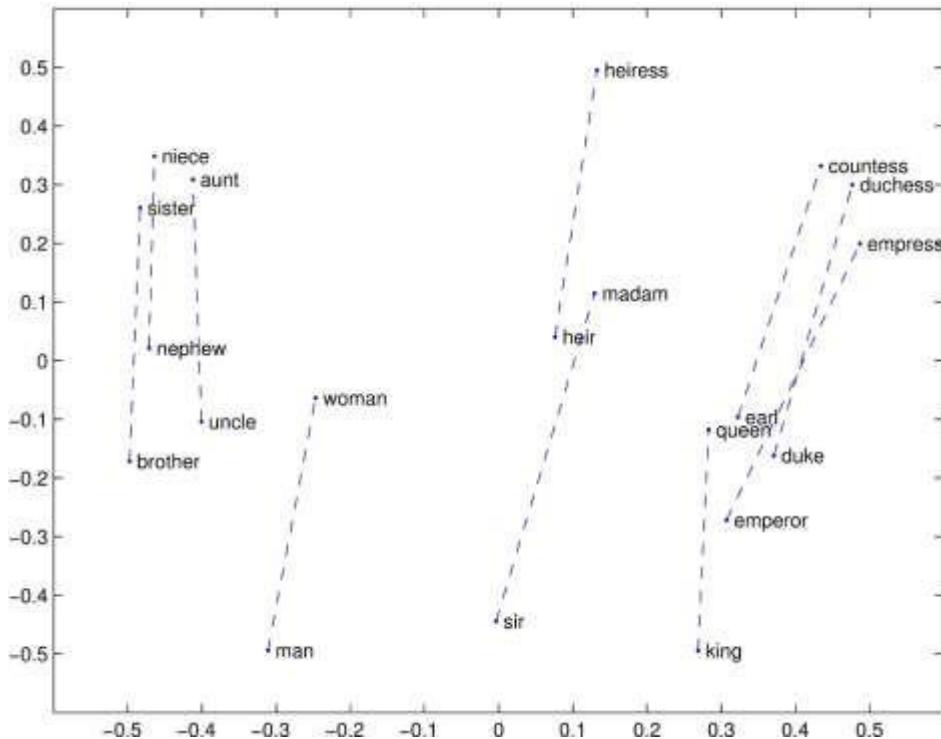
- Deriving meaning from natural language
- Imagine a Concept (aka Semantic or Representation) space
 - In it, any idea/word/concept has unique computer representation
 - Usually via a vector space
 - NLU → Mapping language into this space

Exemple simple :

Les mots "roi", "reine", "homme", "femme" peuvent être représentés par des vecteurs.

Dans cet espace vectoriel, on observe souvent des relations :

- **roi – homme + femme ≈ reine**



Natural Language Generation (NLG)

- Mapping from computer representation space to language space
- Opposite direction of NLU
 - Usually need NLU to perform NLG!
- NLG is really hard!

Exemple :

Représentation interne : {'action': 'manger', 'objet': 'pomme', 'temps': 'matin'}

Sortie NLG : "J'ai mangé une pomme ce matin."



History of NLP

- NLP has been through (at least) 3 major eras:
 - 1950s-1980s: Linguistics Methods and Handwritten Rules
 - 1980s-Now: Corpus/Statistical Methods
 - Now-???: Deep Learning
- Lucky you! You're right near the start of a paradigm shift!

1950s - 1980s: Linguistics/Rule Systems

- NLP systems focus on:
 - Linguistics: Grammar rules, sentence structure parsing, etc
 - Handwritten Rules: Huge sets of logical (if/else) statements
 - Ontologies: Manually created (domain-specific!) knowledge bases to augment rules above
- Problems:
 - Too complex to maintain
 - Can't scale!
 - Can't generalize!

1980s - Now: Corpus/Statistical Methods

- NLP starts using Machine Learning methods
- Use statistical learning over huge datasets of unstructured text
 - Corpus: Collection of text documents
 - e.g. Supervised Learning: Machine Translation
 - e.g. Unsupervised Learning: Deriving Word "Meanings" (vectors)

Now - ???: Deep Learning

- Deep Learning made its name with Images first
- 2012: Deep Learning has major NLP breakthroughs
 - Researchers use a neural network to win the Large Scale Visual Recognition Challenge (LSVRC)
 - Le Deep Learning, d'abord révolutionnaire pour les images, a ensuite permis des avancées majeures en NLP, ouvrant la voie à des systèmes capables de traiter efficacement le langage, les images, ou les deux simultanément.
- Very useful for unified processing of Language + Images

NLP Definitions

- Phonemes: the smallest *sound* units in a language
- Morphemes: the smallest units of *meaning* in a language
- Syntax: how words and sentences are constructed from these two building blocks
- Semantics: the *meaning* of those words and sentences
- Discourse: semantics *in context*. Conversation, persuasive writing, etc.

Un morphème = la plus petite unité qui porte un sens dans une langue.

• Exemple 1 : le mot "écrivons"

- écriv- = morphème (racine → l'action d'écrire)
- -ons = morphème (1^{re} personne du pluriel, conjugaison)

• Exemple 2 : le mot "invisible"

- in- = morphème (préfixe de négation)
- vis- = morphème (racine liée à « voir »)
- -ible = morphème (suffixe qui signifie « qui peut être »)

• Un phonème = la plus petite unité sonore qui permet de distinguer les mots.

• Exemple 1 : "balle" vs "palle"

- /b/ et /p/ sont deux phonèmes différents.

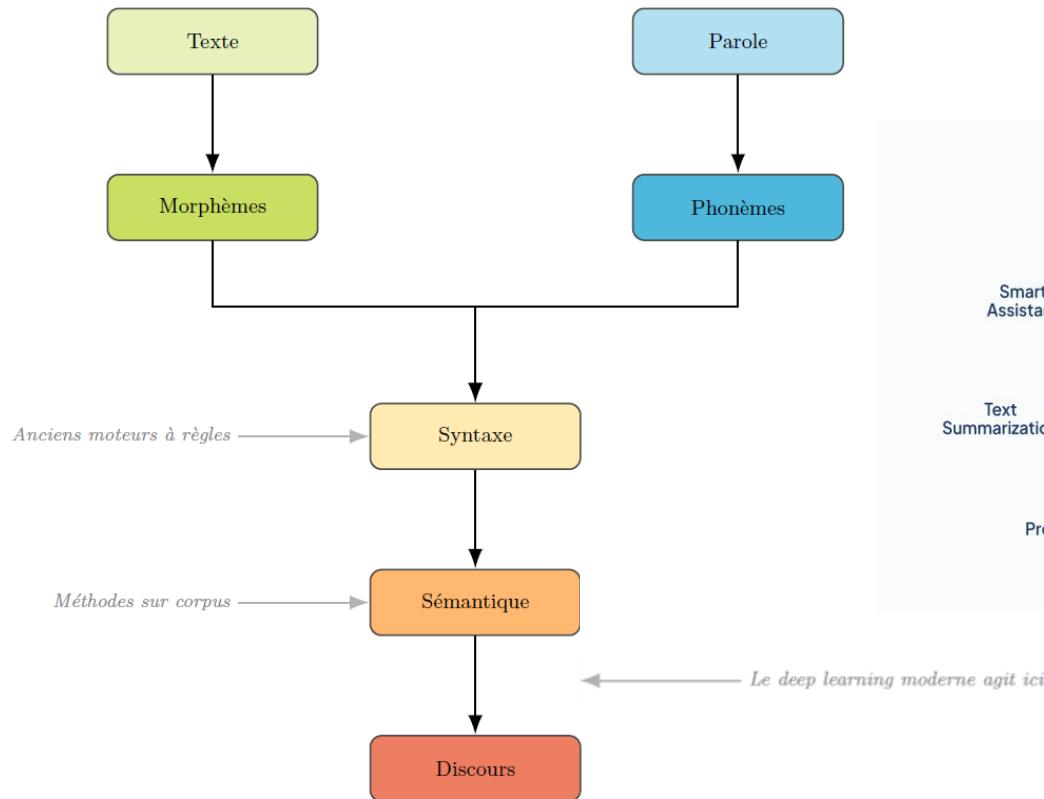
- Le changement d'un seul phonème change complètement le sens du mot.

• Exemple 2 : le mot "chat"

- Phonèmes : /ʃ/ – /a/ – /t/

- Si on remplace /ʃ/ par /r/, on obtient "rat", avec un autre sens.

Niveaux du TALN



NLU Applications

- ML on Text (Classification, Regression, Clustering)
- Document Recommendation
- Language Identification
- Natural Language Search
- Sentiment Analysis
- Text Summarization
- Extracting Word/Document Meaning (vectors)
- Relationship Extraction
- Topic Modeling
- ...and more!

