

# Modèles du NLP

1

## CHAPITRE 3

1. Introduction
2. Les expressions régulières
3. Modèles de concepts
4. Représentations Basées sur des Modèles Statistiques
5. Modèles statistiques de langage
6. Modèles Graphiques : TextRank et Graphes de Similarité)
7. Modèles de langage
  - N-gramme
  - Modèles neuronaux
  - LLMs

## 2. Les modèles du NLP

### Introduction

2

- Après avoir exploré les différentes manières d'aborder les données textuelles, nous pouvons maintenant analyser les **technologies et les types de modèles** associés au TALN.
- Elles sont classées selon la la manière dont le texte est représenté :

Modèles	Expressions régulières	Importance relative TF-IDF	Classification	WordNet	Plongements prédictifs	Statistiques de langage	TextRank
Représentation du texte	Chaine de caractères	Poids chiffré par mot	Tags lexicaux, syntaxiques, sémantiques	Concepts et synsets	Vecteurs de prédiction	Distribution de probabilité	Graphes de similarité

→ Cette liste, non exhaustive, regroupe un échantillon représentatif des modèles actuellement utilisés.

## 2. Les modèles du NLP

# Les expressions régulières (ou regex)

3

- Constituent un mode d'application **déterministe** du NLP.
- Fournissent une méthode **puissante, flexible et efficace** pour le traitement du texte.
- Représentation du texte: chaînes de caractères
- **Fonctionnalités Principales :**
  - Analyser rapidement de grandes quantités de texte pour trouver des motifs de caractères spécifiques.
  - Découper un texte en paragraphes, phrases, mots.
  - Valider le texte pour s'assurer qu'il correspond à un modèle prédéfini (par exemple, une adresse électronique).
  - Extraire, modifier, remplacer ou supprimer des sous-chaînes du texte.
  - Ajouter des chaînes extraites suivant des règles prédéfinies à une collection pour générer un rapport.
- **Limite:** nécessite une grande expertise de la langue.

## 2. Les modèles du NLP

### Les expressions régulières (ou regex)

4

- **Définition d'une expression régulière :**
  - Soit  $\Sigma$  un ensemble de symboles, appelé alphabet.
  - Chaque symbole  $a$  de  $\Sigma$  est une expression régulière. Le symbole  $\epsilon$  est souvent ajouté pour représenter la chaîne vide.
- À partir de deux expressions régulières  $R$  et  $S$ , on peut former de **nouvelles** expressions régulières avec les opérations suivantes :
  - **Concaténation** ( $RS$ ) : cela "colle" deux expressions régulières.
    - Par exemple,  $ab$  concatène  $a$  et  $b$  pour représenter l'ensemble  $\{ab\}$ .
  - **Union** ( $R|S$ ) : cette opération permet de choisir entre deux expressions régulières.
    - Par exemple,  $a|b$  représente l'ensemble  $\{a, b\}$ .
  - **Étoile de Kleene** ( $R^*$ ) : cette opération indique la répétition de l'expression régulière  $R$ , de 0 à  $n$  fois (pour  $n$  quelconque).
    - Par exemple,  $a^*$  représente l'ensemble  $\{\epsilon, a, aa, aaa, aaaa, \dots\}$ .
- Les **parenthèses** peuvent être utilisées pour **regrouper** des termes et spécifier l'ordre des priorités entre les opérations.

## 2. Les modèles du NLP

# Les expressions régulières (ou regex)

5

- **Opérations avancées**
- On peut définir des opérations avancées simplifiant l'écriture (mais n'ajoutant pas à l'expressivité)
  - R? dit que R est présent 0 ou 1 fois ( $= R|\varepsilon$ ) → **Optionnel**
  - R+ dit que R est répété au minimum une fois ( $= RR^*$ )
  - R{n} répète n fois R ( $= RRRRR\dots$ ) → **Répétition exacte**
  - R{n, m} répète R entre n et m fois → **Intervalle de répétition**
  - R{n,} répète R au moins n fois → **Répétition minimale**
  - ^ = début de ligne, \$ = fin de ligne
  - [liste] : union de tous les éléments dans la liste ([abc] = a|b|c)
    - Possibilité de faire des intervalles avec - ([0-9] = 0|1|2|3|4|5|6|7|8|9)
  - [^liste] : union de tous les éléments pas dans la liste ([^abc] = d|e|f|g|...)
  - On peut échapper des caractères spéciaux avec \ (ex.: \|)

## 2. Les modèles du NLP

# Les expressions régulières (ou regex)

6

- **Raccourcis utiles**

- . représente n'importe quel caractère (= a|b|c|d ... pour tout symbole dans  $\Sigma$ )
- \d représente les chiffres (= [0-9])
- \s représente les “espaces” (= [ \t\n\r\f\v])
- \w représente les caractères alphanumériques (= [a-zA-Z0-9\_])
- \b caractère blanc

- **Groupes Nommés dans les Expressions Régulières**

- Un groupe nommé permet de capturer une partie d'une expression régulière en lui attribuant un **nom explicite**.
  - Syntaxe : (?P<nom\_du\_groupe>expression)
  - nom\_du\_groupe : Nom du groupe capturé, utilisé pour accéder facilement à la valeur.
  - expression : Partie de l'expression régulière à capturer.

## 2. Les modèles du NLP

# Les expressions régulières (ou regex)

7

- Avantages des Groupes Nommés
  - **Lisibilité** : chaque groupe capturé a un nom descriptif, ce qui rend le code plus clair.
  - **Facilité d'accès** : accédez aux valeurs capturées en utilisant des noms explicites plutôt que des indices.
  - **Code plus explicite** : idéal pour extraire des données structurées (ex. : adresses email, numéros de téléphone).
- Exemple:

```
import re

# Définition d'une regex avec des groupes nommés
regex = re.compile(r'(?P<username>\w+)@(?P<domain>\w+\.\w+ )')

# Test de la regex
email = "Takwa@gmail.com"
email1="1Takwa@gmail.com"
match = regex.match(email)

# Accès aux groupes par nom et remplacement des valeurs
if match:
    username = match.group("username")
    domain = match.group("domain")

    # Remplacement des groupes par des valeurs spécifiques
    modified_email = email.replace(username, "***").replace(domain, "domaine")
    modified_email1 = email1.replace(username, "***").replace(domain, "domaine")
    print(modified_email) # Résultat : ***@domaine
    print(modified_email1) # Résultat : 1***@domaine
```

## 2. Les modèles du NLP

# Les expressions régulières (ou regex)

8

- Regex en Python

```
import re

if __name__ == '__main__':
    regex = re.compile(r'[a-z]+') # Les mots
    # match commence au début de la string
    print(regex.match("bonjour à toi")) # <re.Match object; span=(0, 7), match='bonjour'>
    print(regex.match("0123456789")) # None
    print(regex.match("0 bonjour")) # None
    # search commence n'importe où
    print(regex.search("0 bonjour")) # <re.Match object; span=(2, 9), match='bonjour'>

    # On peut accéder à des groupes et leur donner des noms
    regex_images = re.compile(r"(?P<filename>\w+)\.(gif|jpeg|jpg|eps|svg|png) ")
    print(regex_images.match("toto.gif").group("filename")) # toto
```

Module pour manipuler des ER

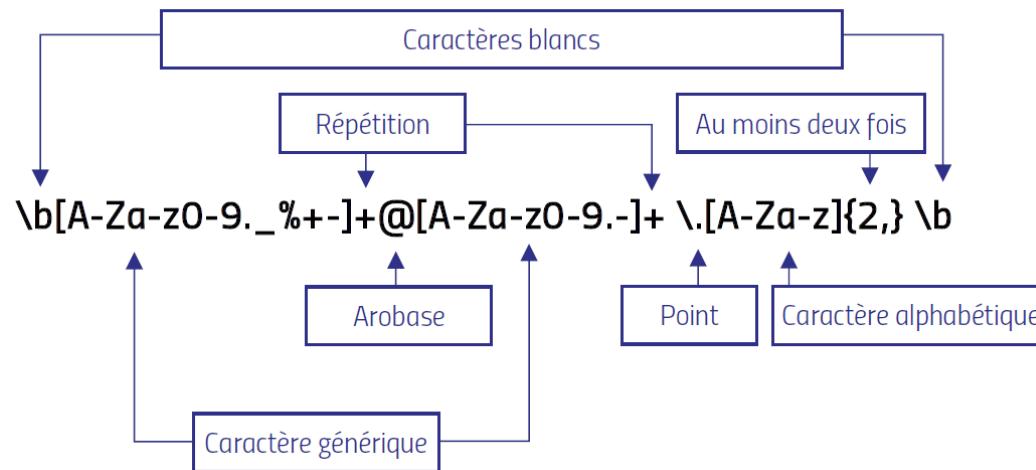
Module pour manipuler des ER

## 2. Les modèles du NLP

### Les expressions régulières-Cas d'utilisation

9

- Les éditeurs de texte utilisent les expressions régulières (ER) pour la recherche et le remplacement.
- La plupart des langages de programmation introduisent des mécanismes pour utiliser les ER.
- Extraction de données : par exemple, extraire les emails et les numéros de téléphones à partir des blogs et des réseaux sociaux.



## 2. Les modèles du NLP

# Les expressions régulières-Cas d'utilisation

10

- À essayer sur <https://regex101.com/>
  - Numéros de téléphones Tunisien : `(\+216)?\d{8}`
  - Adresses mails : `[\w\.-]+@[\\w-]+\.\+[\\w-]\{3\}`
  - Fichiers images : `\w+\.(gif|jpeg|jpg|eps|svg|png)`
  - Extraction d'un message d'erreur : `^\[Error\].*\$`
  - Fichier.txt: `^.*\.txt\$`

## 2. Les modèles du NLP

### Modèles de classification de texte

11

#### 1. Pour l'analyse lexicale: Part-Of-Speech Tagging

- Les modèles de classification appelés « Part-Of-Speech Tagging » (étiquetage des parties du discours) sont conçus pour identifier et classer les mots d'une phrase en fonction de leur catégorie grammaticale.
  - Ces modèles utilisent diverses caractéristiques des mots, telles que :
    - Position dans la phrase
    - Mots voisins (précédents et suivants)
    - Casse des mots (majuscule ou minuscule)
- l'exemple de la phrase suivante:

Une souris a mangé du fromage

Déterminant Nom Verbe Participe Préposition Nom

## 2. Les modèles du NLP

### Modèles de classification de texte

12

- Approche Déterministe vs. Probabiliste
- Il est possible de créer un système déterministe utilisant un référentiel préenregistré pour associer chaque mot à sa catégorie grammaticale. Cependant, établir un tel référentiel est complexe, car de nombreux mots ont des significations variées selon le contexte.
- Exemples :
  - « faible » peut être un nom ou un adjectif.
  - « bien » peut être un nom, un adjectif ou un adverbe.
- Seul un modèle **probabiliste**, capable d'analyser le contexte des mots, peut identifier leur catégorie grammaticale avec **précision**, soulignant ainsi l'importance de l'apprentissage automatique dans le traitement du langage naturel.

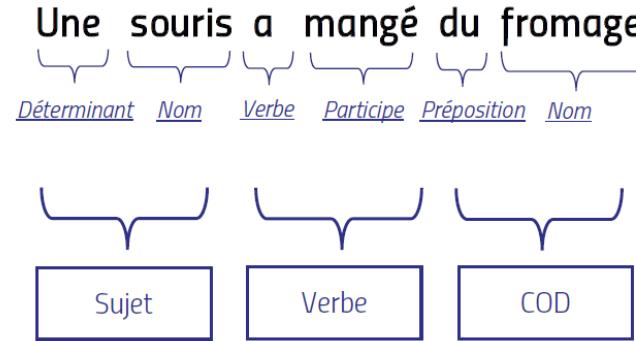
## 2. Les modèles du NLP

### Modèles de classification de texte

13

#### 2. Pour l'analyse syntaxique

- On peut utiliser des modèles de **parsing**, qui permettent, à partir du POS Tagging des mots d'une phrase, de **regrouper les mots en groupes** et de leur donner une **fonction**.
- L'exemple précédent devient alors :



#### 3. Pour l'analyse sémantique: Reconnaissance d'Entités Nommées

- L'analyse sémantique est essentielle pour **comprendre le sens du texte**. L'une des méthodes clés dans ce domaine est la Reconnaissance d'Entités Nommées (Named Entity Recognition, NER), qui **identifie** et **classe les mots** en fonction de **catégories sémantiques** importantes.

→ La NER consiste à déetecter des entités significatives dans un texte et à les classer dans des catégories prédéfinies.

## 2. Les modèles du NLP

# Modèles de classification de texte

14

- Catégories courantes :
  - Entreprises, Personnes, Organisations, Dates, Pays, Villes, ...
- La NER **repose sur l'analyse syntaxique des groupes de mots**, obtenue grâce à des étapes préliminaires.
  - Cela permet d'extraire des caractéristiques essentielles qui aident à la classification.
  - Caractéristiques Utilisées :
    - Position des mots dans la phrase
    - Structure grammaticale
    - Relations entre les mots
- Approches de Détection:
  - Référentiels Préenregistrés : on peut utiliser des référentiels qui associent des mots à leurs catégories, facilitant ainsi leur identification.
  - Modèles Avancés de NER :
    - Capables de détecter des **mots inconnus** au référentiel.
    - Classifient des mots **pouvant appartenir à plusieurs catégories** selon le contexte (ex. : « Paris » peut désigner une ville ou une entreprise).

## 2. Les modèles du NLP

### Modèles de classification de texte

15

- Importance de la NER
  - Compréhension Contextuelle : elle améliore la compréhension sémantique en tenant compte du contexte d'utilisation des mots, permettant une analyse plus précise des données textuelles.
  - Applications Pratiques : utilisée dans divers domaines tels que:
    - l'analyse des sentiments
    - la recherche d'informations
    - la gestion de contenu,...

#### 4. Pour l'analyse de sentiment

- L'analyse de sentiment est une tâche essentielle en TALN, visant à évaluer les émotions exprimées dans un texte.
- Elle permet de classer les propos en différentes catégories sentimentales.
- Les **modèles de classification binaire (SVM / RN)** sont utilisés pour catégoriser les sentiments exprimés dans un texte.
  - Ils associent chaque mot à un coefficient de positivité ou de négativité en fonction du sentiment ciblé.

## 2. Les modèles du NLP

# Modèles de classification de texte

16

- Types de Sentiments Classifiés :

- **Positif** : exprime un sentiment favorable ou agréable.
- **Négatif** : exprime une critique ou un mécontentement.
- **Neutre** : pas de connotation émotionnelle forte.

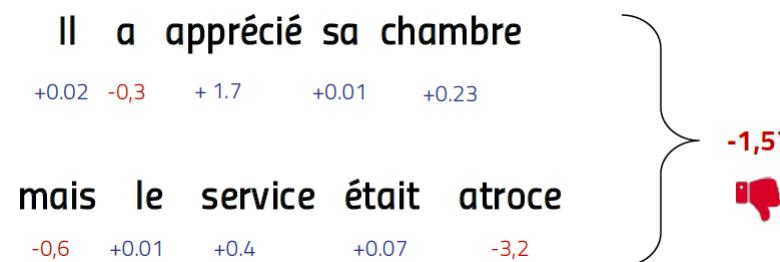
- Processus d'Apprentissage

1. Association de Coefficients :

1. Pendant la phase d'apprentissage, chaque mot du corpus est associé à un coefficient représentant son impact sur le sentiment (positif ou négatif).
2. Ces coefficients sont calculés à partir d'exemples préalablement classés, permettant au modèle d'apprendre les nuances du langage.