NLU Application: Document Classification

- Classify “documents” - discrete collections of text - into categories
 - Example: classify emails as spam vs. not spam
 - Example: classify movie reviews as positive vs. negative
 - Example: classify legal documents as relevant vs. not relevant to a topic

NLU Application: Document Recommendation

- Choosing the most relevant document based on some information:
 - Example: show most relevant webpages based on query to search engine
 - Example: recommend news articles based on past articles liked
 - Example: recommend restaurants based on Yelp reviews

NLU Application: Topic Modeling

- **Breaking a set of documents into topics at the word level**
 - Example: see how prevalence of certain topics covered in a magazine changes over time
 - Example: find documents belonging to a certain topic

NLG Applications

- Image Captioning
- (Better) Text Summarization
- Machine Translation
- Question Answering/Chatbots
- ...so much more
- Notice NLU is almost a prerequisite for NLG

NLG Application: Image Captioning

- Automatically generate captions for images



man in black shirt is playing guitar.



construction worker in orange safety vest is working on road.



two young girls are playing with lego toy.



boy is doing backflip on wakeboard.

Captions automatically generated.

Source: <https://cs.stanford.edu/people/karpathy/cvpr2015.pdf>

NLG Application: Machine Translation

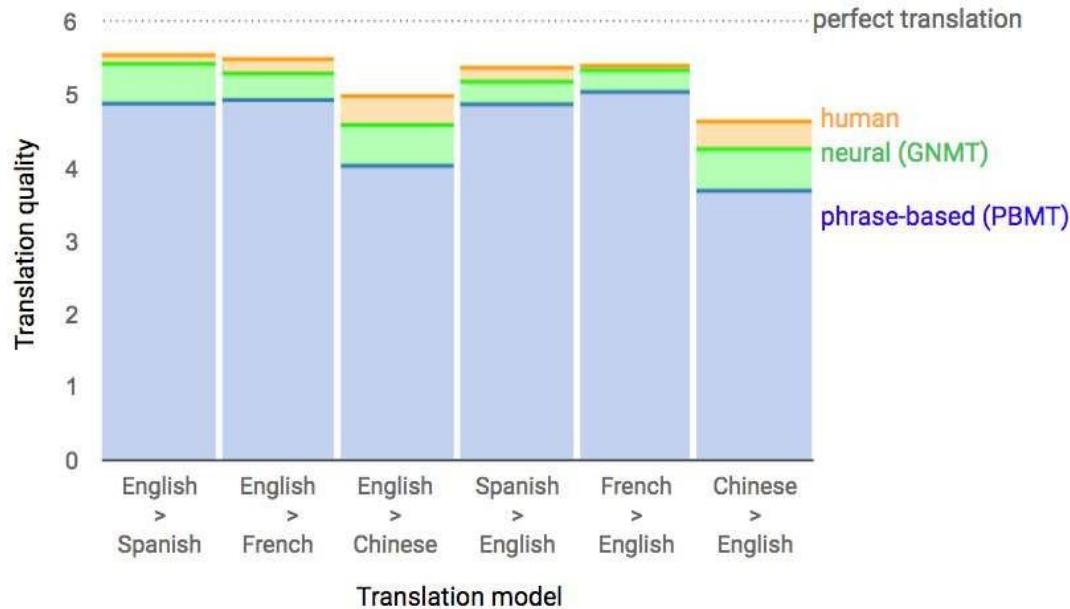
- Automatically translate text between language

<i>Input sentence:</i>	<i>Translation (PBMT):</i>	<i>Translation (GNMT):</i>	<i>Translation (human):</i>
李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。	Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.	Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.	Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.

Example from Google®'s machine translation system (2016)

Source: <https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html>

NLG Application: Machine Translation



Source: <https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html>

NLG Application: Text Summarization

- Automatically generate text summaries of documents
 - Example: generate headlines of news articles

Input: Article 1st sentence	Model-written headline
metro-goldwyn-mayer reported a third-quarter net loss of dlr\$ 16 million due mainly to the effect of accounting rules adopted this year	mgm reports 16 million net loss on higher revenue
starting from july 1, the island province of hainan in southern china will implement strict market access control on all incoming livestock and animal products to prevent the possible spread of epidemic diseases	hainan to curb spread of diseases
australian wine exports hit a record 52.1 million liters worth 260 million dollars (143 million us) in september, the government statistics office reported on monday	australian wine exports hit record high in september

Source: <https://ai.googleblog.com/2016/08/text-summarization-with-tensorflow.html>

Regular Expressions

What are Regular Expressions (“RegEx”)?

Une expression régulière décrit un motif à rechercher dans un texte

Exemples :

Une chaîne contient-elle le mot "chat" ?

Une chaîne contient-elle 3 lettres puis 2 chiffres suivis d'un espace ?

Une chaîne contient-elle une ou plusieurs lettres (et uniquement des lettres) ?

Utilisation typique :

Trouver les chaînes qui contiennent le motif de l'expression régulière

Récupérer la partie d'une chaîne correspondant au motif

Use Cases for Regex

- **Analyse de documents avec une structure de composants attendue**

Utiliser les expressions régulières pour extraire les éléments souhaités

Exemple : en HTML, trouver tous les en-têtes (contenus dans les balises <h>)

Exemple : supprimer les sections standardisées connues des e-mails

- **Validation des entrées utilisateur**

Exemple : l'e-mail correspond-il au format xxx@xxx.com ?

- **NLP Preprocessing**

Quels motifs représentent les mots individuels dans un texte ?

Exemple : espace + lettres + espace → extraire les lettres

Regex in Python

- Import

```
1 import re
```

- Compile pattern

```
1 p = re.compile('Sherlock Holmes')
```

- Pattern methods

```
1 p.search(text)
```

```
1 p.match(text)|
```

Metacharacters

- . Matches any single character
- ^ Beginning of string
- \$ End of string
- * matches 0 or more characters
- + matches 1 or more characters
- ? Optional character

Metacharacters: Examples

- Helper function:

```
import re
def search_pattern_in_string(pattern, string):
    """
        Returns the string from the first match of
        `pattern` in `string`.
        Returns "No match" if not found.
    """
    search = re.compile(pattern).search(string)
    if not search:
        return "No match"
    else:
        return "Found pattern: " + search.group()
```

Metacharacters: Examples

- **.** Matches any single character
- **^** Beginning of string



```
pattern = "c.t"
string = "I have a cat and a cut."
result = search_pattern_in_string(pattern, string)
print(result) # Output: Found pattern: cat
```



```
pattern = "^Hello"
string1 = "Hello, world!"
string2 = "She said Hello."

result1 = search_pattern_in_string(pattern, string1)
result2 = search_pattern_in_string(pattern, string2)

print(result1) # Output: Found pattern: Hello
print(result2) # Output: No match
```

Metacharacters: Examples

- \$ End of string

```
pattern = "world!$"
string1 = "Hello, world!"
string2 = "world! is a common word."

result1 = search_pattern_in_string(pattern, string1)
result2 = search_pattern_in_string(pattern, string2)

print(result1) # Output: Found pattern: world!
print(result2) # Output: No match
```

Metacharacters: Examples

- * matches 0 or more characters



```
pattern = "ca*t"
string = "I have a ct, a cat, and a caaat."
result = search_pattern_in_string(pattern, string)
print(result) # Output: Found pattern: ct
```

- + matches 1 or more characters



```
pattern = "ca+t"
string = "I have a ct, a cat, and a caaat."
result = search_pattern_in_string(pattern, string)
print(result) # Output: Found pattern: cat
```

Metacharacters: Examples

- ? Optional character

```
pattern = "colou?r"
string = "My favorite color is blue. Their favourite colour is red."
result = search_pattern_in_string(pattern, string)
print(result) # Output: Found pattern: color
```

More Metacharacters

- { m,n} specify number of times character is matched between m and n times
- [] list characters to be matched
- \ escape character
- | or
- () capture group inside parenthesis

Metacharacters: Examples

- { m,n} specify number of times character is matched between m and n times

```
print(search_pattern_in_string("2{1,3}", "221B Baker Street, London"))
print(search_pattern_in_string("2{3,4}", "221B Baker Street, London"))
```

Found pattern: 22

No match

- [] list characters to be matched

```
print(search_pattern_in_string("[ik]", "221B Baker Street, London"))
```

Found pattern: k

Metacharacters: Examples

- \ escape character

```
string = "Is there any other point to which you would wish to draw my attention?"  
# print(search_pattern_in_string "?", string) # would error  
print(search_pattern_in_string("\?", string))
```

Found pattern: ?

- | or

```
print(search_pattern_in_string("z|k", "221B Baker Street, London"))
```

Found pattern: k

Character Classes

- `\s` - matches any whitespace
- `\w` - matches any alpha character.
Equivalent to [A-Za-z]
- `\d` - matches any numeric character.
Equivalent to [0-9]

You may negate these by capitalizing. For example, `\D` matches anything not a digit

Pattern	Meaning	Example Matches
<code>\s</code>	Whitespace	space, tab, newline
<code>\S</code>	Not whitespace	'a', '1', '!'
<code>\w</code>	Word character	'a', 'Z', '5', '_'
<code>\W</code>	Not word character	', '!', '@'
<code>\d</code>	Digit	'0', '1', '9'
<code>\D</code>	Not digit	'a', ' ', '!'

```
▶ import re
```

```
text = "Hello World\tHow\nAre You"
matches = re.findall(r'\s', text)
print(f"Whitespaces: {matches}")
```

```
→ Whitespaces: [' ', '\t', '\n', ' ']
```

```
▶ import re
```

```
text = "A B 123\t!"
matches = re.findall(r'\S', text)
print(f"Non-whitespace: {matches}")
```

```
→ Non-whitespace: ['A', 'B', '1', '2', '3', '!']
```

```
▶ import re
```

```
text = "Hello_World123!@#"
matches = re.findall(r'\w', text)
print(f"Word characters: {matches}")
```

```
→ Word characters: ['H', 'e', 'l', 'l', 'o', '_', 'W', 'o', 'r', 'l', 'd', '1', '2', '3']
```

```
▶ import re
```

```
text = "Phone: 123-456-7890"
matches = re.findall(r'\d', text)
print(f"Digits: {matches}")
```

```
→ Digits: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0']
```

```
▶ import re
```

```
text = "Hello_World123!@#"
matches = re.findall(r'\W', text)
print(f"Non-word characters: {matches}")
```

```
→ Non-word characters: ['!', '@', '#']
```

Character Classes and Groups Example

```
pattern = "(\d+\w*)\s+([A-Z]{1})\w+\s+[A-Z]{1}\w+"
p = re.compile(pattern)
m = p.match("221B Baker Street, London")
print(m.group(0)) # entire address
print(m.group(1)) # first part of address
print(m.group(2)) # second part of address
```

```
221B Baker Street
221B
Baker Street
```

Splitting by Regex

- `re.split()`
- Can be used to split a string on any REGEX match
- Frequently done when you don't know the type of whitespace or to handle dashes and underscores

```
import re

string = "hello; earthlings; how do you; do?"
delimiter = ";" 

split_string = re.split(delimiter, string)

print(split_string)
```

```
['hello', ' earthlings', ' how do you', ' do?']
```

Regarder Devant (Positive Lookahead) : (?=...)

Supprime un caractère (ou un motif) seulement s'il est suivi par un autre motif spécifique.

Syntaxe: X(?=Y)

Signification: Trouve X seulement si X est suivi par Y. Y n'est pas inclus dans le match.

Exemple: Supprimer toutes les virgules qui sont suivies d'un chiffre.



```
import re

texte = "J'ai 1,000,500 dollars, mais seulement 50 cents."
# On veut supprimer les virgules des milliers, mais pas la virgule de la phrase.
pattern = r',(?=,\d)' # Une virgule suivie d'un chiffre

resultat = re.sub(pattern, '', texte)
print(resultat)
# Output: "J'ai 1000500 dollars, mais seulement 50 cents."
```

Regarder Derrière (Positive Lookbehind) : (?<=...)

Supprime un caractère seulement s'il est précédé par un autre motif spécifique.

Syntaxe: (?<=Y)X

Signification: Trouve X seulement si X est précédé par Y. Y n'est pas inclus dans le match.

Exemple: Supprimer l'espace qui suit le sigle "€" (pour coller le prix).



```
import re

texte = "Prix: € 50, TVA: € 10"
# On veut supprimer l'espace après le "€"
pattern = r'(?=<=€) ' # Un espace précédé par '€'

resultat = re.sub(pattern, '', texte)
print(resultat)
# Output: "Prix: €50, TVA: €10"
```

Search and Replace

- `re.sub()` allows for quick and easy search and replace

```
import re

string = "Hey this is a random text from your friend.\n How are " \
         "you doing [name goes here]?\\n Well I hope!"

string = re.sub("\[name goes here\]", "REDACTED", string)

print(string)
```

Hey this is a random text from your friend.
How are you doing REDACTED?
Well I hope!

Find all

- Frequently we will want to find all occurrences of a certain bit of text in a corpus
- `pattern.findall(text)` is very useful for doing this

```
import re

# A lot of text with the numbers '20', '50', and '60' in it
string = "Lorem ipsum dolor sit amet, consectetur adipiscing 20 elit, sed do
pattern = re.compile("\d+")

print(pattern.findall(string))
```

Regex Flags

Flag	Nom Complet	Description
re.I	IGNORECASE	Ignore la casse (majuscules/minuscules)
re.L	LOCALE	Rend \w, \W, \b, \B dépendants de la locale
re.M	MULTILINE	^ et \$ matchent début/fin de chaque ligne
re.S	DOTALL	Le point . match tout, y compris les retours à ligne
re.U	UNICODE	Rend \w, \d, etc. compatibles Unicode (défaut Python 3)
re.X	VERBOSE	Permet des regex plus lisibles avec espaces et commentaires

Greedy vs Non-Greedy

- + and * operators are greedy, they try to match as much as they can.

<div>this is some text</div>

<.+> will not match <div> it will match the entire line.

To make it non greedy (lazy) you can add a ? After the operator

<.+?> will match just <div>

Practical Uses of Greedy

```
▶ import re
```

```
# Extract entire article content between headers
html = "<h1>Title</h1><p>Article content here</p><h2>Subtitle</h2>"
content = re.search(r'<p>.*</p>', html) # Gets the entire paragraph
print(f"Full content: {content.group()}"
```

```
☞ Full content: <p>Article content here</p>
```

```
▶ import re
```

```
path = "/home/user/documents/file.txt"
# Extract entire directory path
directory = re.search(r'^.*/', path)
print(f"Directory: {directory.group()}"
```

```
☞ Directory: /home/user/documents/
```

When to Use NON-GREEDY (Lazy)

```
▶ import re
```

```
html = "<div>Content</div><span>More</span><p>Text</p>"
# Extract individual tags
tags = re.findall(r'<.*?>', html)
print(f"Individual tags: {tags}"
```

```
☞ Individual tags: ['<div>', '</div>', '<span>', '</span>', '<p>', '</p>']
```

```
▶ import re
```

```
data = "name:John,age:30,city:New York,country:USA"
# Extract key-value pairs
# Note: added comma for easier parsing
pairs = re.findall(r'(\w+):(.*?),', data + ',')
print("Key-value pairs:")
for key, value in pairs:
    print(f" {key}: {value}"
```

```
☞ Key-value pairs:
name: John
age: 30
city: New York
country: USA
```

CHAPITRE 1

S'INITIER A NLP: Web Scraping



-  Requests
-  HTTPX
-  BeautifulSoup
-  MechanicalSoup
-  Selenium
-  Playwright
-  Scrapy
-  Crawlee

01 – S'INITIER A NLP

Web Scraping

Introduction

Le web scraping consiste à automatiser l'extraction de données disponibles sur des sites web. Plutôt que de copier manuellement le texte d'une page web, un script de scraping peut récupérer ces données en masse, les organiser, et les préparer pour une utilisation ultérieure.

Pourquoi le Web Scraping est-il important ?

- Collecter de données pour les analyses de marché, études de tendances
- Surveiller de prix pour des plateformes de comparaison
- Agréger de données (par exemple : sites d'offres d'emploi, comparateurs de prix)
- Mettre à jour automatique de bases de données à partir de sources en ligne

Outils couramment utilisés

- **Beautiful Soup**: Un module Python utilisé pour analyser le code HTML et extraire des informations de manière structurée
- **Selenium**: Un outil permettant d'automatiser l'interaction avec le navigateur pour récupérer des données sur des sites dynamiques (par exemple, ceux qui utilisent JavaScript pour générer du contenu)

 BeautifulSoup

Pourquoi utiliser le web scraping en NLP ?

Collecte de grands corpus textuels: des forums, des articles de presse, des blogs, etc.

Extraction de données spécifiques: des critiques de produits, des commentaires sur des réseaux sociaux, des transcriptions de dialogues, etc.

Mise à jour fréquente des données : collecter régulièrement de nouvelles données et ainsi enrichir les modèles de NLP en continu.