2. Prédiction du Sentiment Global :

1. Lorsqu'un nouveau texte est analysé, le modèle prédit le sentiment global en sommant les coefficients de tous les mots présents dans le texte pour déterminer si le sentiment exprimé est positif ou négatif.



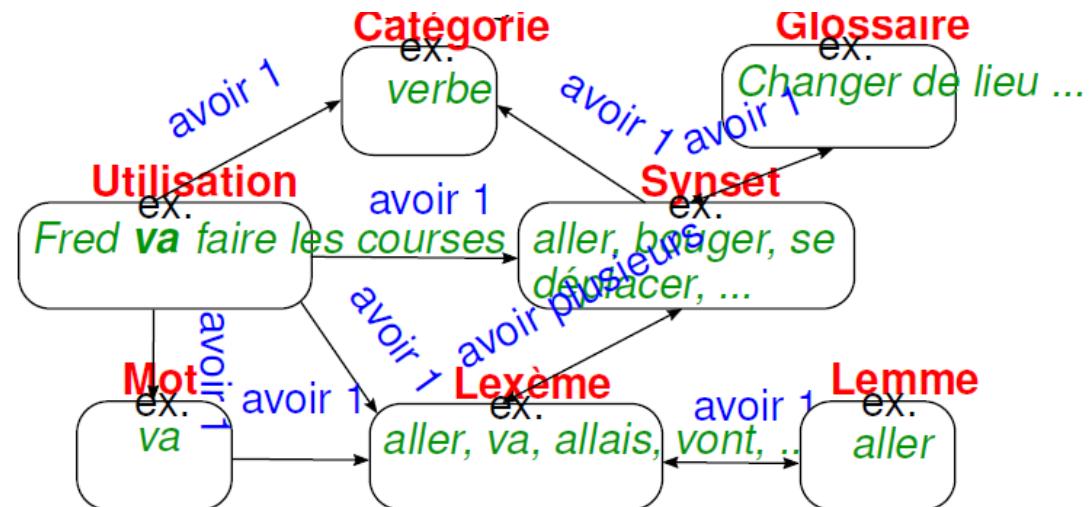
## 2. Les modèles du NLP

### Modèles de concepts

17

#### Bases de données lexicales

- Les linguistes ont répertorié les attributs sémantiques des mots dans des bases de données lexicales pour une meilleure compréhension du langage.
- Un modèle de concept utilise ces bases pour identifier de manière précise les concepts associés à un mot.
- Elles sont essentielles en TALN pour structurer et représenter les informations linguistiques de manière organisée.
- Une BD lexicale regroupe des mots avec leurs significations, leurs relations et leurs usages dans un contexte linguistique



-Exemple des informations et leurs relations dans une base lexicale

## 2. Les modèles du NLP

### Modèles de concepts

18

#### Bases de données lexicales-Relations sémantiques

- **Synonymie** : avoir des sens similaires dans un contexte donné
  - **Antonymie** : avoir des sens opposés dans un contexte donné
  - Les relations taxonomiques (de classification)
    - **Hyponymie** : être plus spécifique qu'un autre sens. Il entraîne une relation **IS-A**.
      - ✖ Ex. voiture IS-A véhicule
    - **Hyperonymie** : être plus générique qu'un autre sens.
    - **Méronymie** : être une partie d'une chose.
      - ✖ Ex. roue est un **méronymie** de voiture ; voiture est le **holonyme** de roue
  - ...
- Permet une meilleure compréhension du texte en enrichissant les modèles avec des connaissances sémantiques.
- Utilisé pour des tâches telles que l'analyse sémantique et la désambiguïsation des mots.

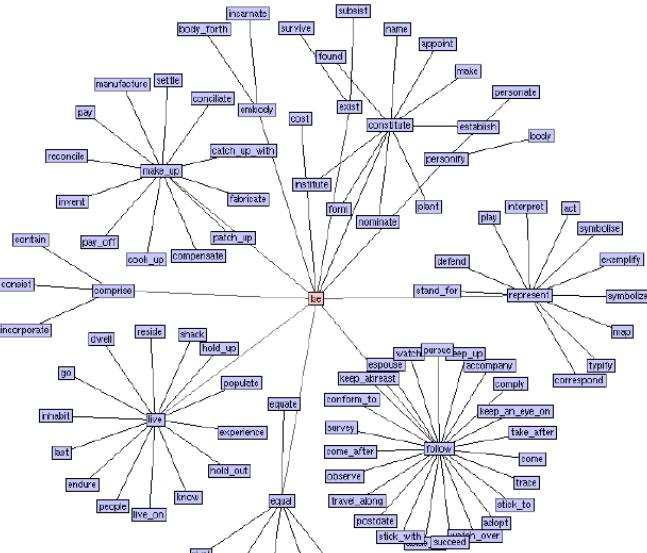
## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

19

# **Organisation et Structure**

- WordNet organise les mots en quatre catégories principales : **noms, verbes, adjectifs et adverbes.**
  - Chaque mot est associé à un ou plusieurs synsets (ensembles de synonymes) en fonction de ses différents sens.
  - Un **synset** correspond à une **signification spécifique d'un mot** et regroupe des synonymes partageant ce même sens.
  - Il offre une approche pour approfondir la compréhension du langage.
  - Par exemple, le synset 05659525 inclut les termes "reason", "understanding" et "intellect", illustrant un concept lié à la capacité de pensée rationnelle.



## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

20

- Ici, le mot « **souris** » est associé à **4 concepts** différents, chacun étant listé avec son « synset » et sa définition :



[Souris, pointeur] : Dispositif de pointage pour ordinateur



[Souris, mus musculus] : Rongeur de petite taille au museau pointu, aux oreilles rondes, au pelage gris-brun et une queue relativement longue et mince.



[Souris, muscle] : Morceau de viande constitué par le tibia de la patte arrière de l'agneau, en bas de cuisse.



[Souris, jeune fille] : Terme familier pour désigner une jeune fille.

## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

21

#### Représentation Sémantique

- Chaque synset est accompagné d'un glossaire (ou gloss) décrivant la signification du terme.
  - Exemple : Pour 05659525, le glossaire est "(the capacity for rational thought or inference or discrimination)".

#### Classification Lexicographique

- Les synsets sont catégorisés par des supersenses (catégories sémantiques générales pour situer le mot dans un cadre conceptuel plus large).
  - Exemple : Le synset 05659525 est étiqueté comme "noun.cognition , indiquant son appartenance au domaine cognitif (pensée, raisonnement, ...)

#### Réseau de Relations Sémantiques

- WordNet relie les sens des mots à travers différentes relations sémantiques comme les synonymies, antonymies, hyperonymies (catégories générales) et hyponymies (sous-catégories).
  - Ces relations permettent de naviguer entre les concepts pour mieux comprendre le langage de manière plus approfondie.

## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

22

→ En représentant le texte sous forme de concepts et de synsets, WordNet permet de réaliser des **analyses sémantiques avancées**, facilitant la désambiguïsation lexicale, la classification de textes, et le traitement contextuel.

- **Catégories lexicographiques\* des noms dans WordNet (supersense)**

Catégorie	Exemple	Catégorie	Exemple	Catégorie	Exemple
ACT	service	GROUP	place	PLANT	tree
ANIMAL	dog	LOCATION	area	POSSESSION	price
ARTIFACT	car	MOTIVE	reason	PROCESS	process
ATTRIBUTE	quality	NATURAL EVENT	experience	QUANTITY	amount
BODY	hair	NATURAL OBJECT	flower	RELATION	portion
COGNITION	way	OTHER	stuff	SHAPE	square
COMMUNICATION	review	PERSON	people	STATE	pain
FEELING	discomfort	PHENOMENON	result	SUBSTANCE	oil
FOOD	food			TIME	day

\*Catégories sémantiques générales

## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

23

- Relations sémantiques (Noms)
- Quelques relations des noms:

Relation	Définition	Exemple
Hypernym	d'un concept spécifique vers un autre générique	breakfast <sup>1</sup> → meal <sup>1</sup>
Hyponym	d'un concept générique vers un autre spécifique	meal <sup>1</sup> → lunch <sup>1</sup>
Instance Hypernym	d'une instance vers son concept	Austen <sup>1</sup> → author <sup>1</sup>
Instance Hyponym	d'un concept vers son instance	composer <sup>1</sup> → Bach <sup>1</sup>
Part Meronym	d'un concept entier vers une partie	table <sup>2</sup> → leg <sup>3</sup>
Part Holonym	d'une partie vers un entier	course <sup>7</sup> → meal <sup>1</sup>
Antonym	d'un concept vers son opposition sémantique	leader <sup>1</sup> ↔ follower <sup>1</sup>
Derivation	d'un mot vers un autre ayant la même racine	destruction <sup>1</sup> ↔ destroy <sup>1</sup>

## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

24

- Relations sémantiques (**Verbes**)
- Quelques relations des verbes:

Relation	Définition	Exemple
Hypernym	d'un évènement spécifique vers un autre générique	fly <sup>9</sup> → travel <sup>5</sup>
Troponym	d'un évènement générique vers un autre spécifique	walk <sup>1</sup> → stroll <sup>1</sup>
Entails	d'un évènement vers un autre qui l'implique	snore <sup>1</sup> → sleep <sup>1</sup>
Antonym	d'un évènement vers son opposition sémantique	increase <sup>1</sup> ↔ decrease <sup>1</sup>

## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

25

- **Rélation avec la génération du texte / autocomplétion**

- WordNet aide à choisir le bon **sens** d'un mot (synonymes, antonymes, hyperonymes, etc.).
- Enrichissement du texte en utilisant des **synonymes**.
- Propositions de **variantes** pour éviter les répétitions.
- **Suggestions adaptées** au contexte sémantique.
- Prédictions du mot suivant en tenant compte des différents sens du mot.
- Choix du sens correct d'un mot en fonction du contexte, grâce aux relations dans WordNet.
  - ✖ ex. : « souris » peut être un animal ou un dispositif.
- WordNet renforce les modèles de langue (ex. GPT, BERT) en fournissant une **connaissance sémantique explicite**.
  - ✖ Meilleure **compréhension du contexte** et génération de texte plus riche et naturel.

## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

26

- Quelques APIs
  - **NLTK** (Python) : <https://www.nltk.org/howto/wordnet.html>  
→ Très utilisée pour l'analyse linguistique, permet d'accéder directement aux synsets, définitions et relations sémantiques.
  - **JWI** (Java WordNet Interface) : <http://projects.csail.mit.edu/jwi/>  
→ Interface Java officielle pour manipuler la base WordNet localement.
  - **Wordnet** (Ruby) : <https://github.com/doches/wordnet>  
→ Une version Ruby utile pour les développeurs web.
  - **OpenNlp** (C#) : <https://github.com/AlexPoint/OpenNlp>  
→ Bibliothèque pour .NET intégrant des fonctionnalités NLP, dont WordNet.
- Autres langues
  - **Global WordNet Association** :  
<http://globalwordnet.org/resources/wordnets-in-the-world/>  
→ Répertorie les WordNets disponibles dans de nombreuses langues (français, espagnol, arabe, chinois, etc.).
  - **Open Multilingual WordNet (OMW)** : <http://compling.hss.ntu.edu.sg/omw/>  
→ Plateforme reliant plusieurs WordNets multilingues pour comparer et traduire les concepts entre langues.

## 2. Les modèles du NLP

### Modèles de concepts-Le modèle WordNet

27

#### Python

```
!pip install nltk
import nltk
nltk.download('wordnet')

from nltk.corpus import wordnet as wn

# 1. Rechercher les synsets pour un mot
mot = "souris"
synsets = wn.synsets(mot, lang='fra')

print(f"Synsets pour '{mot}' :")
for synset in synsets:
    print(f"  {synset.name()} : {synset.definition()}")
print('-----')
# 2. Obtenir les hyperonymes (catégories générales)
premier_synset = synsets[0]
hyperonymes = premier_synset.hypernyms()

print("\nHyperonymes :")
for hyperonyme in hyperonymes:
    print(f"  {hyperonyme.name()} : {hyperonyme.definition()}")
```

```
Synsets pour 'souris' :
mouse.v.02 : manipulate the mouse of a computer
mouse.n.01 : any of numerous small rodents typically resembling dim
house_mouse.n.01 : brownish-grey Old World mouse now a common house
mouse.n.04 : a hand-operated electronic device that controls the co
shiner.n.01 : a swollen bruise caused by a blow to the eye
-----
Hyperonymes :
manipulate.v.02 : hold something in one's hands and move it
```

## 2. Les modèles du NLP

### Modèles de concepts

28

- Autres ressources : Autres BD lexicales:

- FrameNet

- Basée sur la théorie “cadre sémantique” (frame semantic)
    - Un cadre peut être un événement, une relation ou une entité avec ces participants
      - Ex. Le concept “Cuisinier” implique une personne qui cuisine, la nourriture, un récipient et une source de chaleur
    - Chaque cadre est activé par un ensemble des **unités lexicales**.
      - Ex.: blanchir, bouillir, griller, dorer, mijoter, cuire

- VerbNet

- Une base lexicale pour verbes
    - Elle inclue 30 rôles thématiques principaux
    - Les verbes sont organisés en classes

## 2. Les modèles du NLP

### Modèles de concepts

29

- Les bases de données lexicales sont **déterministes** et **rigides**, ce qui les rend peu adaptées à capturer les **variations** et **évolutions dynamiques** du langage.
- Leur construction nécessite un travail manuel intensif, et malgré cela, elles restent souvent incomplètes en raison de la richesse et de la complexité du langage naturel.
- Pour dépasser ces limitations, les modèles **statistiques** et modèles neuronaux ont été proposés.
- → Ces modèles peuvent apprendre automatiquement des structures linguistiques en utilisant de grandes quantités de données textuelles.
- Cette évolution conduit naturellement vers des approches probabilistes telles que:
  - les modèles n-grammes,
  - les modèles neuronaux:
    - ex: Word Embeddings et TextRank: qui permettent une représentation plus flexible et contextuelle du langage.

## 2. Les modèles du NLP

### Modèles statistiques de langage

30

- **Objectif:** un des buts des modèles statistiques de langage est de **construire un modèle qui peut estimer la distribution du langage naturel** de manière aussi précise que possible.
- **Avantages:**
  - fournir un moyen simple de traiter le langage naturel et qui peut s'adapter à des textes très différents.
  - Utilisent l'apprentissage non supervisé, éliminant ainsi **le besoin de fournir des réponses au modèle ou de faire un étiquetage manuel de corpus textuels.**
  - → Le modèle apprend grâce aux associations statistiques entre les mots.
- **Représentation du texte: distribution de probabilité**

## 2. Les modèles du NLP

# Modèles statistiques de langage

31

### Modèles de Thèmes Probabilistes

- **Objectif:** découvrir les thèmes dominants dans des textes donnés
  - Ils sont une sous-catégorie des modèles statistiques de la langue.
- **Représentation des Thèmes :**
  - Les thèmes sont représentés par des **distributions de probabilités**.
  - Ces distributions modélisent la fréquence d'apparition de chaque mot dans les textes associés aux thèmes.
- **Avantages:**
  - Les thèmes englobent non seulement les mots qui les décrivent précisément, mais aussi les mots liés.
  - Un mot peut appartenir à plusieurs thèmes, ce qui est utile pour les mots ayant plusieurs sens.