01 – S'INITIER A NLP

Web Scraping



Web Scraping avec BeautifulSoup

Beautiful Soup est une bibliothèque Python utilisée pour extraire des données d'un document HTML ou XML. Elle simplifie la navigation et la recherche d'éléments au sein de l'arbre HTTML

Beautiful Soup convertit une page HTML en un arbre de balises, facilitant ainsi l'accès aux différents éléments

Beautiful Soup Permet d'extraire facilement du contenu à partir d'éléments HTML spécifiques, comme les titres, paragraphes, liens, etc., en utilisant des sélecteurs basés sur les balises, les classes CSS ou les IDs

- **Avantages de BeautifulSoup :**

Simple et rapide : idéal pour les sites statiques

Flexibilité : permet d'extraire les informations souhaitées en quelques lignes de code

- **Limites de BeautifulSoup :**

Ne supporte pas le rendu des pages dynamiques. Pour les sites générant du contenu via JavaScript, BeautifulSoup seul ne peut pas charger ni traiter les données

Pour surmonter la limite des sites dynamiques, nous utiliserons Selenium, qui nous permet d'interagir directement avec le navigateur

Web Scraping avec Selenium

Selenium est une bibliothèque Python qui permet d'automatiser les interactions avec un navigateur web. Elle est utilisée pour effectuer du web scraping sur des sites dynamiques, c'est-à-dire des sites générant leur contenu à l'aide de JavaScript

Avec Selenium, on peut simuler les actions d'un utilisateur, comme la navigation, le clic sur des boutons, le défilement de page, etc

Selenium ouvre un véritable navigateur (comme Chrome ou Firefox) dans lequel il exécute le code JavaScript de la page. Cela permet d'accéder au contenu complet, même s'il est généré dynamiquement

- **Avantages de Selenium :**

Adapté aux sites dynamiques : Selenium peut interagir avec les pages JavaScript, contrairement à BeautifulSoup

Automatisation complète : Permet de simuler des actions utilisateur, ce qui est utile pour les sites nécessitant des interactions (connexion, défilement, etc.)

- **Limites de Selenium :**

Plus lent : Comme Selenium utilise un vrai navigateur, il est plus lent que BeautifulSoup

Consommation de ressources : Exécuter un navigateur entraîne une consommation plus élevée de CPU et de mémoire

Étapes du web scraping pour le NLP:

Identification de la source de données:

- Des sites de nouvelles comme CNN ou Le Monde.
- Des forums de discussion comme Reddit.
- Des réseaux sociaux comme Twitter ou Facebook (avec des API spécifiques).
- Des bases de données publiques comme Wikipedia.

Accès à la page web:

via des outils de scraping comme BeautifulSoup pour Python, qui permettent de récupérer le code HTML des pages.

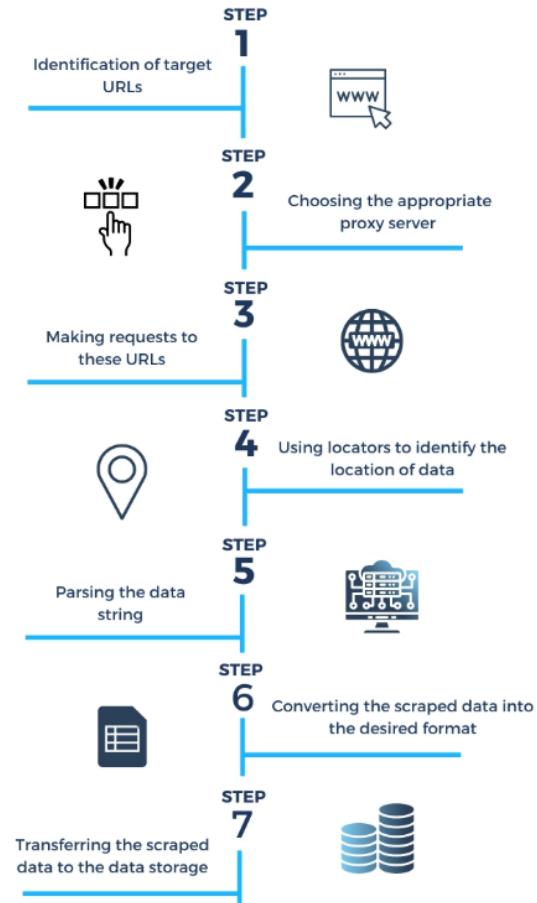
Extraction des données textuelles:

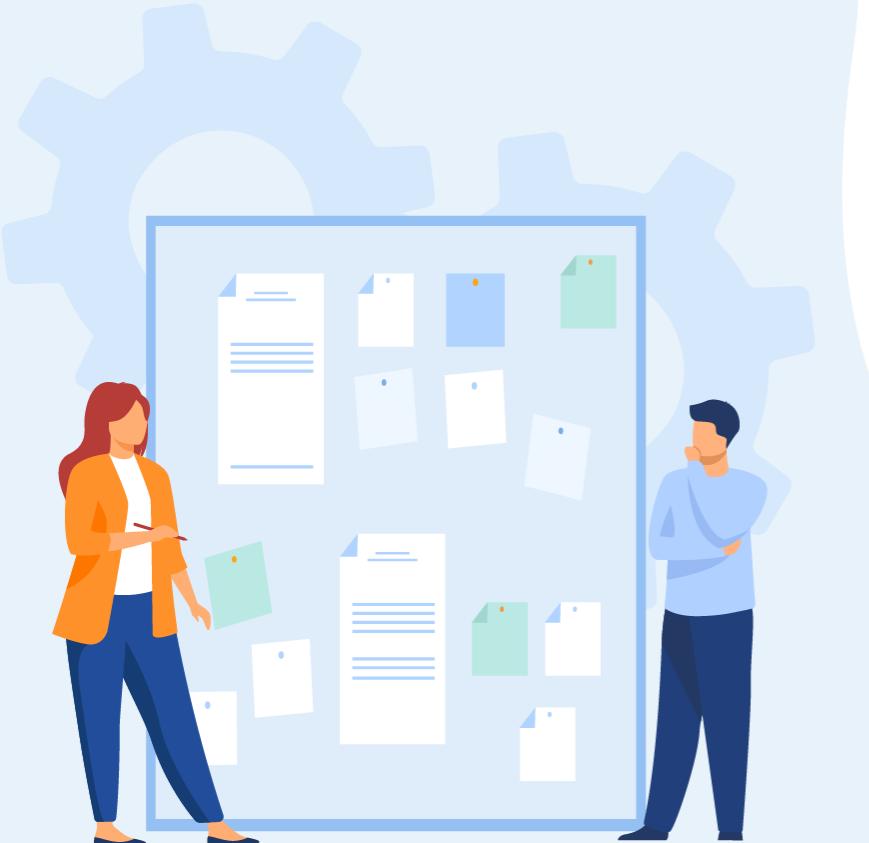
localiser les balises HTML contenant les données souhaitées (<p>, <div>, , etc.).

Nettoyage des données:

Suppression des espaces en trop, Suppression des caractères spéciaux, etc...

The Process of Web scraping





Applications du web scraping en NLP

Analyse de sentiments: Scraper des avis produits ou des tweets pour analyser les sentiments des utilisateurs.

Classification de texte: Scraper des articles de presse pour les classer par catégories (politique, sport, divertissement, etc.).

Résumé automatique: Scraper des articles de blog ou des documents pour générer automatiquement des résumés.

Reconnaissance d'entités nommées (NER): Scraper des pages web pour identifier des entités comme des noms de personnes, lieux, entreprises, etc.

Exemple

Extraction d'un contenu statique avec BeautifulSoup

```
▶ from bs4 import BeautifulSoup
import requests

url = "https://example.com/static"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
titre = soup.find("h1").text
print("Titre de la page:", titre)
```

Extraction d'un contenu dynamique avec Selenium

```
▶ from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.get("https://example.com/dynamic")
titre = driver.find_element(By.TAG_NAME, "h1").text
print("Titre de la page:", titre)
driver.quit()
```

Considérations légales et éthiques:

Le web scraping doit être réalisé dans le respect des lois et des conditions d'utilisation des sites web.

Certaines plateformes ne permettent pas le scraping, ou le restreignent via des fichiers robots.txt ou des API payantes.

Toujours vérifier les termes et conditions d'un site avant d'effectuer du scraping, et respecter les règles en matière de protection des données (comme le RGPD en Europe).



```
import urllib.request
import requests
from bs4 import BeautifulSoup

nb_pages = 1000

def download_file(download_url, filename):
    response = urllib.request.urlopen(download_url)
    file = open(filename + ".pdf", 'wb')
    file.write(response.read())
    file.close()

for n_page in range(1, nb_pages):
    url_i = 'https://core.ac.uk/search.....'
    url = url_i + str(n_page)
    print('HTTP GET: %s', url)
    response = requests.get(url)
    # parse content
    content = BeautifulSoup(response.text, 'lxml')
    # extract URLs referencing PDF documents
    all_urls = content.find_all('a', href=True) #('figure')
    # loop over all URLs
    for url in all_urls:
        try:
            if 'pdf' in url['href']:
                # init PDF url
                pdf_url = ''
```

```
# append base URL if no 'https' available in URL
if 'https' not in url['href']:
    pdf_url = 'https://core.ac.uk/' + url['href']
else:
    pdf_url = url['href']
# make HTTP GET request to fetch PDF bytes
print('HTTP GET: %s', pdf_url)
url_path = pdf_url
pdf_response = requests.get(pdf_url)
# extract PDF file name
filename = urllib.request.unquote(pdf_response.url).....
download_file(url_path, filename)

except:
    pass
```



GitHub



1. Créer un dossier pour le projet

```
mkdir mon-projet  
cd mon-projet
```

2. Initialiser un dépôt Git local

```
git init
```

3. Ajouter tous les fichiers du dossier

```
git add .
```

4. Enregistrer les changements avec un message

```
git commit -m "Premier commit"
```

5. Lier le dossier local au dépôt GitHub

```
git remote add origin https://github.com/votre-nom/mon-projet.git
```

6. Envoyer les fichiers sur GitHub

```
git branch -M main  
git push -u origin main
```

Exercice 2

Vous êtes un trader et, dans votre quotidien, vous devez arrêter les opérations de trading lors de la survenue d'événements économiques critiques (annonces de la FED, taux d'inflation, PIB, etc.). Pour automatiser ce processus, vous devez extraire ces dates à partir du site

<https://www.investing.com/economic-calendar/>

Objectif

Écrire un script **Python** en utilisant **Selenium** pour récupérer les événements économiques critiques d'aujourd'hui

Consignes

- Accéder à la page du calendrier économique
- Extraire les événements classés comme critiques (par exemple, ceux marqués avec une forte importance)
- Extraire la date, l'heure, le titre de l'événement ainsi que les valeurs précédentes, actuelles et prévisionnelles de l'indicateur associé à l'événement
- Retourner ces informations dans un objet Dataframe

Contraintes

- Utiliser **Selenium** pour le scraping en gérant les éventuelles latences du chargement de la page
- Respecter les bonnes pratiques de scraping (headless mode, temps d'attente dynamiques, gestion des erreurs)

Activité 1

Correction

Exercice 2 :

On importe les bibliothèques nécessaires pour le script :

- selenium pour automatiser la navigation web
- pandas pour manipuler les données sous forme de tableau
- time pour gérer les délais d'attente

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import pandas as pd
import time
```

Activité 1

Correction

- On configure les options du navigateur Chrome pour qu'il fonctionne en mode "headless" (sans interface graphique)
- On initialise le pilote (driver) de Selenium pour Chrome en utilisant ChromeDriverManager pour gérer automatiquement le téléchargement du pilote

```
options = webdriver.ChromeOptions()  
options.add_argument("--headless")  
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
```

- On accède à la page web du calendrier économique sur Investing.com
- On attend 5 secondes pour s'assurer que la page est entièrement chargée

```
url = "https://www.investing.com/economic-calendar/"  
driver.get(url)  
  
time.sleep(5)
```

Activité 1

Correction

- On tente de localiser et de cliquer sur le bouton "Accept" pour accepter les cookies
- Si la pop-up n'est pas trouvée, un message est affiché pour indiquer son absence

```
try:  
    accept_button = driver.find_element(By.XPATH, "//button[contains(text(),'Accept')]"")  
    accept_button.click()  
    time.sleep(2)  
except:  
    print("Pas de pop-up de cookies.")
```

- On récupère tous les éléments HTML correspondant aux événements économiques
- Pour chaque événement, on extrait les informations suivantes : heure, devise, nom de l'événement, valeur actuelle, prévision et valeur précédente
- Les données sont stockées dans une liste data

Activité 1

Correction

```
events = driver.find_elements(By.CLASS_NAME, "js-event-item")

data = []
for event in events:
    try:
        time_ = event.find_element(By.CLASS_NAME, "time").text
        currency = event.find_element(By.CLASS_NAME,
"left.flagCur.noWrap").text
        event_name = event.find_element(By.CLASS_NAME, "event").text
        actual = event.find_element(By.CLASS_NAME, "act").text
        forecast = event.find_element(By.CLASS_NAME, "fore").text
        previous = event.find_element(By.CLASS_NAME, "prev").text

        data.append([time_, currency, event_name, actual, forecast, previous])
    except:
        continue
```

- On transforme la liste data en un DataFrame pandas pour une manipulation plus facile
- On sauvegarde les données dans un fichier CSV nommé calendrier_economique.csv
- On affiche un message de confirmation

Activité 1

Correction

```
df = pd.DataFrame(data, columns=["Heure", "Devise", "Événement", "Actuel", "Prévision", "Précédent"])

df.to_csv("calendrier_economique.csv", index=False, encoding="utf-8")

print("Données enregistrées avec succès !")
```

- On ferme le navigateur et termine la session Selenium

```
driver.quit()
```

- On affiche calendrier_economique.csv

```
df = pd.read_csv('calendrier_economique.csv')