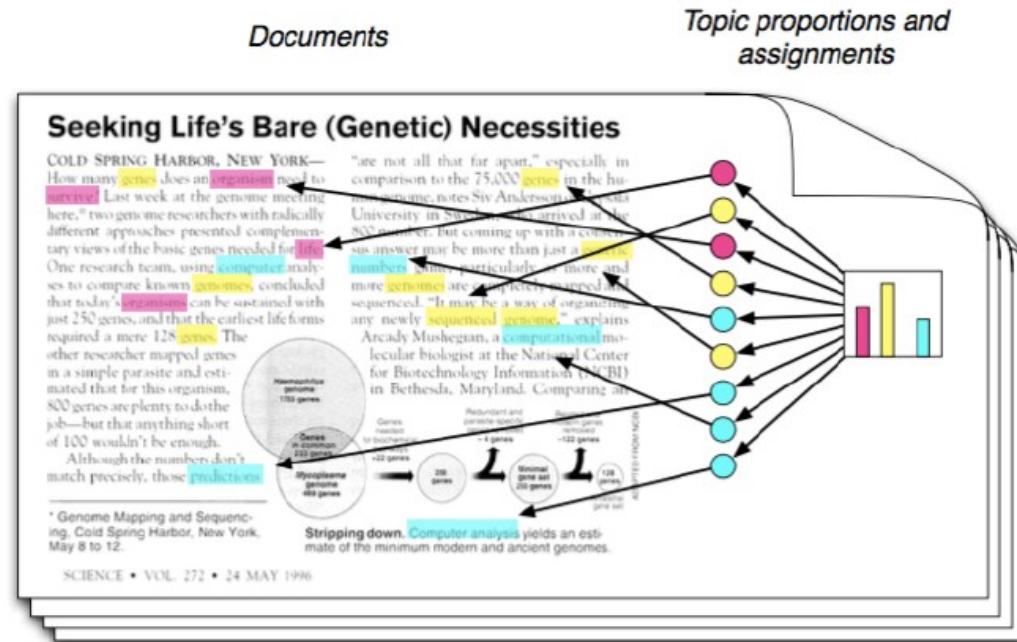
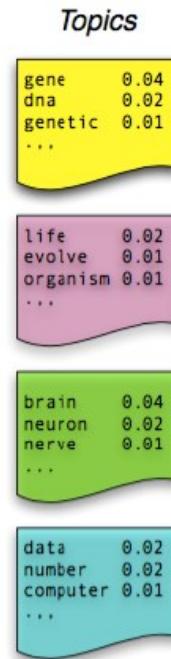
## 2. Les modèles du NLP

### Modèles statistiques de langage

(32)

## Fonctionnement d'un modèle de thème probabiliste

*Source:*  
David Blei, Commun. ACM  
(2012) 77-84



## Fonctionnement d'un modèle de thèmes probabiliste

## 2. Les modèles du NLP

### Modèles statistiques de langage

33

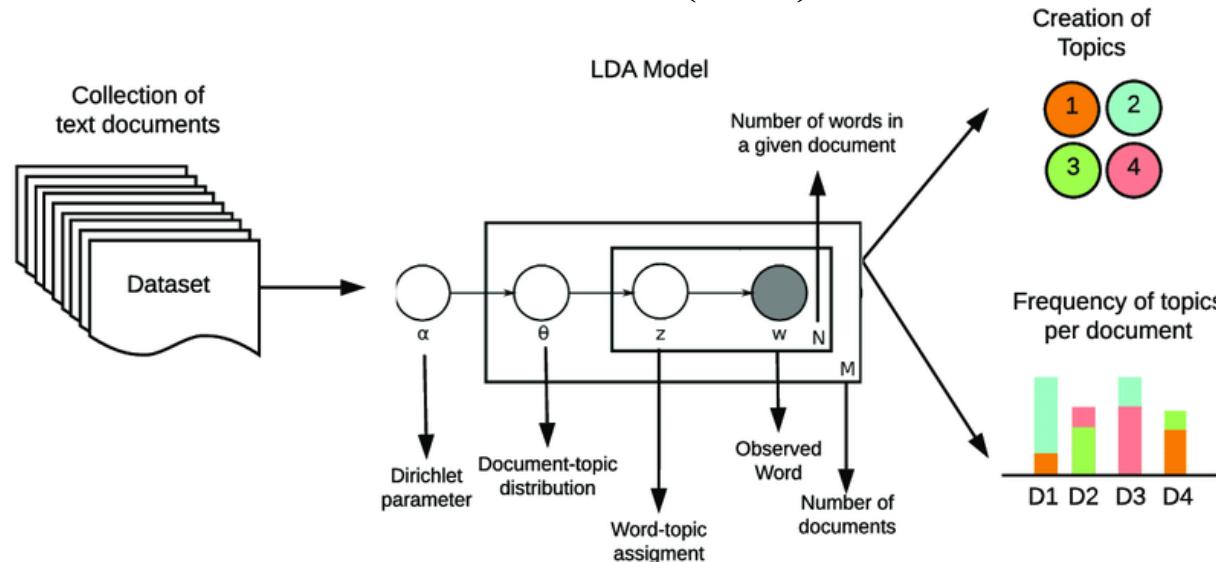
- **Exemple des modèles:**
  - **Mixture Model** : il suppose que les données proviennent de plusieurs distributions différentes (**différents types de langage** sont mélangés).
  - **Modèle de Langue de Dirichlet (LDA)** : il est utilisé pour modéliser des distributions multinomiales, c'est-à-dire des distributions où **plusieurs catégories ou thèmes sont présents dans un document**. Il est utilisé pour extraire des thèmes latents dans un ensemble de textes.
  - **Modèle de Langue de Maximum d'Entropie (MaxEnt)** : Ce modèle est basé sur le principe du **maximum d'entropie**, qui permet de choisir la distribution de probabilité la moins **influencée** en l'absence d'information supplémentaire.
    - ▣ Il est souvent utilisé lorsqu'on veut éviter d'introduire des hypothèses non fondées.

## 2. Les modèles du NLP

# Modèles statistiques de langage

34

### Exemple d'application: Latent Dirichlet Allocation (LDA)



**Documents :**

Une [collection de documents](#), comme des articles de presse, des blogs, ou des revues scientifiques.

**Mots :**

Chaque document est composé de mots. Ces mots forment le [vocabulaire](#) de votre collection.

**Objectif du modèle :**

Découvrir les thèmes qui traversent ces documents, même si ces thèmes ne sont pas explicitement étiquetés.

## 2. Les modèles du NLP

# Modèles statistiques de langage

35

### Étapes et Composants du Modèle LDA

**1. Collection de documents** : Chaque document est vu comme une combinaison de plusieurs sujets.

**2. Paramètre Dirichlet ( $\alpha$ )** : Contrôle le nombre de sujets qu'un document peut aborder.

- Une faible valeur de  $\alpha$  favorise moins de sujets par document, tandis qu'une valeur élevée en permet davantage.

### Algorithme :

**1. Initialisation** : Chaque document est une combinaison de plusieurs thèmes. Au départ, chaque mot est attribué aléatoirement à un sujet (catégories vides à  $t=0$ ).

**2. Itérations (Réattribution de thèmes)** : Pour chaque mot de chaque document :

- Réattribuer le mot à un sujet selon la probabilité qu'il appartienne à chaque sujet (par la méthode Gibbs sampling), en tenant compte de la distribution des sujets dans le document et dans l'ensemble des documents.

**3. Optimisation** : Répéter la réattribution jusqu'à convergence, c'est-à-dire lorsque les sujets attribués stabilisent la structure sous-jacente des thèmes.

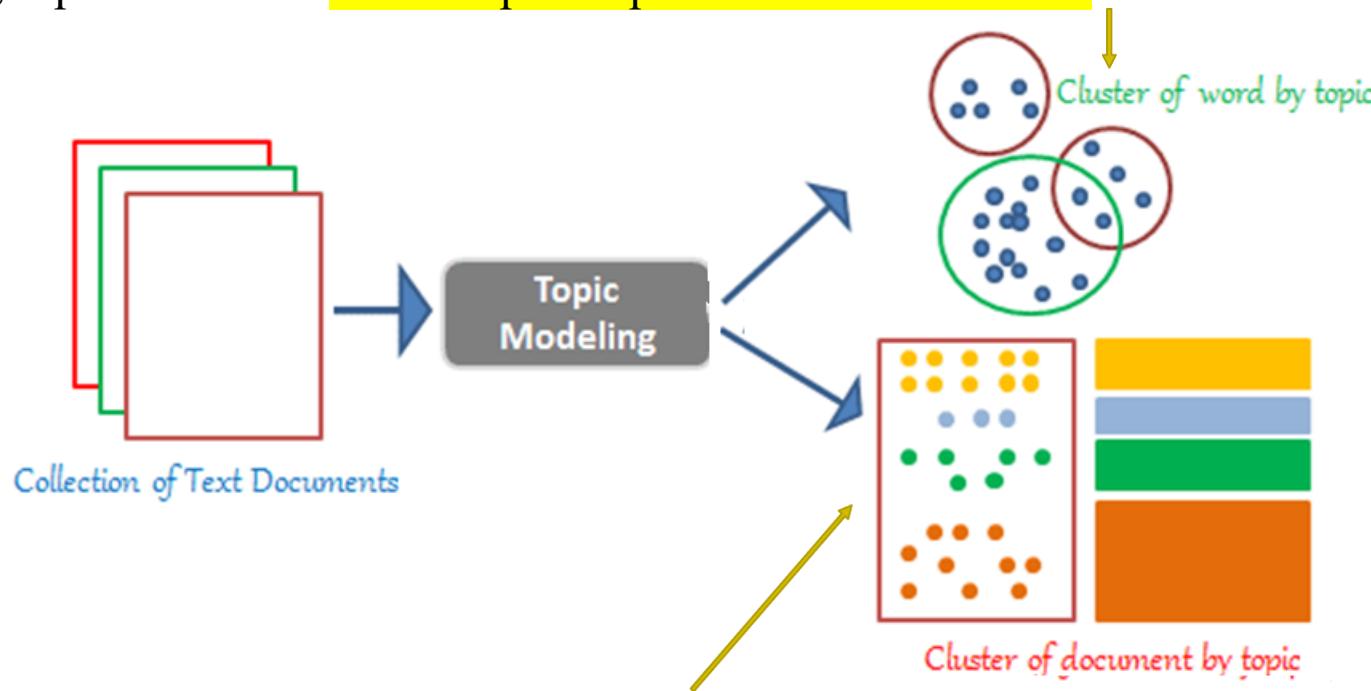
## 2. Les modèles du NLP

### Modèles statistiques de langage

36

#### Résultat :

- **Distribution de mots par sujet** : Chaque thème est caractérisé par une distribution de mots, représentant les mots les plus représentatifs de ce thème.



- **Distribution de sujets par document** : Chaque document est caractérisé par une distribution de thèmes, représentant la probabilité de présence de chaque thème.

## 2. Les modèles du NLP

# Modèles statistiques de langage

37

Cas d'utilisation réels de la découverte de thématiques (Topic Discovery)

- **Catégorisation et analyse des tendances médiatiques**
  - Classe automatiquement des milliers d'articles selon leurs thèmes (politique, sport, technologie...).
  - Permet aux journalistes et analystes de détecter les tendances émergentes et l'évolution de l'opinion publique.
- **Analyse des retours clients**
  - Extrait les sujets principaux à partir des avis produits ou enquêtes de satisfaction.
- **Recherche scientifique et veille documentaire**
  - Regroupe automatiquement les publications par domaine de recherche.
  - Facilite la revue de littérature et la découverte de liens interdisciplinaires.
- **Suivi des réseaux sociaux**
  - Analyse les tweets ou publications pour détecter les sujets populaires et les discussions émergentes.
  - Utile pour le marketing, les campagnes politiques ou la gestion de crise.
- **Gestion des connaissances en entreprise**
  - Organise les rapports, e-mails et comptes rendus selon des groupes thématiques cohérents.

→ Exemples d'outils : [LDA](#), [NMF](#), [BERTopic](#), [Top2Vec](#), [KeyBERT](#)

# 2. Les modèles du NLP

## Modèles statistiques de langage

38

### Python

```
# Importer les bibliothèques nécessaires
import gensim
from gensim import corpora
from pprint import pprint

# Exemple de corpus de documents (texte)
documents = [
    "Le chat aime chasser la souris",
    "Le chien aime jouer avec la balle",
    "Le chat et le chien peuvent être amis",
    "La souris se cache du chat",
    "Le chien aboie fort"
]

# Prétraitement du texte (tokenisation et suppression des mots vides)
stop_words = set('pour avec du le la et les des un une'.split())
texts = [[word for word in document.lower().split() if word not in stop_words] for document in documents]

# Création d'un dictionnaire (mapping de mots à des IDs)
dictionary = corpora.Dictionary(texts)

# Création de la matrice document-terme (corpus)
corpus = [dictionary.doc2bow(text) for text in texts]
```

# 2. Les modèles du NLP

## Modèles statistiques de langage

39

### Python

```
# Entraînement du modèle LDA
lda_model = gensim.models.LdaModel(corpus, num_topics=2, id2word=dictionary, passes=15)

# Affichage des thèmes et choix des noms
topics = lda_model.show_topics(formatted=False)
topic_names = {}

# Logique d'attribution des noms améliorée
for idx, topic in enumerate(topics):
    words = [word for word, _ in topic[1]]
    print(f"Thème {idx} mots dominants: {words}")
    if 'chat' in words[:5] or 'souris' in words[:5]:
        topic_names[idx] = 'Animaux domestiques - Chats'
    elif 'chien' in words[:5] or 'balle' in words[:5]:
        topic_names[idx] = 'Animaux domestiques - Chiens'
    else:
        topic_names[idx] = f'Thème {idx}'

# Affichage des thèmes nommés
for idx, topic in enumerate(topics):
    print(f"{topic_names[idx]}: {topic}")
```

## 2. Les modèles du NLP

# Modèles statistiques de langage

4C

# Python

```
# Affichage des proportions de thèmes dans chaque document avec noms
for idx, doc in enumerate(corpus):
    print(f"\nDocument {idx + 1}:")
    for topic, prob in lda_model.get_document_topics(doc):
        print(f"  {topic_names[topic]}: probabilité {prob:.4f}")
```

```
Thème 0 mots dominants: ['chat', 'souris', 'peuvent', 'être', 'amis', 'cache', 'se', 'chien', 'chasser', 'aime']
Thème 1 mots dominants: ['chien', 'aime', 'balle', 'jouer', 'aboie', 'fort', 'chasser', 'souris', 'chat', 'se']
Animaux domestiques - Chats: (0, [('chat', 0.18027444), ('souris', 0.1087963), ('peuvent', 0.087979704), ('être', 0.08797721), ('amis', 0.08797721), ('cache', 0.08797721), ('se', 0.08797721), ('chien', 0.08797721), ('chasser', 0.08797721), ('aime', 0.08797721)])
Animaux domestiques - Chiens: (1, [('chien', 0.14853112), ('aime', 0.14481926), ('balle', 0.08774567), ('jouer', 0.08774026), ('aboie', 0.08758491), ('fort', 0.08758491), ('chasser', 0.08758491), ('souris', 0.08758491), ('chat', 0.08758491), ('se', 0.08758491)])
Document 1:
    Animaux domestiques - Chats: probabilité 0.3759
    Animaux domestiques - Chiens: probabilité 0.6241

Document 2:
    Animaux domestiques - Chats: probabilité 0.1071
    Animaux domestiques - Chiens: probabilité 0.8929

Document 3:
    Animaux domestiques - Chats: probabilité 0.8978
    Animaux domestiques - Chiens: probabilité 0.1022

Document 4:
    Animaux domestiques - Chats: probabilité 0.8916
    Animaux domestiques - Chiens: probabilité 0.1084

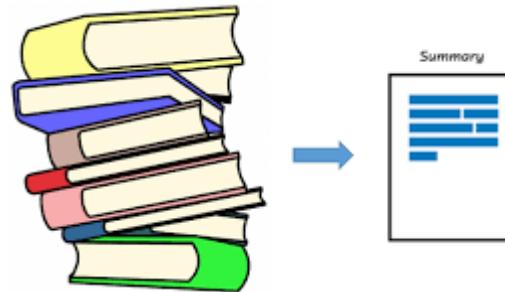
Document 5:
    Animaux domestiques - Chats: probabilité 0.1361
    Animaux domestiques - Chiens: probabilité 0.8639
```

## 2. Les modèles du NLP

### TextRank (Graphes de similarité)

41

- C'est un algorithme d'analyse de texte basé sur des **graphes** de **similarité**.
- Inspiré par l'algorithme PageRank de Google.
- Vise à **extraire des informations importantes** et à **identifier les éléments clés d'un texte** en modélisant les relations de similarité entre les mots où phrases.
- Il permet un **apprentissage non-supervisé**, basé sur la statistique d'apparition des mots
- Représentation du texte: distribution de probabilité



## 2. Les modèles du NLP

### TextRank (Graphes de similarité)

42

- Processus de création des graphes:
  - 1. Construction du graphe
    - Chaque unité textuelle (mot, phrase ou paragraphe) devient un nœud du graphe.
    - Des arêtes (liens) sont créées entre les nœuds selon leur relation de co-occurrence dans le texte.
    - Exemple : deux mots sont connectés s'ils apparaissent dans la même fenêtre de contexte (ex. 2–3 mots d'écart).
  - 2. Calcul des scores de similarité
    - Chaque arête reçoit un poids mesurant le degré de similarité entre les nœuds connectés.
    - Ces scores peuvent être calculés à l'aide de :
      - La fréquence de co-occurrence (plus deux mots apparaissent ensemble, plus leur lien est fort).
      - La similarité cosinus entre leurs vecteurs d'embedding (Word2Vec, TF-IDF, etc.).
      - Etc.
    - Le graphe devient donc pondéré, chaque arête représentant la force de relation sémantique entre les mots.