print(df)
```

	Heure	Devise	Événement	Actuel	Prévision	Précédent
0	00:30	JPY	Tertiary Industry Activity Index (Jan)	-10.00		7.40
1	04:00	CHF	SECO Economic Forecasts			
2	04:30	HKD	Unemployment Rate (Feb)			3.1%

Étapes de réalisation

<https://unsplash.com/s/photos/laptop>

Préparation de l'environnement :

— Installer les bibliothèques nécessaires :

- ✓ pip install selenium
- ✓ pip install pandas
- ✓ pip install requests
- ✓ pip install webdriver-manager

— Importer les modules webdriver, By, et Service.

Lancer le navigateur automatisé :

— Configurer le navigateur Chrome en mode headless.

— Ouvrir le site choisi à l'aide de driver.get(url).

Scrapper les images :

— Localiser les balises à l'aide de find_elements(By.TAG_NAME, 'img')).

— Extraire les valeurs d'attribut src.

Enregistrer les résultats :

— Créer un dossier images/ s'il n'existe pas.

— Télécharger les images à l'aide de la bibliothèque requests.

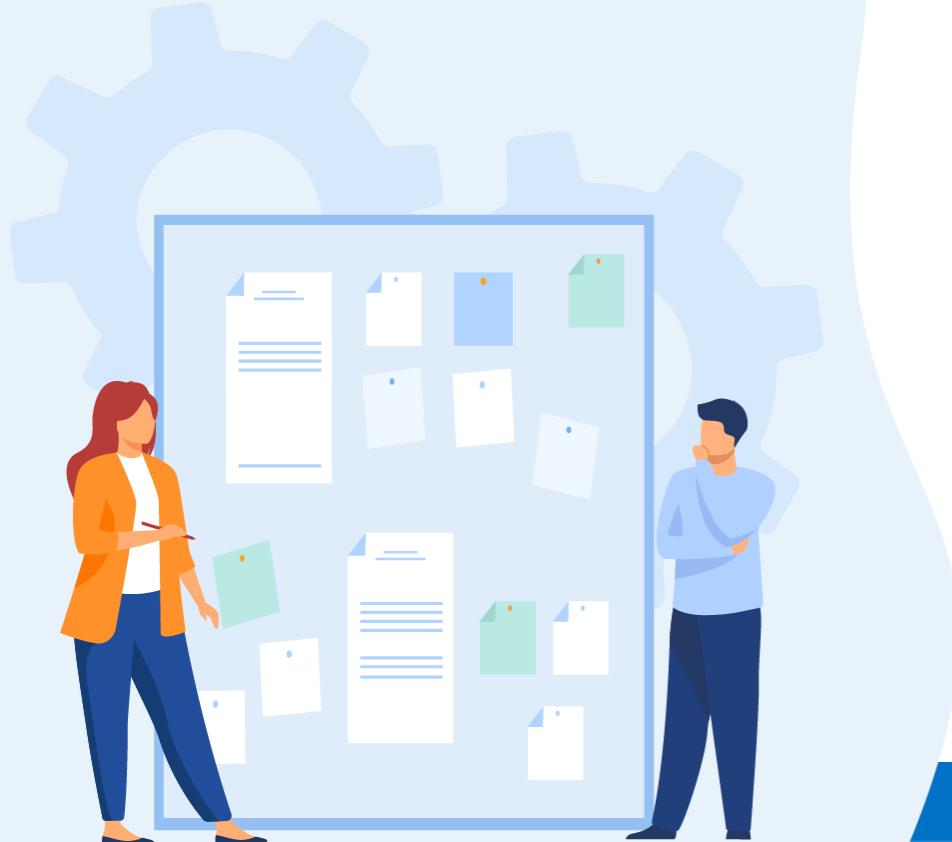
— Sauvegarder les liens dans un fichier CSV.

Exercice : Découverte du Web Scraping avec Selenium en Python

Exercice

Vous allez écrire un script Python utilisant la bibliothèque Selenium pour :

1. Ouvrir un site web de votre choix (par exemple [Unsplash](#) ou [Pixabay](#)).
2. Rechercher une catégorie d'images (ex. : "ordinateur", "chat", "voiture").
3. Extraire les URL des 10 premières images affichées.
4. Télécharger ces images et les enregistrer dans un dossier local nommé images/.
5. Enregistrer les URL des images dans un fichier images.csv.
6. Les stagiaires devront produire :
 - Un script Python fonctionnel nommé scraping_images.py.
 - Un dossier images/ contenant les fichiers téléchargés.
 - Un fichier images.csv listant les liens extraits.



01 – S'INITIER A NLP

Généralités sur NLP

Concepts fondamentaux de NLP

Linguistique vs linguistique computationnelle.

Linguistique: l'étude scientifique du langage humain.

linguistique computationnelle: représenter et modéliser les aspects du langage naturel de manière à pouvoir être traités par des ordinateurs.

Concepts clés de la linguistique:

Morphologie: Étude de la structure des mots et des morphèmes (les plus petites unités de sens ou de fonction).

Exemple : en français, le mot "chanteurs" contient les morphèmes "chant-(racine)", "-eur" (indicateur d'agent) et "-s" (pluriel).

- **Syntaxe:** Étude de la structure des phrases et des règles qui régissent la combinaison des mots dans une langue.

Exemple : en français, l'ordre sujet-verbe-objet (SVO) est courant ("Marie mange une pomme").

- **Sociolinguistique :** Étude des aspects sociaux du langage, y compris les variations selon des facteurs comme la classe sociale, l'âge, le genre, ou la région.

01 – S'INITIER A NLP

Généralités sur NLP

Concepts fondamentaux de NLP

Sémantique: Étude du sens des mots et des phrases.

Comprend la signification littérale des expressions et la manière dont les significations se combinent dans une phrase.

Ambiguïté sémantique: Une phrase peut avoir plusieurs interprétations possibles

Exemple :

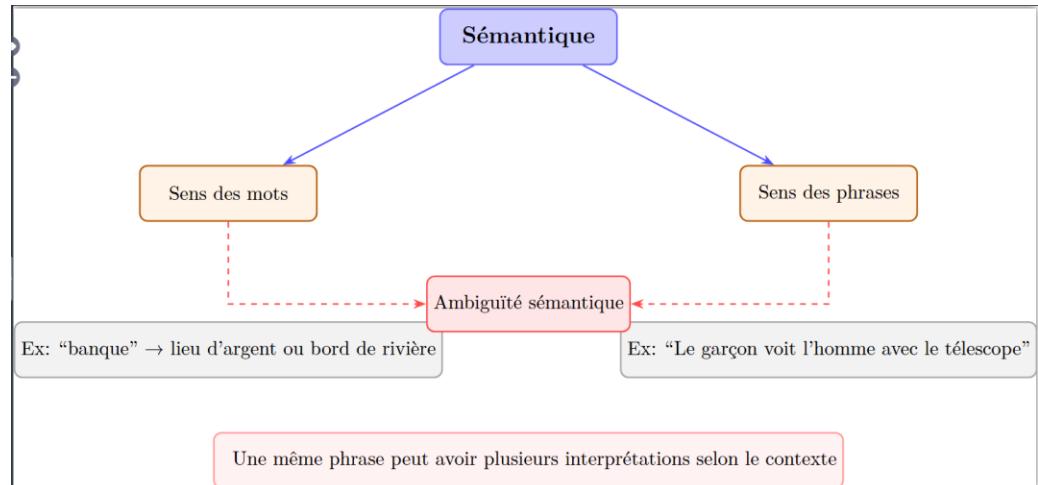
- "Le professeur a dit à l'élève qu'il était intelligent."

- "Ali a vu l'homme avec des lunettes."

- "Je suis un menteur, ne me crois pas."

==> Comment capturer ces concepts clés de la

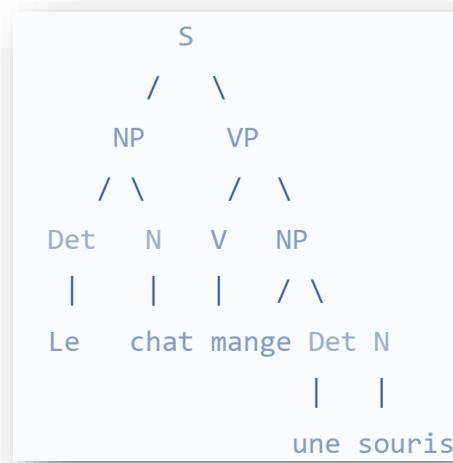
linguistique dans une machine?



L'analyse syntaxique consiste à décomposer une phrase en une structure hiérarchique (appelée arbre syntaxique) selon les règles grammaticales de la langue.

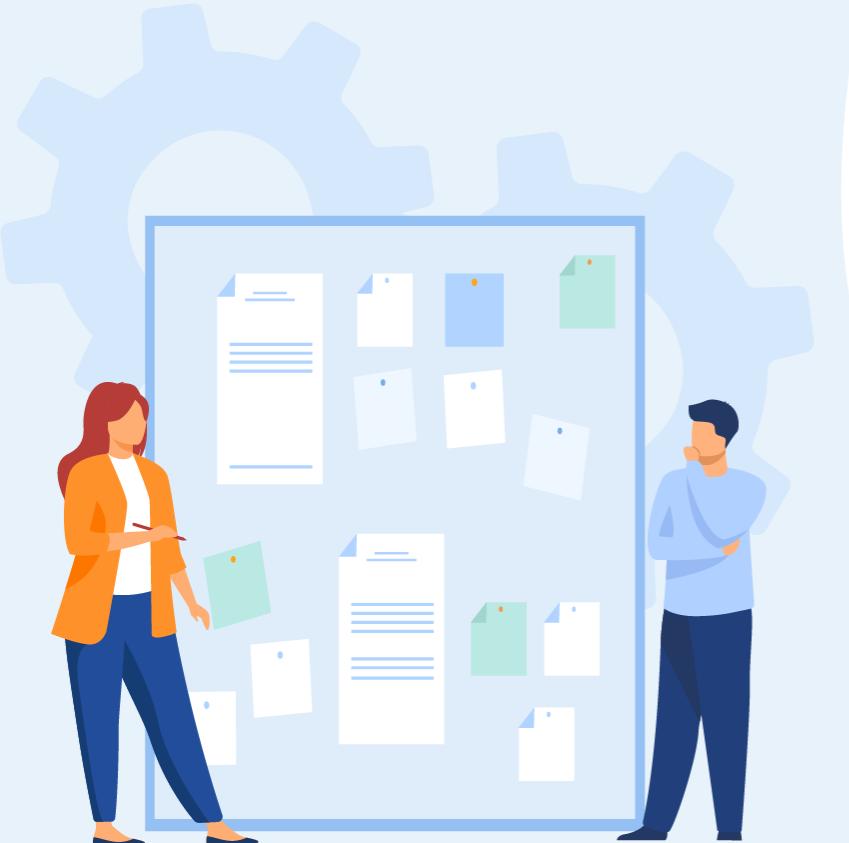
C'est une étape essentielle du NLP, notamment pour la traduction automatique, la compréhension du langage ou l'extraction d'informations.

Arbre syntaxique ==> représenter les relations entre les mots.



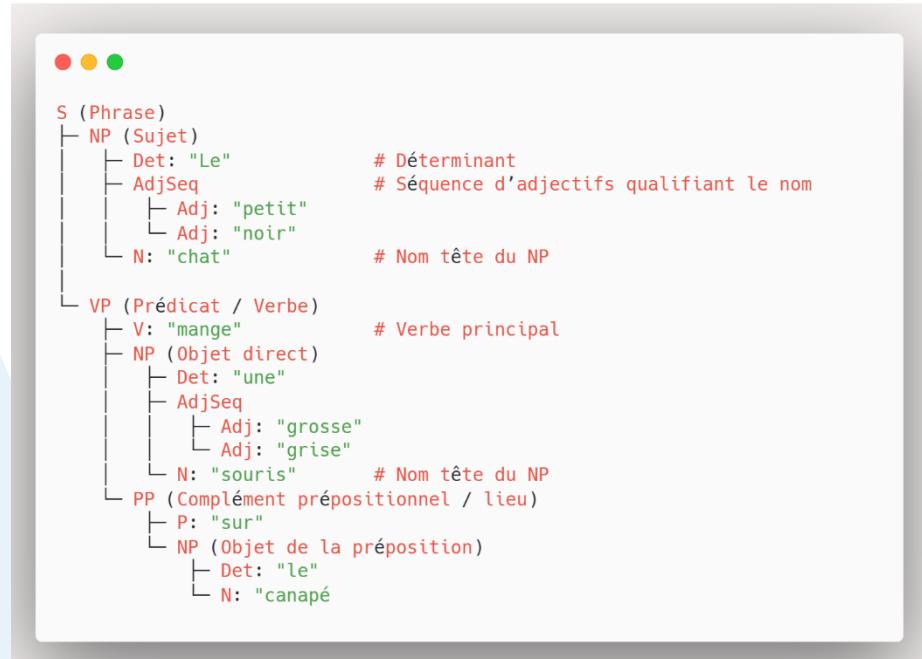
Analyse syntaxique : "Le chat mange une souris."

Symbol	Règle de production (Grammaire contextuelle)	Interprétation linguistique / Exemple
S	$S \rightarrow NP \ VP$	La phrase (S) est composée d'un groupe nominal (NP) suivi d'un groupe verbal (VP).
NP	$NP \rightarrow Det \ N$	Le groupe nominal contient un déterminant (Det) et un nom (N). Exemple : <i>Le chat, une souris.</i>
VP	$VP \rightarrow V \ NP$	Le groupe verbal contient un verbe (V) suivi d'un groupe nominal (NP). Exemple : <i>mange une souris.</i>
Det	$Det \rightarrow 'Le' \mid 'une'$	Les déterminants possibles sont <i>Le</i> (masculin singulier) et <i>une</i> (féminin singulier).
N	$N \rightarrow 'chat' \mid 'souris'$	Les noms définis dans la grammaire sont <i>chat</i> et <i>souris</i> .
V	$V \rightarrow 'mange'$	Le verbe est <i>mange</i> , indiquant l'action effectuée par le sujet.



Exemple: Concepts clés en linguistique computationnelle
Déterminer le grammaire contextuelle et l'arbre syntaxique de la phrase suivante:

S = "Le petit chat noir mange une grosse souris grise sur le canapé"



01 – S'INITIER A NLP

Généralités sur NLP

Concepts fondamentaux de NLP

Pour apprêhender pleinement les capacités et les enjeux de NLP, il est intéressant de se familiariser avec plusieurs concepts fondamentaux. Voici un aperçu des notions clés du NLP

Tokenization

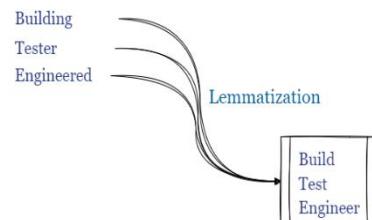
Découpage du texte en unités plus petites: consiste essentiellement à diviser une chaîne de séquence en des unités plus petites. Ces unités sont appelées tokens. Un token peut être un mot, un caractère ou un signe de ponctuation.

Exemple : "J'aime le café." → [" J ", " aime ", " le ", " café ", " . "]