## 2. Les modèles du NLP

### TextRank (Graphes de similarité)

43

#### ○ 3. Calcul des scores de pertinence

- Chaque nœud reçoit un score de pertinence (ou d'importance) calculé de manière itérative.
- Le principe est similaire à PageRank :
  - La pertinence d'un nœud dépend de la somme pondérée des similarités des nœuds qui lui sont connectés.
- Ainsi, un mot connecté à plusieurs mots importants aura lui-même un score élevé.
- Les itérations continuent jusqu'à la stabilisation des poids (convergence).

#### ○ 4. Sélection des éléments clés

- Les nœuds ayant les scores les plus élevés sont sélectionnés comme :
  - Mots-clés dans le cas de l'extraction de termes importants.
  - Phrases clés dans le cas d'un résumé automatique.
- Ces nœuds représentent les concepts les plus centraux du texte, reflétant sa structure sémantique globale.

Exemples d'algorithmes basés sur ce principe : TextRank, LexRank, PositionRank, KeyGraph, etc.

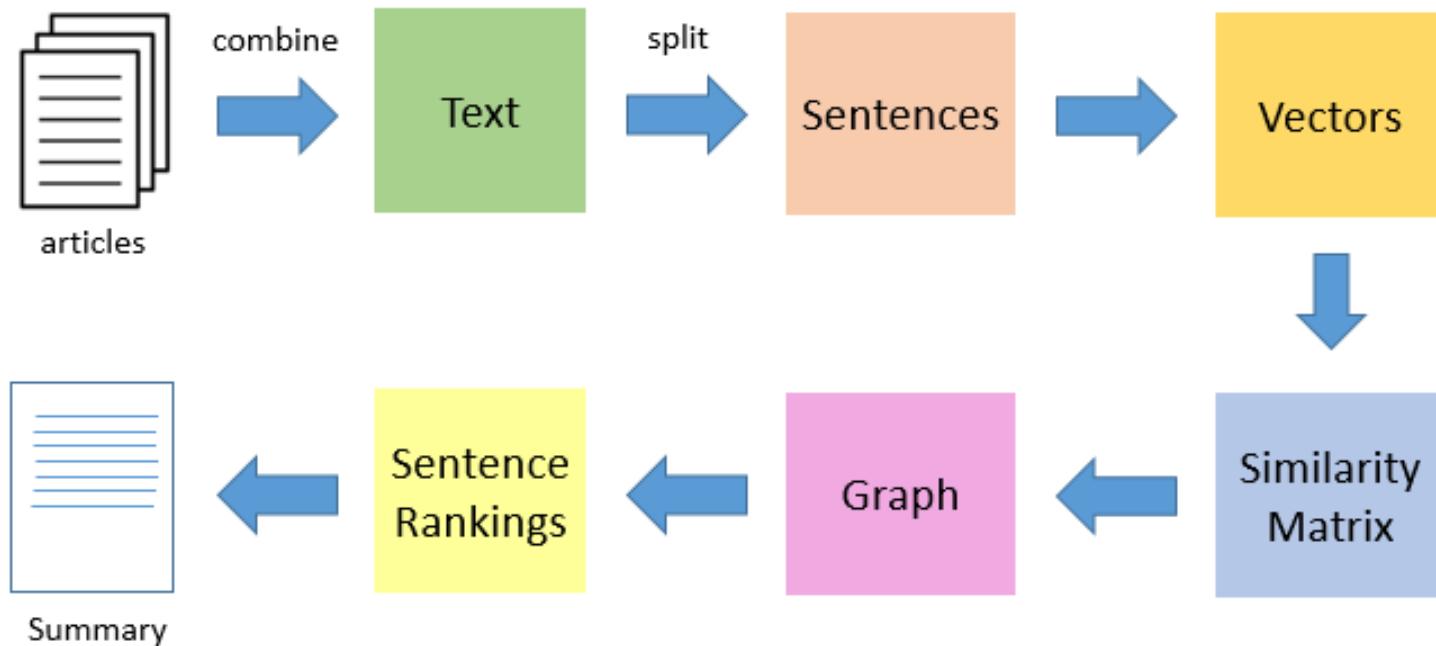
## 2. Les modèles du NLP

### TextRank (Graphes de similarité)

44

- **Applications de TextRank**

- **Résumé automatique** : il est souvent utilisé pour générer des résumés automatiques en identifiant les phrases les plus importantes d'un texte.



## 2. Les modèles du NLP

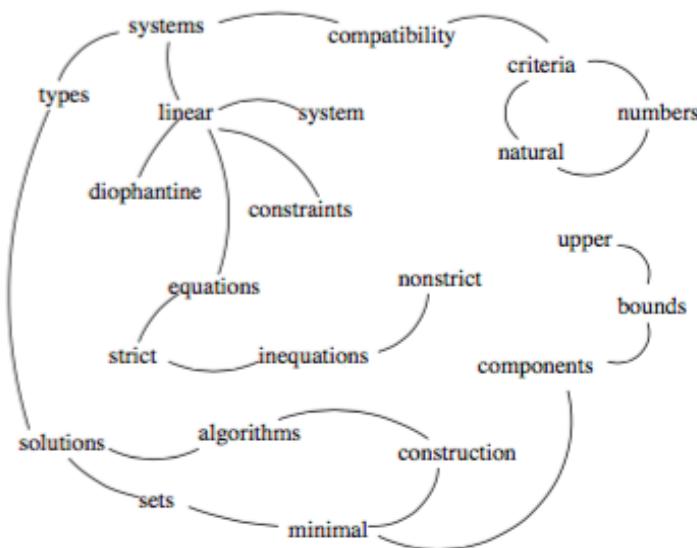
### TextRank (Graphes de similarité)

45

#### • Applications de TextRank

- Extraction de mots-clés : il peut extraire des mots-clés significatifs en identifiant les nœuds les plus pertinents dans le graphe.

Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types systems and systems of mixed types.



#### Keywords assigned by TextRank:

linear constraints; linear diophantine equations; natural numbers; nonstrict inequations; strict inequations; upper bounds

#### Keywords assigned by human annotators:

linear constraints; linear diophantine equations; minimal generating sets; non-strict inequations; set of natural numbers; strict inequations; upper bounds

#### Exemple de fonctionnement de TextRank

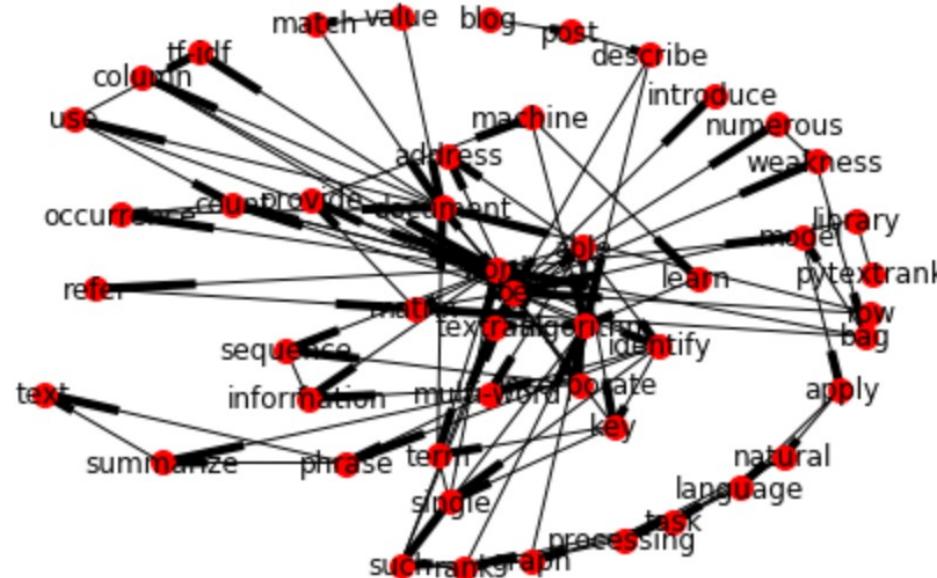
## 2. Les modèles du NLP

### TextRank (Graphes de similarité)

46

- **Applications de TextRank**

- **Analyse de similarité** : il peut être utilisé pour mesurer la similarité entre différents textes en comparant les structures de leurs graphes.



## 2. Les modèles du NLP

# TextRank (Graphes de similarité)

47

### Travail à faire pour la séance prochaine :

- **Objectif** : Implémenter en Python le modèle TextRank pour l'une des applications ci-dessous.

#### 1. Résumé automatique :

1. Implémenter TextRank pour générer un résumé automatique à partir d'un texte long.
2. L'algorithme devra identifier les phrases les plus importantes en calculant leur importance relative dans le texte (comme des "nœuds" dans un graphe) et produire un résumé constitué de ces phrases clés.

#### 3. Livrables attendus :

1. Le script Python de votre implémentation.
2. Un fichier contenant le texte original et son résumé automatique généré par votre modèle.

#### 2. Extraction de mots-clés :

1. Modéliser TextRank pour extraire des mots-clés significatifs d'un texte.
2. L'algorithme devra identifier les nœuds les plus pertinents dans le graphe de mots en se basant sur la co-occurrence de ces derniers dans des fenêtres de phrases ou de contextes.

#### 3. Livrables attendus :

1. Le script Python pour l'extraction des mots-clés.
2. Une liste de mots-clés extraits du texte testé. .

## 2. Les modèles du NLP

# TextRank (Graphes de similarité)

48

### Travail à faire pour la séance prochaine :

#### 1. Analyse de similarité :

1. Utiliser TextRank pour mesurer la similarité entre plusieurs textes en comparant leurs structures de graphes.
2. Vous devez construire un graphe de phrases ou de documents pour chacun des textes, puis comparer les graphes pour calculer leur degré de similarité.

#### 3. Livrables attendus :

1. Le code Python qui mesure la similarité entre les textes.
2. Un rapport expliquant les résultats de la similarité entre les documents testés.

## 2. Les modèles du NLP

### Modèle de langage-Introduction

49

#### Objectif:

- Construire un modèle de langue statistique vise à estimer une **distribution de probabilité sur les séquences de mots** dans une langue donnée. Cela permet au modèle d'assigner une probabilité spécifique à chaque séquence de mots, indiquant la probabilité qu'une phrase ou un texte donné apparaisse dans cette langue.
- Les modèles de langue statistique ne reposent pas sur des **règles formelles** mais sur des **statistiques** apprises à partir d'un vaste corpus de textes.
- En analysant un grand volume de données, ces modèles peuvent reconnaître et prédire des schémas linguistiques basés sur la **fréquence** et la **probabilité d'apparition** de certaines séquences de mots.
- A quoi sert une distribution de probabilité sur une langue?
- Comment concevoir une telle distribution?
- Comment la construire?
- Comment la tester et évaluer sa performance?
- ...



## 2. Les modèles du NLP

### Modèle de langage-Introduction

50

#### Probabilité d'une phrase :

- La probabilité d'une phrase  $S$ , composée de mots  $w_1, w_2, \dots, w_n$ , est exprimée par la fonction :

$$P(S) = P(w_1, w_2, \dots, w_n)$$

- Traduction automatique :
  - My tall brother → P(Mon grand frère) > P(Mon haut frère)
  - Le modèle de langue choisit la traduction la plus probable en fonction du contexte des mots.
- Correction des fautes grammaticales :
  - P(Un objet qu'on puisse emporter) > P(Un objet qu'ont puisse emporter)
  - En analysant les séquences probables, le modèle identifie les erreurs syntaxiques et grammaticales, proposant ainsi la correction la plus plausible.
- Reconnaissance de paroles :
  - P(Jeudi matin) > P(Je dis matin)
  - Le modèle utilise le contexte pour distinguer entre des mots phoniquement proches et déterminer la séquence la plus probable.

## 2. Les modèles du NLP

### Modèle de langage-Introduction

51

- **Probabilité d'occurrence d'un mot  $w_i$  :**  $P(w_i | w_1, \dots, w_{i-1})$ 
  - Auto-compléTION :
    - P(traitement automatique de l'information) > P(traitement automatique de l'eau)
    - L'auto-compléTION utilise la probabilité d'occurrence de mots dans un contexte donné pour suggérer des séquences cohérentes.
  - Génération automatique de textes
    - Création de phrases, paragraphes, et textes cohérents à partir de phrases de départ ou phrases initiales.
    - Le modèle génère du texte en prédisant **les mots les plus probables** dans une séquence, permettant la production de **contenu** linguistiquement pertinent et fluide.

## 2. Les modèles du NLP

### Modèle de langage

52

#### Cas d'utilisation des n-grammes :

- **Modélisation de langage**: ils aident à prédire la probabilité d'une séquence de mots, ce qui est essentiel dans les systèmes de génération de texte ou de correction grammaticale.
- **Analyse de sentiment**: ils permettent d'extraire des caractéristiques pertinentes des textes pour évaluer les sentiments exprimés dans les avis ou les commentaires.
- **Classification du texte**: En utilisant des n-grammes comme caractéristiques, on peut classifier des documents dans des catégories, comme le spam ou le non-spam.
- **Recherche d'information** : Les n-grammes sont utilisés pour améliorer les systèmes de recherche en permettant une meilleure correspondance entre les requêtes et les documents, en particulier pour les recherches par phrase.
- **Détection de plagiat**: En comparant les n-grammes d'un texte suspect avec ceux d'un texte de référence, il est possible d'identifier des similarités significatives.

## 2. Les modèles du NLP

### Modèle de langage

53

- **Recommandation de mots:** dans les systèmes de saisie semi-automatique, les n-grammes aident à prédire les mots suivants basés sur les mots précédents saisis par l'utilisateur.
- **Traducteurs automatiques:** ils peuvent être utilisés pour établir des correspondances entre des séquences de mots dans différentes langues.
- **Extraction d'informations:** ils facilitent l'identification de patterns ou d'expressions fréquentes dans un corpus, utile pour l'extraction d'entités nommées.
- **Résumés automatiques:** Ils aident à identifier les phrases les plus pertinentes d'un texte à inclure dans un résumé.

## 2. Les modèles du NLP

### Modèle de langage

54

- Pour évaluer la **probabilité d'une séquence** de N mots, on se base sur la fréquence d'occurrence de ces séquences dans un vaste corpus de texte. L'objectif est de déterminer la **probabilité de chaque mot en fonction des mots précédents** pour produire des séquences linguistiquement naturelles.

**Formule des probabilités composées:**

- La probabilité d'une séquence de mots  $w_1, w_2, \dots, w_m$  peut être représentée par le produit des probabilités conditionnelles successives :

$$P(w_1 \dots w_m) = P(w_1 \dots w_{m-1}) \times P(w_m | w_1 \dots w_{m-1}) =$$

**Règle de la chaîne :**  $P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \dots P(w_n|w_1, \dots, w_{n-1})$

- Exemple de probabilité d'une phrase:**

$$P(\text{je travaille à l'EPI}) = P(\text{je})P(\text{travaille}| \text{je})P(\text{à}| \text{je travaille}) \dots P(\text{EPI}| \text{je travaille à l'})$$

- $P(\text{EPI}| \text{je travaille à l'}) = \frac{C(\text{je travaille à l'EPI})}{C(\text{je travaille à l'})}$ , où
  - C est une fonction de comptage qui représente le **nombre d'occurrences** d'une séquence donnée dans le corpus d'entraînement.

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

55

- Il est quasiment impossible de calculer directement pour toutes les séquences jointes  $P(S) = P(w_1 w_2, \dots, w_m)$ , pour toute séquence  $S=(w_1 w_2, \dots, w_m)$  du corpus.

#### → Hypothèse de Markov:

- Repose sur le principe que l'état futur dépend uniquement de l'état actuel, et non de l'historique complet des états passés.
- Dans le cadre d'un modèle de langue, cela signifie que la probabilité d'un mot ne dépend que des derniers mots, plutôt que de toute la séquence précédente.
- Propriété de Markov simplifiée (Premier Ordre)
  - Un état futur ne dépend que de l'état présent.
    - $P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-1})$
  - Cas général avec  $n - 1$  états passés (n-grammes)
    - Un *n-gram* est une séquence de  $n$  mots consécutifs trouvée dans un corpus de texte.
    - Dans le cas des modèles de **N-grammes** (modèles à  $n$  états), la probabilité d'apparition d'un mot est déterminée par les  $n - 1$  mots précédents, soit :
      - $P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-n+1}, \dots, w_{i-1})$

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

56

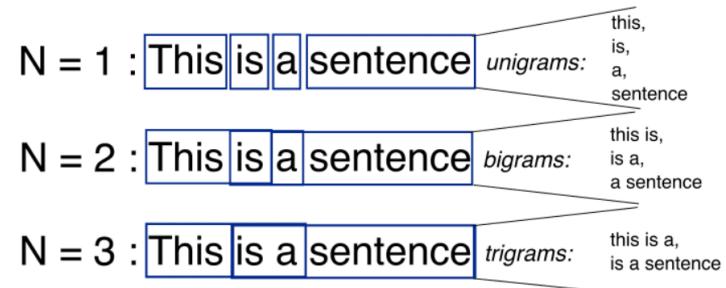
- **Estimation de probabilité en utilisant les N-grammes**
- Dans un modèle de langue N-gramme, la probabilité d'une séquence de mots  $w_1, \dots, w_m$  est approximée par le produit des probabilités conditionnelles basées sur les **n-1** mots précédents :

$$P(w_1, \dots, w_m) \approx \prod_{i=1}^m P(w_i | w_{i-k+1}, \dots, w_{i-1})$$

Où k correspond à l'ordre du modèle N-gramme, déterminant combien de mots précédents influencent la probabilité de chaque mot.

#### Quelque modèles N-gramme

- **Modèle uni-gramme** :  $P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i)$ 
  - Chaque mot est considéré individuellement
- **Modèle bi-gramme** :  $P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-1})$ 
  - Le contexte d'un mot est défini par le mot qui le précède
- **Modèle tri-gramme** :  $P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-2}, w_{i-1})$ 
  - Le contexte d'un mot est défini par le couple de mots qui le précédent

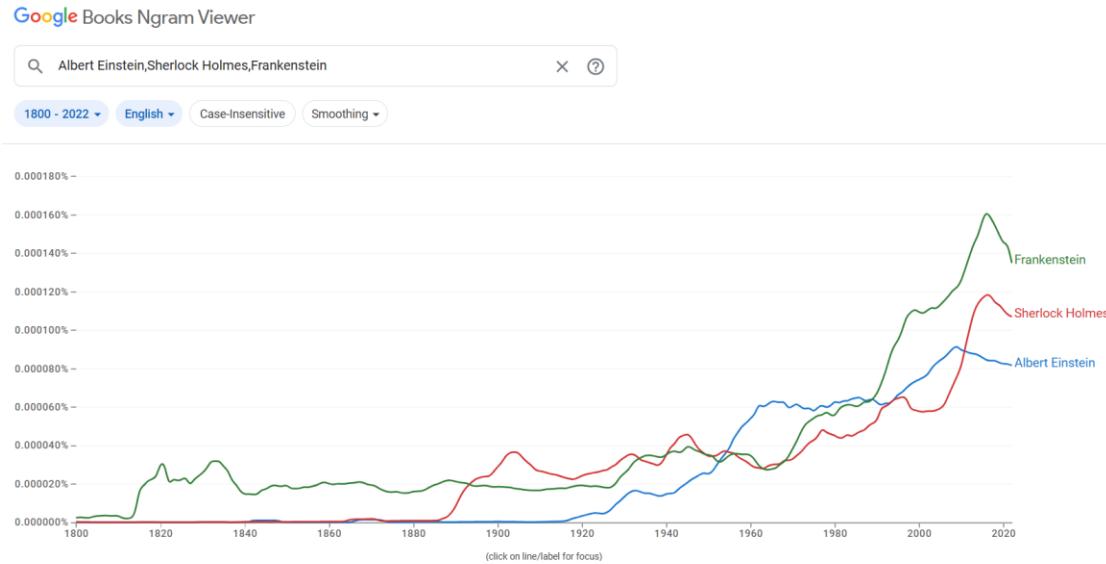


## 2. Les modèles du NLP

# Modèle de langage-N-gramme

57

- **Google Books Ngram Viewer**
  - <https://books.google.com/ngrams>
  - Modèles pré-traitées à partir des livres
  - Téléchargement gratuit : <https://storage.googleapis.com/books/ngrams/books/datasetsv3.html>
- **Objectif des n-grams :**
  - Permet d'observer la popularité de mots ou expressions dans un large corpus de livres au fil du temps.
- **Pics:** liés aux publications, adaptations médiatiques, ou événements historiques.
- **Limites :** mesure uniquement la fréquence dans les livres, sans prendre en compte l'importance contextuelle.



Évolution de la fréquence des termes "Albert Einstein," "Sherlock Holmes," et "Frankenstein" dans les livres (1800-2022)

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

58

#### Formulation: Estimation des Probabilités des N-grammes

- **Principe:**

- Utilisation d'un corpus d'entraînement large pour calculer les probabilités de cooccurrence des séquences de mots.
- Début et fin des phrases marqués par `< s >` et `< /s >` pour assurer que les n-grammes capturent les contextes de phrase complets (1 marqueur pour les bi-grammes, 2 pour les tri-grammes, etc.).

- **Estimateur du maximum de vraisemblance (MLE):**

- La probabilité d'apparition d'un mot  $w_i$  après une séquence  $w_{i-n+1} \dots w_{i-1}$  est estimée par la formule :

$$P(w_i | w_{i-n+1}, \dots, w_{i-1}) = \frac{C(w_{i-n+1} \dots w_{i-1} w_i)}{\sum_i C(w_{i-n+1} \dots w_{i-1} w_i)} = \frac{C(w_{i-n+1} \dots w_{i-1} w_i)}{C(w_{i-n+1} \dots w_{i-1})}$$

Où  $C$  est le nombre d'occurrences des N-grammes dans le corpus

$$\text{Bi-grammes : } P(w_i | w_{i-1}) = \frac{C(w_{i-1} w_i)}{C(w_{i-1})}$$

- Par convention pour le dénominateur,  $C(\emptyset) = \text{nombre}_{total} \text{ des mots dans le corpus}_{train}$

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

59

- C'est une **estimation de maximum de vraisemblance** → Faire en sorte que les séquences les plus fréquentes dans le corpus aient une probabilité maximale dans le modèle.
- En particulier, si à chaque fois, une séquence  $w_1, \dots, w_{m-1}$  est suivie de  $w_m$ , alors  $P(w_m | w_1, \dots, w_{m-1}) = 1$ .

#### Exemple d'un corpus d'entraînement

- <s>un ordinateur peut vous aider </s>
- <s>il veut vous aider </s>
- <s>il veut un ordinateur </s>
- <s>il peut nager </s>

$$\bullet P(\text{peut}|il) = \frac{C(il \text{ peut})}{C(il)} = \frac{1}{3}$$

$$\bullet P(<\text{s}>il \text{ peut vous aider } </\text{s}>) = \\ \underbrace{P(il|<\text{s}>)}_{\frac{3}{4}} \underbrace{P(\text{peut}|il)}_{\frac{1}{3}} \underbrace{P(\text{vous}|\text{peut})}_{\frac{1}{2}} \underbrace{P(\text{aider}|\text{vous})}_{\frac{2}{2}} \underbrace{P(</\text{s}>|\text{aider})}_{\frac{2}{2}} = \frac{1}{8}$$

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

60

#### Limites:

- Si on utilise des petits N-grammes ⇒ Perte de l'information
  - Les langues permettent des dépendances à long terme.
  - L'ordinateur que j'ai utilisé hier à l'EPI pendant la séance du cours a planté
- Si on utilise des grands N-grammes ⇒ Complexité élevée du modèle
  - Il faut un corpus plus grand.
  - Représentation des N-grammes :  $V^N$  où  $V$  est la taille du vocabulaire et  $N$  est le nombre de grammes.
- Problème des N-grammes absents dans le corpus d'entraînement

$$P(< \text{s} > \text{il veut nager} < / \text{s} >) = \\ P(\text{il} | < \text{s} >) P(\text{veut} | \text{il}) P(\text{nager} | \text{veut}) P(< / \text{s} > | \text{nager}) = \frac{3}{4} \frac{2}{3} \frac{0}{1} \frac{1}{1} = 0$$

#### Solution → Lissage (Smoothing)

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

61

#### Smoothing :

- Lissage de Lidstone (Bi-grammes comme exemple)

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1}w_i) + \sigma}{C(w_{i-1}) + \sigma V}$$

Où V est la taille du vocabulaire du modèle

- $\alpha = 1$  : Lissage de Laplace
- $\alpha = 0.5$  : Loi de Jeffreys-Perks

#### Exemple : lissage de Laplace

- Le corpus contient 8 mots différents
- $|V| = 8 + 10 = 10$  (on compte les marqueurs de début et de fin)
- $P(< s > il \text{ veut nager } < /s >) =$

$$P(il|<s>)P(veut|il)P(nager|veut)P(</s>|nager) = \frac{3+1}{4+10} \frac{2+1}{3+10} \frac{0+1}{1+10} \frac{1+1}{1+10}$$

- Lisseur: Interpolation, Back-off de Katz,...

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

62

#### Python

```
# Importation des bibliothèques nécessaires
import nltk
from nltk import trigrams
from collections import defaultdict
import re

# Téléchargement des ressources nécessaires pour NLTK
nltk.download('punkt')

# Corpus d'exemple
corpus = """
La vie est belle. La vie est une aventure. Belle est la vie.
L'aventure est belle quand elle est bien vécue.
"""

# Prétraitement du corpus
corpus = re.sub(r'[^w\s]', '', corpus).lower()
words = nltk.word_tokenize(corpus)

# Création des trigrams
tri_grams = list(trigrams(words))

# Création du modèle de trigrammes avec des dictionnaires imbriqués
model = defaultdict(lambda: defaultdict(lambda: 0))

# Comptage de la fréquence d'occurrence des trigrams
for w1, w2, w3 in tri_grams:
    model[(w1, w2)][w3] += 1
```

utilise une expression régulière pour conserver uniquement les mots (lettres et chiffres, représentés par \w) et les espaces (\s).

Cela permet de construire un modèle probabiliste où chaque paire de mots ( $w_1, w_2$ ) contient un dictionnaire des mots  $w_3$  possibles, ainsi que leur fréquence d'apparition après cette paire de mots dans le texte.

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

Python



```
# Conversion des fréquences en probabilités
for w1_w2 in model:
    total_count = float(sum(model[w1_w2].values()))
    for w3 in model[w1_w2]:
        model[w1_w2][w3] /= total_count

# Fonction pour prédire le prochain mot
def predict_next_word(w1, w2):
    """
    Prédit le mot suivant en fonction des deux mots précédents, en utilisant le modèle trigramme.

    Args:
        w1 (str): Le premier mot.
        w2 (str): Le deuxième mot.

    Returns:
        str: Le mot prédit ou un message si aucune prédiction n'est disponible.
    """
    next_word = model[(w1, w2)]
    if next_word:
        predicted_word = max(next_word, key=next_word.get) # Choix du mot avec la probabilité la plus élevée
        return predicted_word
    else:
        return "Aucune prédiction disponible"

# Exemple d'utilisation
print("Mot suivant :", predict_next_word('la', 'vie'))
print("Mot suivant :", predict_next_word('est', 'belle'))
print("Mot suivant :", predict_next_word('belle', 'la'))
```

un dictionnaire où les clés sont les mots candidats pouvant suivre la paire (w1, w2) et les valeurs sont les probabilités associées à chaque mot.

Args:

w1 (str): Le premier mot.  
w2 (str): Le deuxième mot.

Returns:

str: Le mot prédit ou un message si aucune prédiction n'est disponible.

"""

```
next_word = model[(w1, w2)]
if next_word:
    predicted_word = max(next_word, key=next_word.get) # Choix du mot avec la probabilité la plus élevée
    return predicted_word
else:
    return "Aucune prédiction disponible"
```

pour obtenir la probabilité de chaque mot

Mot suivant : est  
Mot suivant : la  
Mot suivant : vie

## 2. Les modèles du NLP

### Modèle de langage-N-gramme

64

#### Limites de lissage:

- **Perte d'information** : attribution de probabilités non nulles à des événements non observés, entraînant une perte de précision.
- **Sensibilité aux données** : sa précision dépend de la taille et de la représentativité du corpus d'entraînement.
- **Choix du paramètre** : nécessite un réglage délicat du coefficient de lissage, avec un risque d'estimations inappropriées.
- **Non-adaptabilité aux contextes** : il traite tous les contextes de manière uniforme, sans tenir compte des spécificités des différents contextes linguistiques.
- **Moins efficace pour les grands N-grammes** : peu adapté aux N-grammes de grande taille, nécessitant des données massives pour des estimations précises.

**Solution → Modèles neuronaux**

## 2. Les modèles du NLP

### Modèle de langage-Modèles neuronaux

65

- **Objectif:** tout comme les modèles n-grammes, l'objectif est **d'estimer la probabilité d'une séquence de mots** dans une langue donnée.
- Ils exploitent les réseaux de neurones récurrents (RNN), en particulier les **LSTM\***:
  - sont capables de **conserver la mémoire** du traitement des mots précédents dans une séquence.  
→ Cela permet **d'intégrer le mot à prédire dans son contexte**, améliorant ainsi la compréhension du modèle.