Part-of-Speech Tagging (Etiquetage grammatical)

Attribution de catégories grammaticales à chaque mot : verbe, nom, adjectif, etc

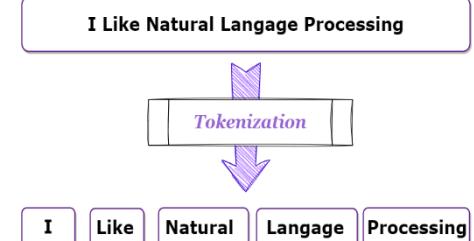
Exemple : " chat " → nom ; " mange " → verbe



Lemmatization (Lemmatisation)

Réduction d'un mot à sa forme canonique

Exemple : " mangera " → " manger "



01 – S'INITIER A NLP

Généralités sur NLP

Concepts fondamentaux de NLP

Stemming (Racine)

Réduction d'un mot à sa racine

Exemple : " jouant " , " joue " , " jouerons " → " jou "

Named Entity Recognition (NER)

Identification d'entités dans un texte : noms propres, lieux, dates et organisations

Exemple : " Barack Obama a visité Paris en 2020. " → Barack Obama (personne), Paris (lieu), 2020 (date)

Sentiment Analysis (Analyse de sentiment)

Détection de l'émotion ou de l'opinion exprimée dans un texte

Exemple : " Ce film est incroyable ! " → Sentiment positif

Parsing (Analyse syntaxique)

Identification de la structure grammaticale d'une phrase.

Exemple : Arbre syntaxique pour " Le chat mange une souris "

01 – S'INITIER A NLP

Généralités sur NLP

Concepts fondamentaux de NLP

Word Embeddings (Représentation vectorielle)

Transformation des mots en vecteurs dans un espace de grande dimension pour capter leur sens et relations

Exemple : Les mots " roi " et " reine " seront proches dans l'espace

Machine Translation (Traduction automatique)

Conversion d'un texte d'une langue à une autre

Exemple : " Bonjour " → " Hello "

Text Generation (Génération de texte)

Production automatique de texte cohérent à partir d'une consigne ou d'un contexte

Exemple : ChatGPT produit des réponses en fonction des questions posées

Speech-to-Text et Text-to-Speech

Speech-to-Text : Transcription d'un discours oral en texte

Text-to-Speech : Conversion d'un texte en voix synthétique

Vecteurs de mots (Word Embeddings): Représentation des mots en tant que vecteurs dans un espace continu, capturant leurs relations sémantiques.

Techniques populaires: Word2Vec, GloVe, et FastText. mots ayant des significations similaires auront des représentations vectorielles proches dans cet espace.

Word2Vec est basée sur deux modèles: CBOW et Skip-gram

CBOW(Continuous Bag of Words): prend en entrée une fenêtre de mots de contexte autour d'un mot et essaie de prédire ce mot.

Exemple :

- ✓ Phrase : "Le chat mange une souris."
- ✓ Contexte : ["Le", "mange", "une", "souris"]
- ✓ Mot cible : "chat"

Skip-gram: fonctionne à l'inverse, il prend un mot cible en entrée et essaie de prédire les mots de son contexte.

Exemple :

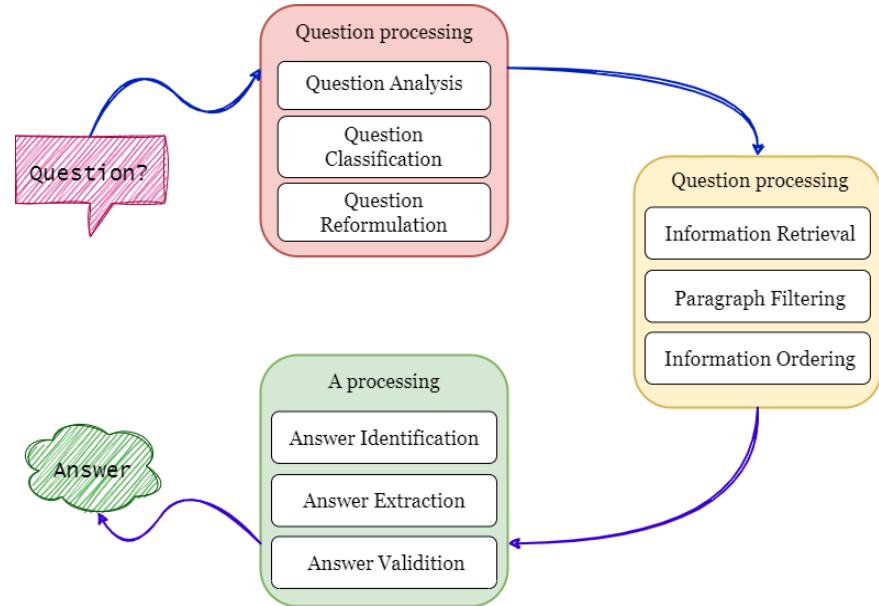
- ✓ Phrase : "Le chat mange une souris."
- ✓ Mot cible : "chat"
- ✓ Contexte : ["Le", "mange", "une", "souris"]



Maîtriser les méthodes quantitatives pour l'analyse textuelle

1. Pratiquer l'analyse syntaxique et sémantique
2. Pratiquer l'apprentissage approfondi

- **Compréhension de la question** : le système analyse la question pour identifier son type, les mots-clés et les entités importantes (ex. personnes, lieux, dates).
- **Recherche de l'information** : il explore une base de données, un corpus ou le web pour repérer les passages les plus pertinents.
- **Extraction et formulation de la réponse** : il extrait la réponse exacte des textes trouvés et la reformule de manière claire et concise pour l'utilisateur.



Le prétraitement est une étape cruciale qui consiste à nettoyer et préparer le texte brut pour qu'il puisse être analysé efficacement par des modèles de Machine Learning ou Deep Learning

Removing Stop-words (Suppression des mots vides)

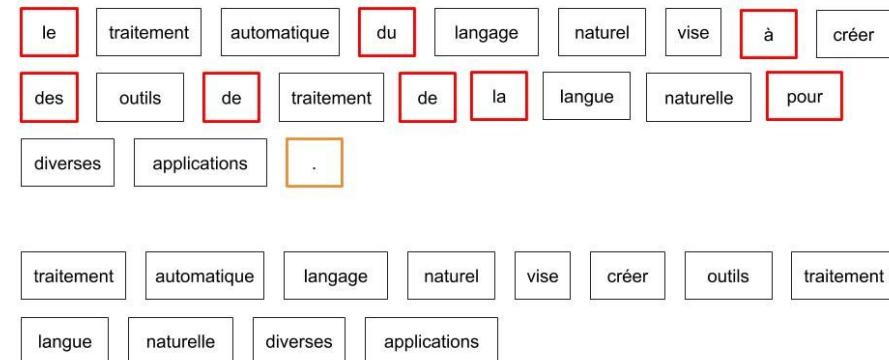
Les mots vides (stop-words) sont des mots très fréquents mais peu informatifs, comme « le », « de », « et », « à » en français.

Les supprimer permet de réduire le bruit dans le texte et de se concentrer sur les mots significatifs.

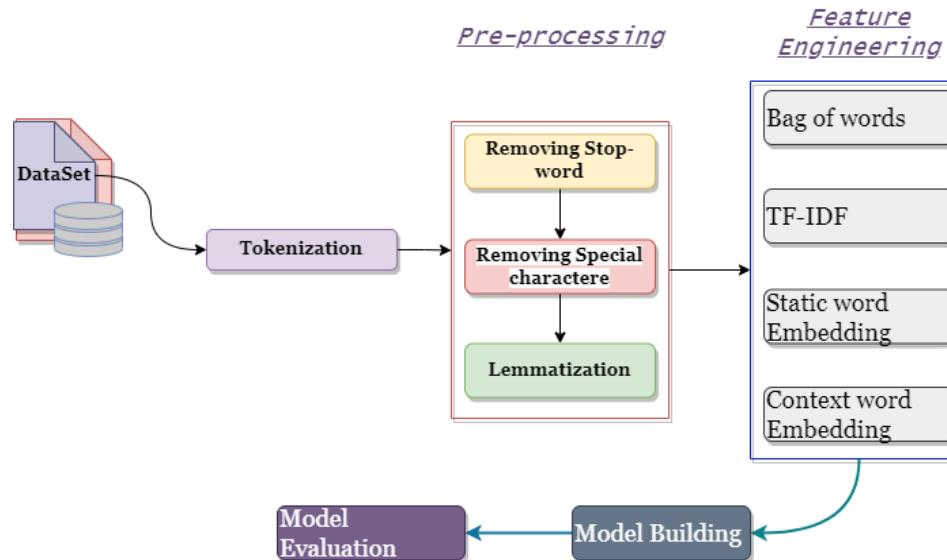
Removing Special Characters (Suppression des caractères spéciaux)

Les caractères spéciaux (punctuation, symboles, emojis, chiffres inutiles) sont souvent supprimés pour normaliser le texte et éviter que le modèle soit perturbé par des éléments non pertinents.

Retrait des stop words



L'analyse de données textuelles consiste à collecter et nettoyer les textes, puis à les transformer en unités exploitables (tokens, lemmes) et en représentations numériques (TF-IDF, embeddings, Transformers). Elle inclut l'identification des entités, l'analyse grammaticale et sémantique, ainsi que l'extraction d'informations pertinentes comme les thèmes ou sentiments. Les résultats sont ensuite utilisés pour la modélisation (classification, extraction, résumé automatique) et peuvent être visualisés ou intégrés dans des applications pour faciliter la prise de décision.



Les mots

- Les mots sont les éléments primitifs de tout texte
- La première tâche d'un programme de NLP consiste souvent à tokeniser un texte, c'est-à-dire à séparer le texte en l'ensemble des éléments qui le constitue
 - ✓ En général, en mot
 - ✓ Mais aussi en groupe de lettres (c.f. Grands modèles de langage)
- La séparation en mots n'est pas si triviale

« He obtained his Ph.D., and then said: I am happy! »

 - Il faut souvent faire plus que séparer avec les espaces et enlever la ponctuation.
 - ✓ Parfois, on veut même grouper des mots ou ponctuations ensemble (ex. : New York, émoticones, hashtags)
- On appelle le vocabulaire l'ensemble des mots d'un texte
- Parfois, on a besoin de séparer les phrases avec un tokenizer spécifique



Tokenisation en Python