*Exemple de prédiction d'un mot dans le cas d'homophones en reconnaissance vocale.*

*Elle adore chanter mais elle perd sa \_\_ [voie, voix]*

*Elle adore chanter mais elle perd sa **voix***

\*Long Short-Term Memory

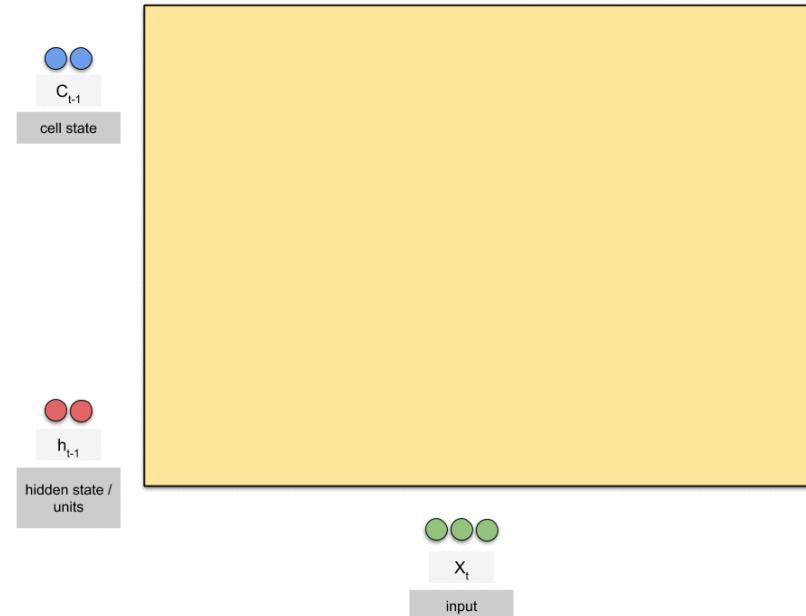
## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

66

### A. Réseau de neurones RNNs

- Ces réseaux sont un type spécial de réseau de neurones conçu pour travailler avec des données séquentielles, comme le langage naturel.
- **Point fort:** leur capacité à gérer les dépendances à long terme dans les séquences.  
→ Ils sont capables de retenir des informations importantes sur une séquence sur une longue période.



## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

67

### A. Réseau de neurones RNNs

#### Etapes:

##### 1. Représentation Vectorielle des Mots:

- Les séquences de mots sont traduites en **vecteurs de mots denses** (word vectors/ word embedding) avant d'être injectées dans le réseau.
  - ▣ où des mots conceptuellement similaires sont proches l'un de l'autre.
  - ▣ Exemples de modèles pour gérer des vecteurs de mots: Word2Vec et GloVe.

##### 2. Apprentissage des Relations et du Sens

- Pendant l'entraînement, le modèle **apprend à ajuster les word embeddings** en fonction des contextes dans lesquels les mots apparaissent.
- Les **relations sémantiques** et **syntaxiques** entre les mots sont apprises automatiquement à mesure que le modèle est exposé à un large ensemble de données d'entraînement.

→ Le modèle **apprend à généraliser** à partir de contextes similaires.

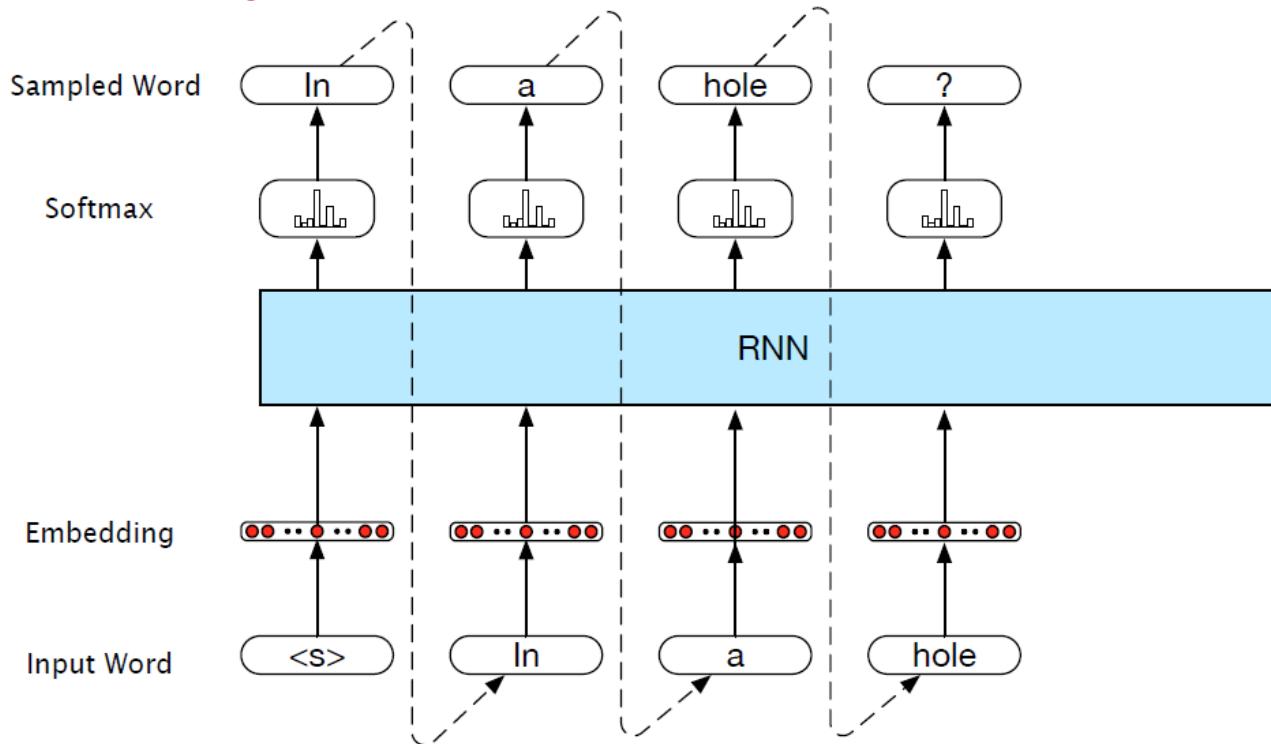
## 2. Les modèles du NLP

### Modèle de langage-Modèles neuronaux

68

#### A. Réseau de neurones RNNs

Cas1: Génération de la langue



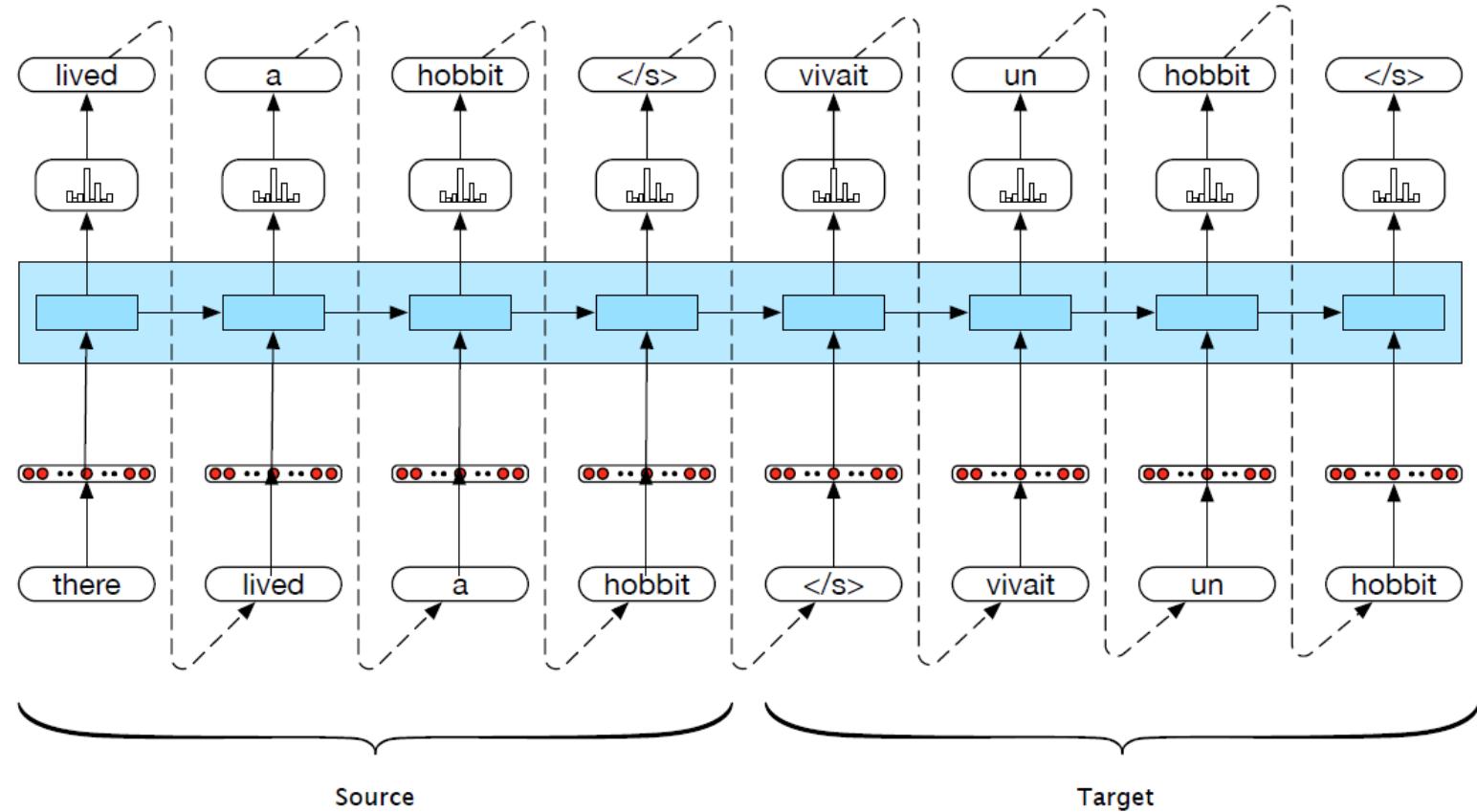
## 2. Les modèles du NLP

### Modèle de langage-Modèles neuronaux

69

#### A. Réseau de neurones RNNs

##### Cas2: Traduction automatique



## 2. Les modèles du NLP

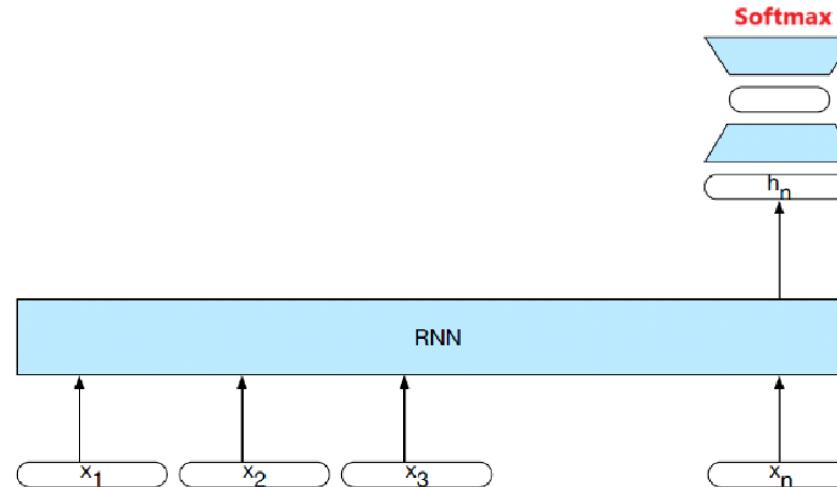
### Modèle de langage-Modèles neuronaux

70

#### A. Réseau de neurones RNNs

##### Cas3: Classification de séquences

- Lorsque l'objectif est **d'attribuer une seule étiquette à l'ensemble d'une séquence**, il n'est pas nécessaire de produire une sortie à chaque étape temporelle.
- Dans ce cas, nous pouvons adopter une architecture plus simple.
- Pour cela, nous pouvons utiliser **l'état caché du dernier mot de la séquence** comme entrée pour un réseau de neurones à propagation avant (feedforward network).  
→ Cette approche permet d'optimiser le processus de classification en se concentrant sur la représentation finale de la séquence, ce qui est souvent suffisant pour déterminer l'étiquette appropriée.



## 2. Les modèles du NLP

### Modèle de langage-Modèles neuronaux

71

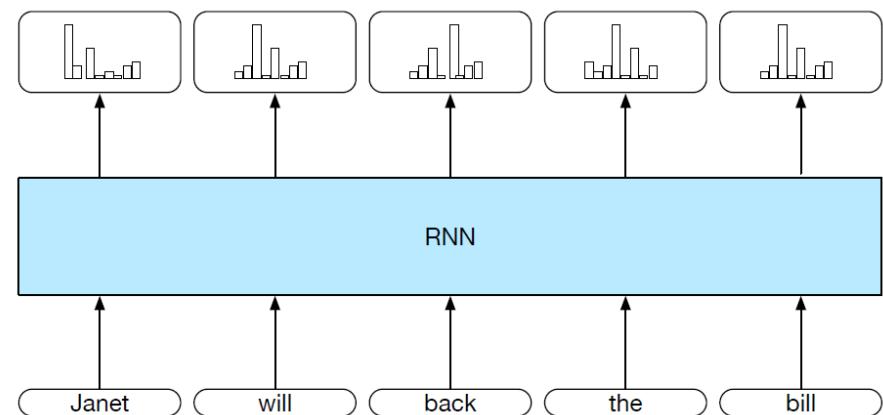
#### A. Réseau de neurones RNNs

Cas4: Étiquetage de séquences (par exemple, étiquetage POS) :

- Attribuer une étiquette à chaque élément de la séquence.
- **Architecture des RNN**
  - Chaque étape temporelle possède une **distribution sur les classes de sortie**.
  - L'**étiquette finale** attribuée à chaque élément de la séquence sera celle avec la probabilité la plus élevée dans cette distribution.
- **Extension:**
  - Ajouter une couche **CRF** (Conditional Random Field) pour capturer les dépendances entre les étiquettes des tokens adjacents.

Une couche CRF:

- Optimise la séquence de labels en tenant compte des dépendances entre étiquettes successives, ce qui garantit une **cohérence** dans les prédictions de séquençage.
- Souvent appliquée après des couches LSTM ou CNN pour améliorer les résultats dans des tâches comme la reconnaissance d'entités ou l'étiquetage de parties du discours.



## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

72

### A. Réseau de neurones RNNs

- Limitations des Modèles RNNs:
  - Les RNN nécessitent un vaste corpus d'entraînement pour de bonnes performances, mais l'augmentation des données peut **réduire la sémantique** contextuelle.
  - Temps de traitement élevé : en raison du traitement séquentiel, les RNN sont lents et difficilement parallélisables, ce qui augmente le temps de calcul.
  - Biais en fin de séquence : le modèle capte davantage le **contexte en fin de séquence qu'au début**, ce qui peut créer un déséquilibre.
  - Poids sur l'historique proche : les RNN privilégient les informations récentes, limitant l'influence des mots plus anciens dans la séquence.
- **Solution** : Le architectures Encoder/Decoder

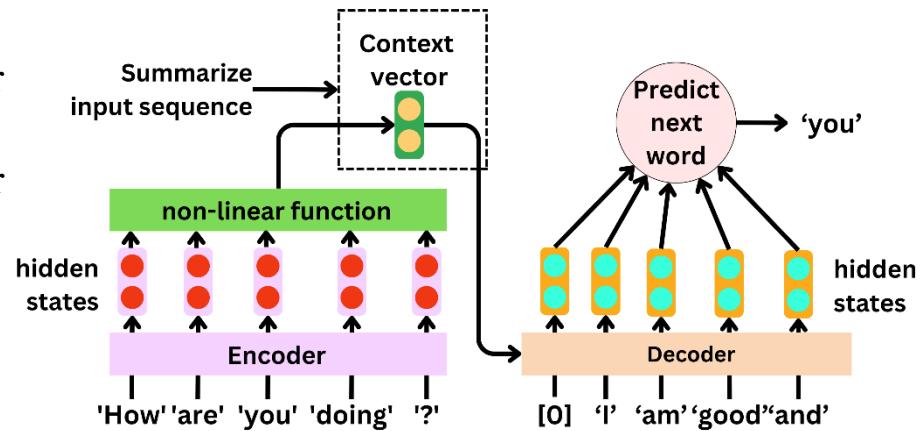
## 2. Les modèles du NLP

### Modèle de langage-Modèles neuronaux

73

#### B. Modèle Encodeur-Décodeur (seq2seq)

- Composants : Architecture encodeur-décodeur récurrent
  - L'encodeur peut être bidirectionnel, capturant ainsi le contexte avant et après chaque mot de la séquence.
  - l'encodeur utilise des états cachés pour encoder les informations de la séquence d'entrée, et le décodeur utilise ces états pour générer la sortie.
- a) L'encodeur: un réseau de neurones utilisé pour transformer la séquence d'entrée en une représentation vectorielle.
  - b) Mécanisme d'attention (simple): aide le décodeur à se concentrer sur des parties spécifiques de la séquence d'entrée (résultantes de la partie encoder) lors de la génération de chaque mot, en évitant de résumer toute la séquence en un seul vecteur.
    - Génère un **vecteur de contexte** pour chaque mot.
  - c) Le décodeur: utilise le vecteur de contexte pour générer la séquence de sortie mot par mot.



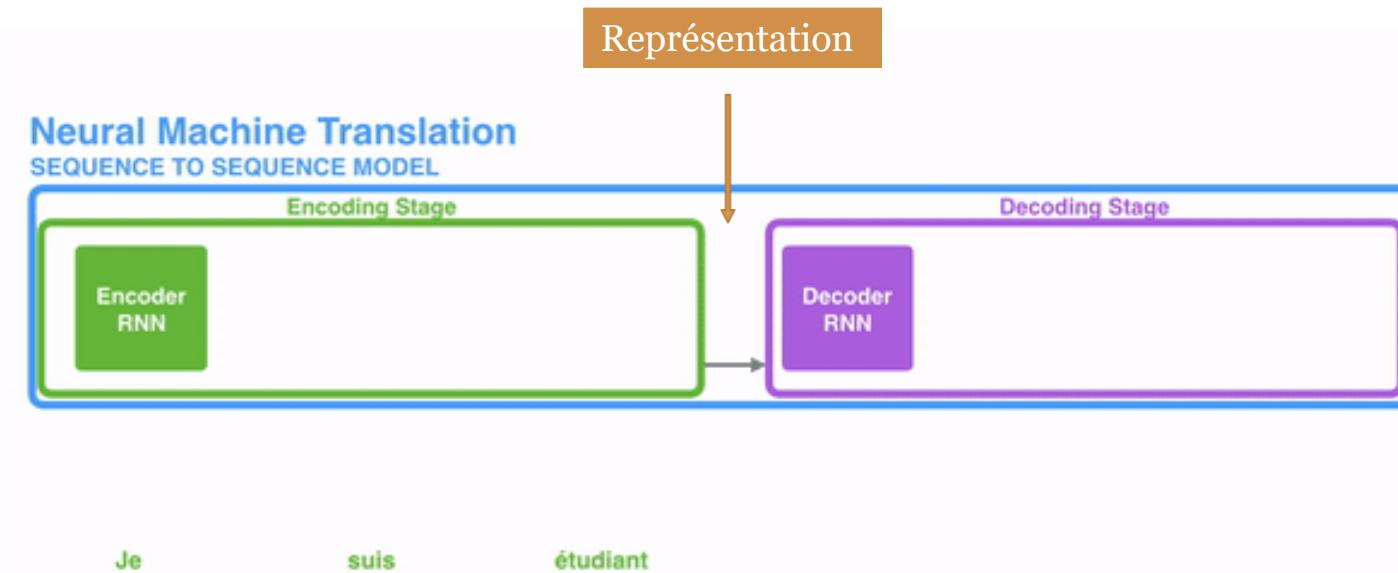
## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

74

### B. Modèle Encodeur-Décodeur (encodeur-décodeur récurrent - seq2seq)

- Dans le cas particulier de la **traduction automatique**:
  - **L'encodeur** transforme le texte source en vecteur qui est ensuite donné en entrée au décodeur.
  - **Le décodeur** génère alors la séquence correspondante dans la langue cible.



## 2. Les modèles du NLP

### Modèle de langage-Modèles neuronaux

75

#### C. Les Transformers

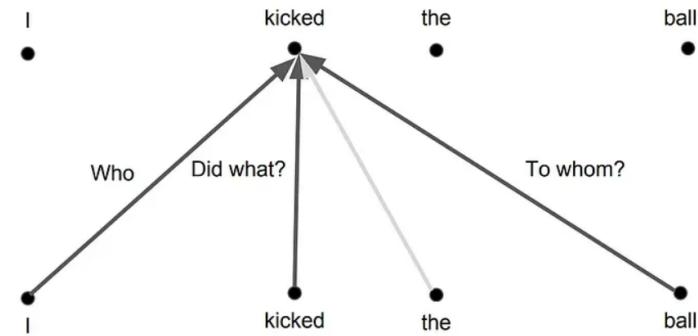
- Sont une architecture de RNN qui ont révolutionné le NLP en permettant aux modèles de comprendre **les relations entre les mots sans dépendre de l'ordre séquentiel** (comme dans les RNNs).
- Architecture **encodeur-décodeur non récurrent** :
  - Pas d'états cachés
  - Contexte capturé grâce à l'attention et aux encodages de position
  - Constitué de couches empilées avec différentes sous-couches

#### Attention

Les Transformers utilisent une version avancée appelée **Multi-Head Self-Attention**, qui permet au modèle de mieux comprendre les relations entre les mots d'une séquence.

##### 1. Self-Attention:

- Chaque mot de la séquence est comparé à tous les autres mots pour en calculer l'**importance relative**.  
→ Cela permet de capturer des dépendances de longue portée et d'améliorer la représentation contextuelle des mots.