```
import spacy

if __name__ == '__main__':
    # python -m spacy download en_core_web_sm
    nlp = spacy.load("en_core_web_sm")
    doc = nlp("He obtained his Ph.D., and then said: I am happy! ")
    print([token.text for token in doc])
    # ['He', 'obtained', 'his', 'Ph.D.', ',', 'and', 'then', 'said', ':', 'I', 'am', 'happy',
    '''']
```

Normalisation en Python

```
import spacy
from nltk.stem import *

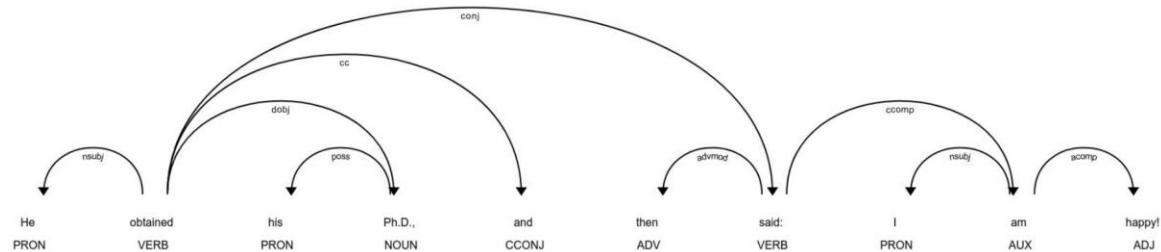
print([token.lemma_ for token in doc])
# ['he', 'obtain', 'his', 'ph.d.', ',', 'and', 'then', 'say', ':', 'I', 'be', 'happy', '!']
stemmer = PorterStemmer()
print([stemmer.stem(token.text) for token in doc])
# ['he', 'obtain', 'hi', 'ph.d.', ',', 'and', 'then', 'said', ':', 'i', 'am', 'happi', '!']
print([token.lemma_ for token in doc if not token.is_stop and not token.is_punct])
# ['obtain', 'ph.d.', 'say', 'happy']
```

Annotations grammaticales - Part-of-Speech & - Dependency parsing

POS = "quelle catégorie est chaque mot", Dependency = "comment les mots se relient entre eux"

```
print([token.pos_ for token in doc]) # PoS simple
# ['PRON', 'VERB', 'PRON', 'NOUN', 'PUNCT', 'CCONJ', 'ADV', 'VERB', 'PUNCT', 'PRON', 'AUX',
'ADJ', 'PUNCT']
print([token.tag_ for token in doc]) # PoS détaillé
# ['PRP', 'VBD', 'PRP$', 'NN', ',', 'CC', 'RB', 'VBD', ':', 'PRP', 'VBP', 'JJ', '.']
print([token.dep_ for token in doc])
# ['nsubj', 'ROOT', 'poss', 'dobj', 'punct', 'cc', 'advmod', 'conj', 'punct', 'nsubj', 'ccomp',
'acomp', 'punct']

# Visualisation Graphique
image = displacy.render(doc, style="dep")
output_path = Path("dependency_plot.svg")
output_path.open("w", encoding="utf-8").write(image)
```



Annotations grammaticales - Named Entity Recognition

La reconnaissance d'entités nommées (Named Entity Recognition = NER) consiste à trouver les noms propres dans un texte et à les classifier (personne, organisation, city, ...)

```
doc = nlp("Elon Musk secretly acquired Twitter stock before buying company, Morgan Stanley hid detail, says shareholder lawsuit ")
```

```
for ent in doc.ents:  
    print(ent.text, ent.label_ )  
  
# Elon Musk PERSON  
# Twitter PRODUCT  
# Morgan Stanley ORG
```

contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok PERSON in posts on



مدن المهن والكفاءات
+٤٣٨٤١٤١ | +٩٦٢٥٧٦٠٨
Cités des Métiers et des Compétences