## 2. Les modèles du NLP

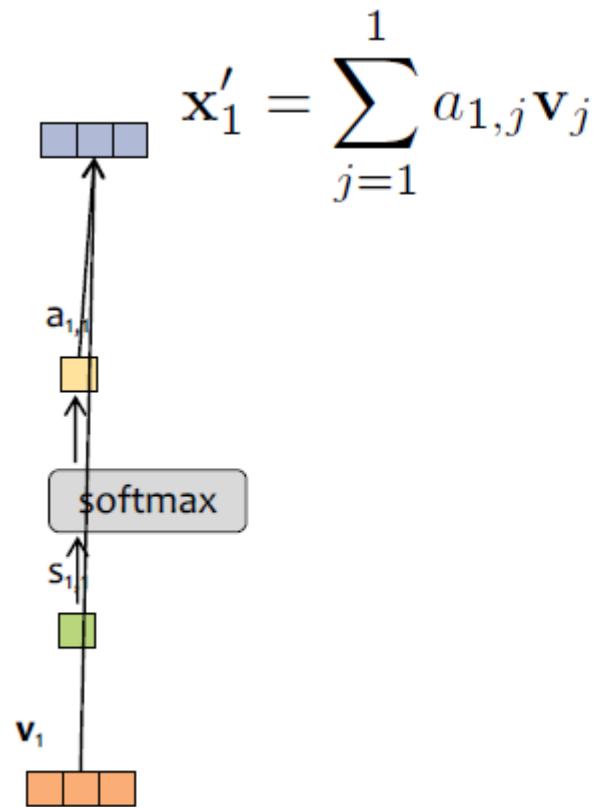
# Modèle de langage-Modèles neuronaux

76

### C. Les Transformers

#### Attention

##### 1. Self-Attention:



## 2. Les modèles du NLP

### Modèle de langage-Modèles neuronaux

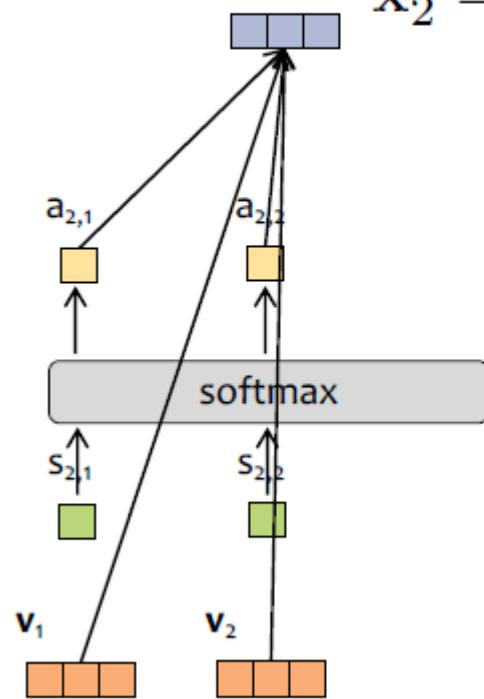
77

#### C. Les Transformers

##### Attention

###### 1. Self-Attention:

$$\mathbf{x}'_2 = \sum_{j=1}^2 a_{2,j} \mathbf{v}_j$$



## 2. Les modèles du NLP

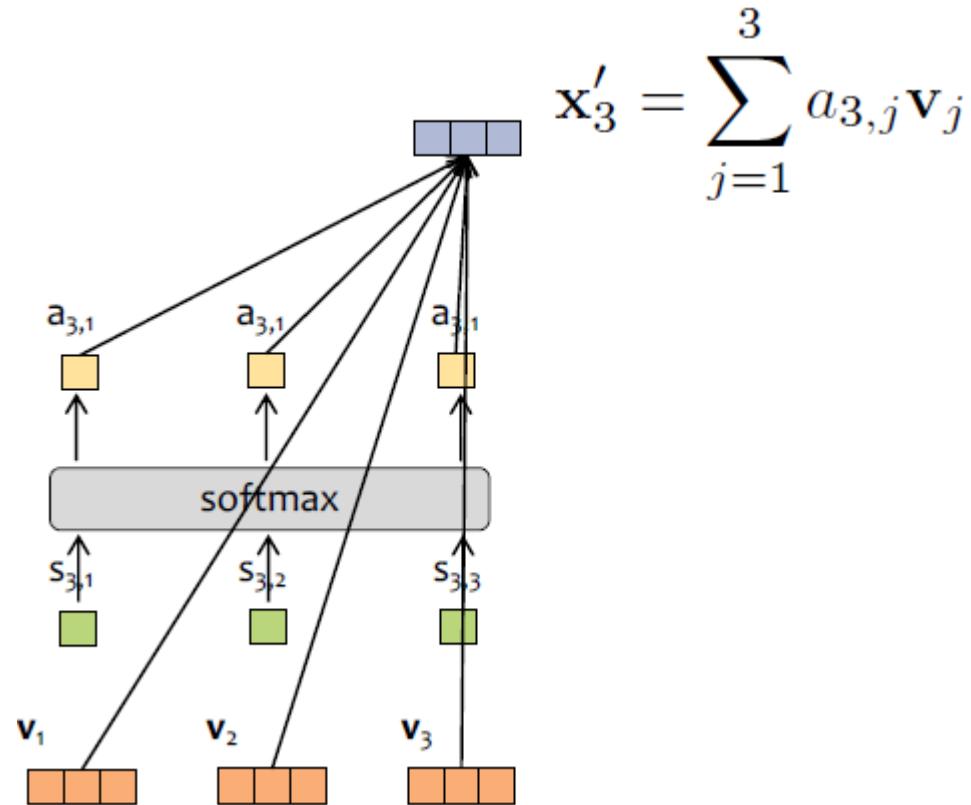
### Modèle de langage-Modèles neuronaux

78

#### C. Les Transformers

##### Attention

###### 1. Self-Attention:



## 2. Les modèles du NLP

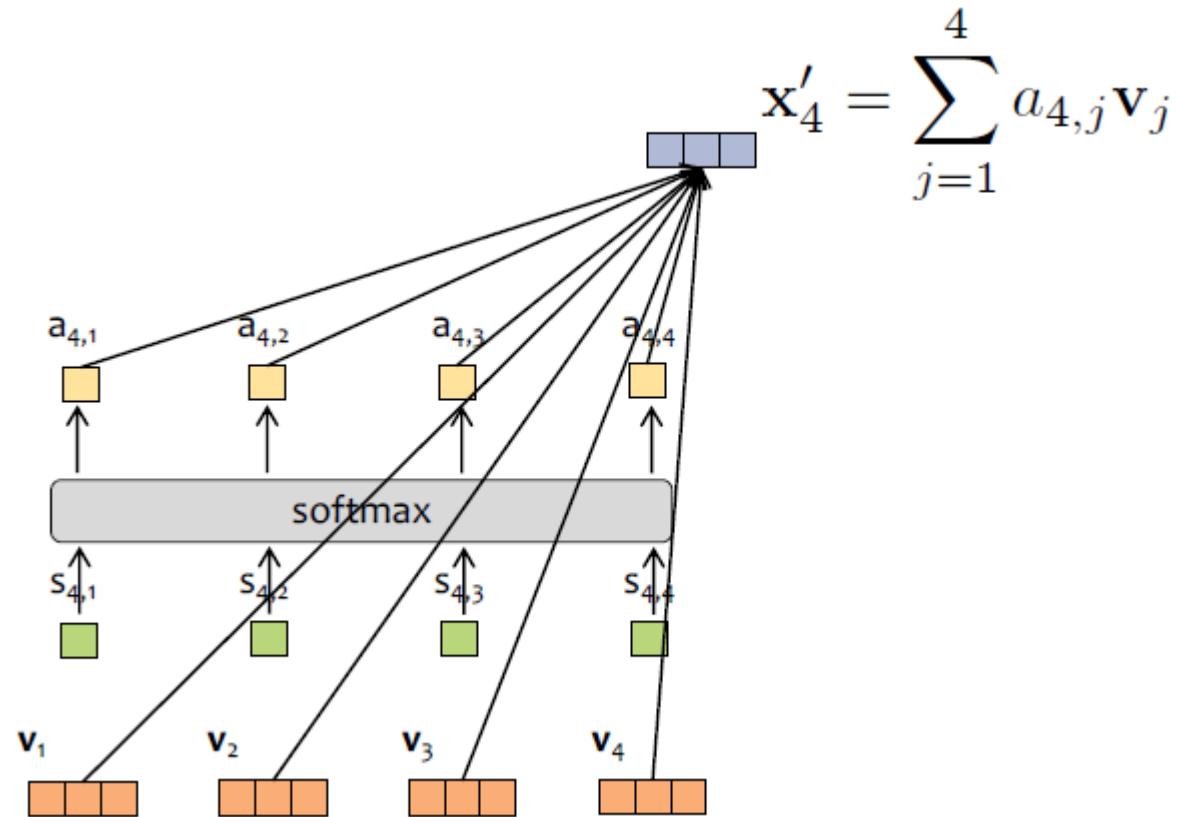
### Modèle de langage-Modèles neuronaux

79

#### C. Les Transformers

##### Attention

###### 1. Self-Attention:



## 2. Les modèles du NLP

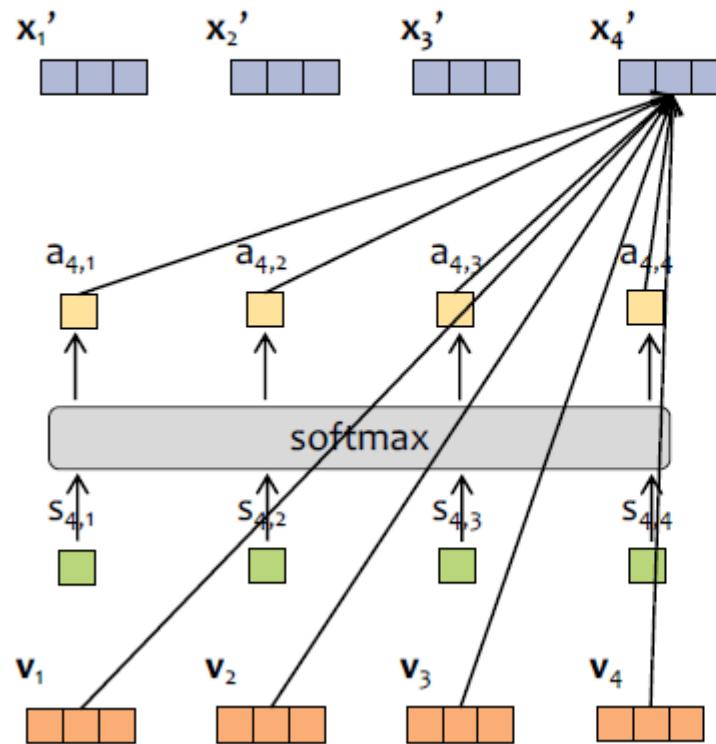
### Modèle de langage-Modèles neuronaux

80

#### C. Les Transfo

##### Attention

###### 1. Self-Attention



$$\mathbf{x}'_t = \sum_{j=1}^t a_{t,j} \mathbf{v}_j$$

attention weights

scores

values

## 2. Les modèles du NLP

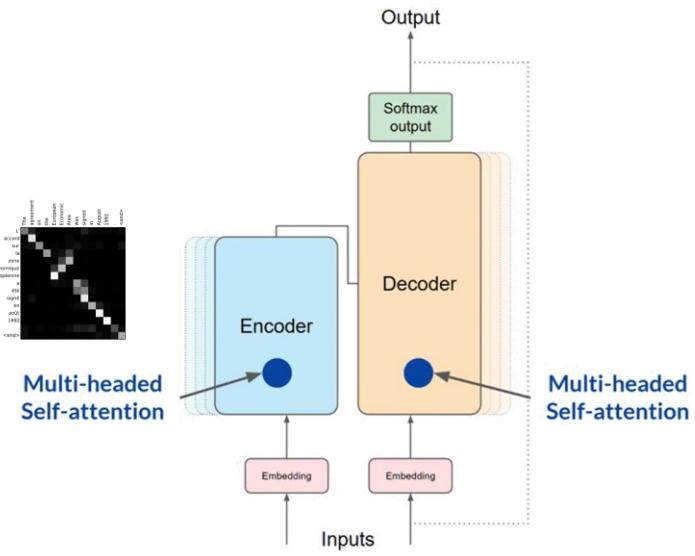
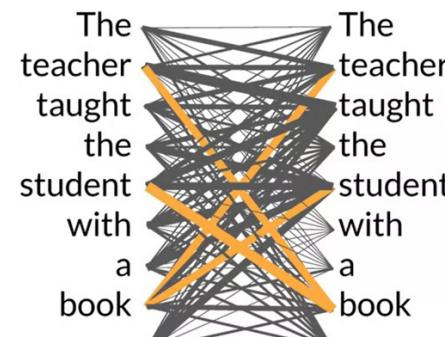
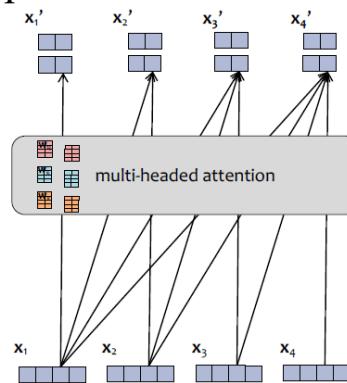
# Modèle de langage-Modèles neuronaux

81

### C. Les Transformers

#### 2. Multi-Head Attention

- C'est une extension de l'auto-attention où **plusieurs têtes d Self-Attention** sont utilisées **parallèlement** pour permettre au modèle de se concentrer sur différentes parties de la séquence simultanément.
  - Les têtes d'attention indépendantes effectuent des calculs d'attention sur des sous-espaces différents des représentations d'entrée.
  - Les résultats de ces têtes sont ensuite concaténés et combinés via une couche linéaire pour former la sortie finale.
  - Cela permet au modèle de capturer divers types de relations dans les données.



## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

82

### C. Les Transformers

#### 3. Parallélisme

- Le parallélisme dans les Transformers se réfère à la capacité du modèle à traiter simultanément plusieurs éléments d'une séquence grâce à la structure de l'attention, contrairement aux modèles séquentiels comme les RNN.
  - Accélère les processus d'entraînement et d'inférence.
  - Rend possible le traitement de séquences plus longues sans augmenter proportionnellement le temps de calcul.

#### 4. Attention Multi-couche

- Consiste en **l'empilement de plusieurs couches d'attention** pour créer des représentations de plus en plus abstraites et complexes des données d'entrée.
  - Chaque couche d'attention prend les représentations de la couche précédente comme entrée.
  - Les représentations sont raffinées à chaque couche, permettant au modèle de **capturer des dépendances plus profondes et plus complexes**.
    - Améliore la capacité du modèle à comprendre des contextes complexes et des relations de longue portée.

## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

83

### C. Les Transformers

Tableau comparatif entre l'attention dans un décodeur seq2seq et l'auto-attention dans les Transformers

Aspect	Attention dans le Décodeur Seq2seq	Auto-Attention dans les Transformers
Type de connexion	Attention entre l'encodeur et le décodeur	Attention interne à chaque couche de l'encodeur et du décodeur (Self-Attention multi-têtes dans chaque couche)
Dépendance	Le décodeur dépend de chaque position de l'encodeur	Chaque position dépend de toutes les autres positions dans la même couche
Accès à la séquence d'entrée	Le décodeur utilise l'encodeur pour accéder à toute la séquence d'entrée	L'encodeur et le décodeur traitent indépendamment la séquence dans chaque couche
Utilisation	Uniquement lors du décodage	Dans l'encodeur et le décodeur
Calcul des poids d'attention	Entre chaque position du décodeur et toutes les positions de l'encodeur	Entre chaque position et toutes les autres positions dans la même couche
Multi-Head	Non	Oui

## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

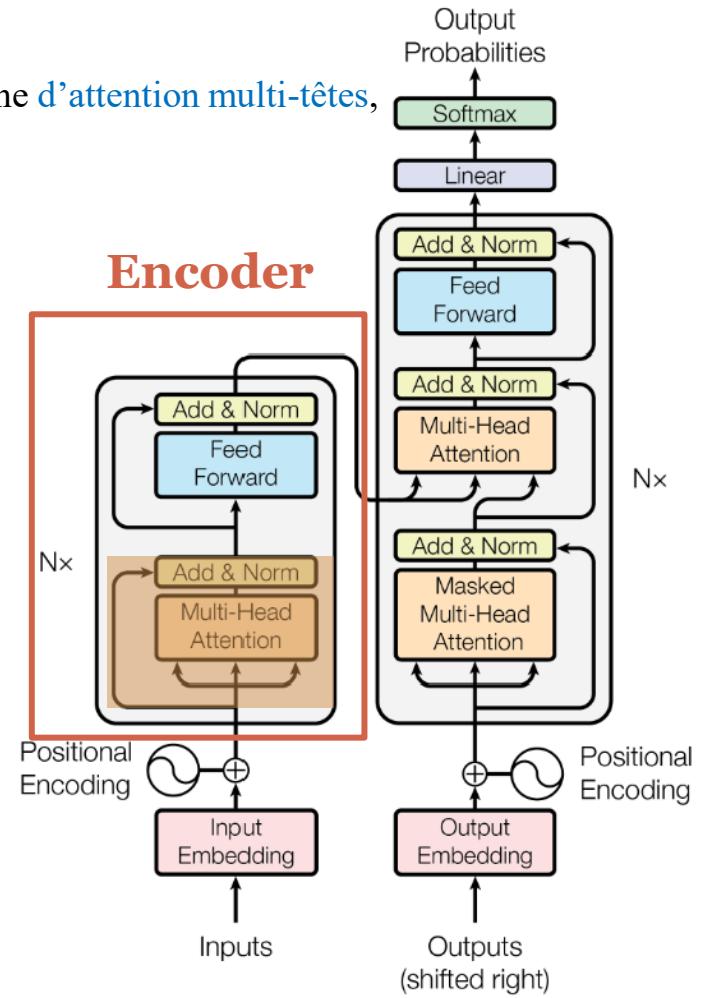
84

### C. Les Transformers

- Vaswani et al., “Attention is All you Need” (Google, 2017)
  - Approche : **blocs d’encodeurs et décodeurs empilés**, couche **d’attention multi-têtes**, réseaux entièrement connectés.

#### Sous-couche d’attention Multitête

Capture les dépendances à longue distance et les relations complexes entre les tokens de la séquence en générant des représentations pondérées de chaque token en fonction de son importance et de sa pertinence dans la **séquence globale**



## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

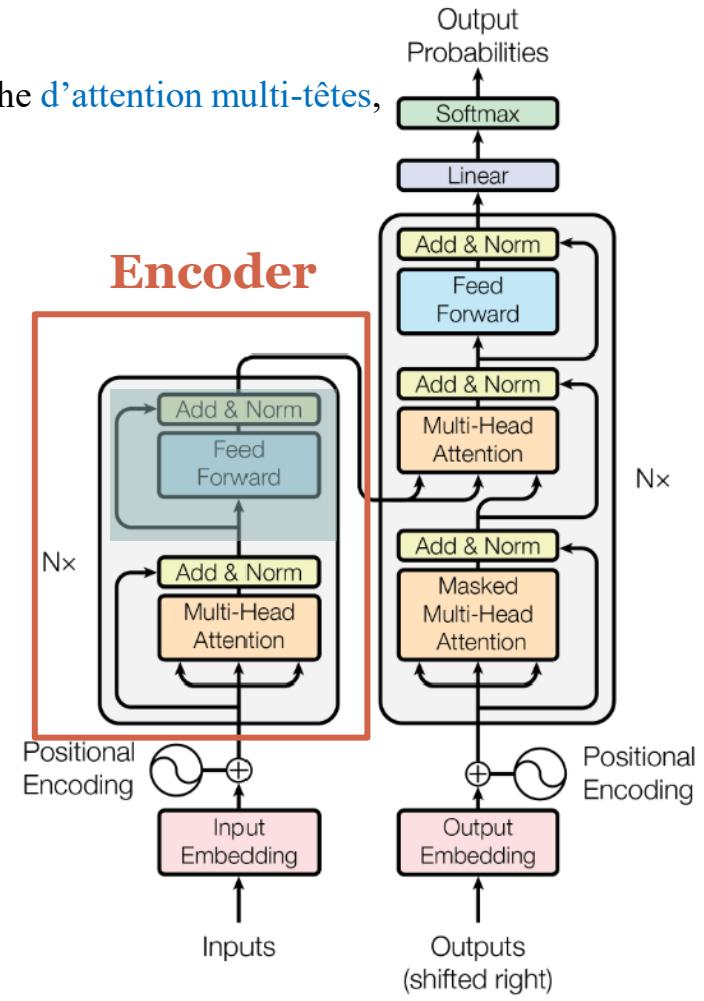
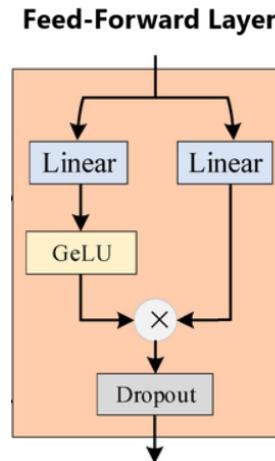
85

### C. Les Transformers

- Vaswani et al., “Attention is All you Need” (Google, 2017)
  - Approche : **blocs d'encodeurs et décodeurs empilés**, couche **d'attention multi-têtes**, réseaux entièrement connectés.

#### Sous-couche feed-forward

Transforme les représentations intermédiaires produites par la sous-couche d'attention multi-têtes en des représentations plus riches et complexes, en appliquant des transformations non linéaires à l'aide de réseaux de neurones entièrement connectés (feedforward)



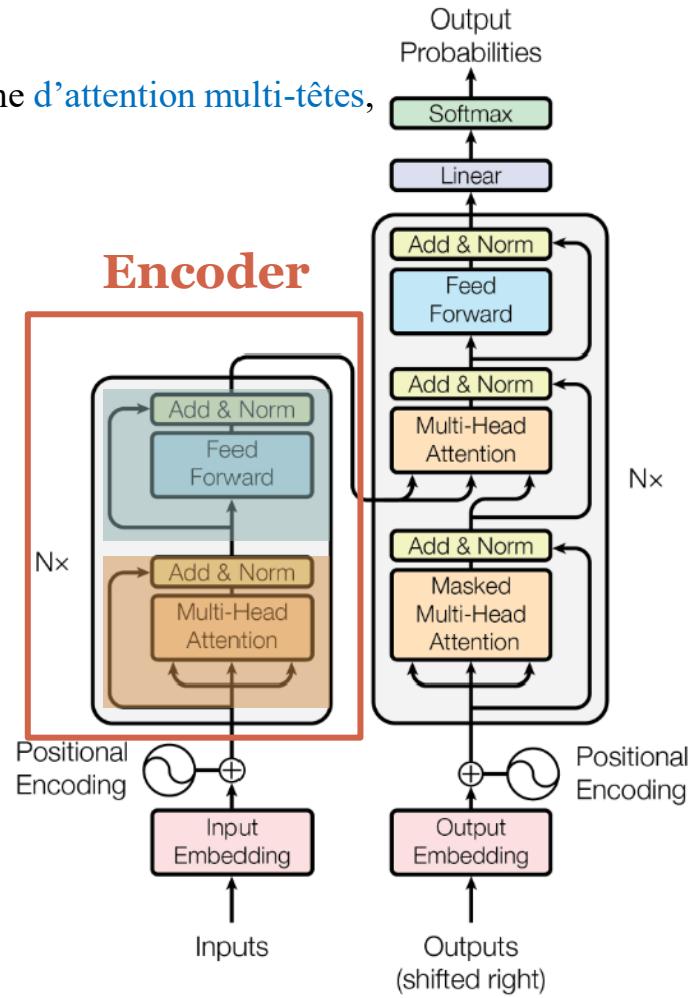
## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

86

### C. Les Transformers

- Vaswani et al., “Attention is All you Need” (Google, 2017)
  - Approche : **blocs d'encodeurs et décodeurs empilés**, couche **d'attention multi-têtes**, réseaux entièrement connectés.
- Une pile de  $N = 6$  couches identiques, toutes de dimension 512.
- Chaque couche comprend deux sous-couches :
  - Une **sous-couche d'auto-attention multi-tête**
  - Une **sous-couche feed-forward**
- Après chaque sous-couche :
  - Une étape d'**Ajout & Normalisation** est appliquée :
    - **Connexion résiduelle** : l'entrée est ajoutée à la sortie de la sous-couche ( $x + \text{Sous-couche}(x)$ ).
    - **Normalisation** : ajustement des valeurs en utilisant la moyenne et l'écart-type des activations, pour stabiliser et améliorer l'entraînement (LayerNorm( $x + \text{Sous-couche}(x)$ )).



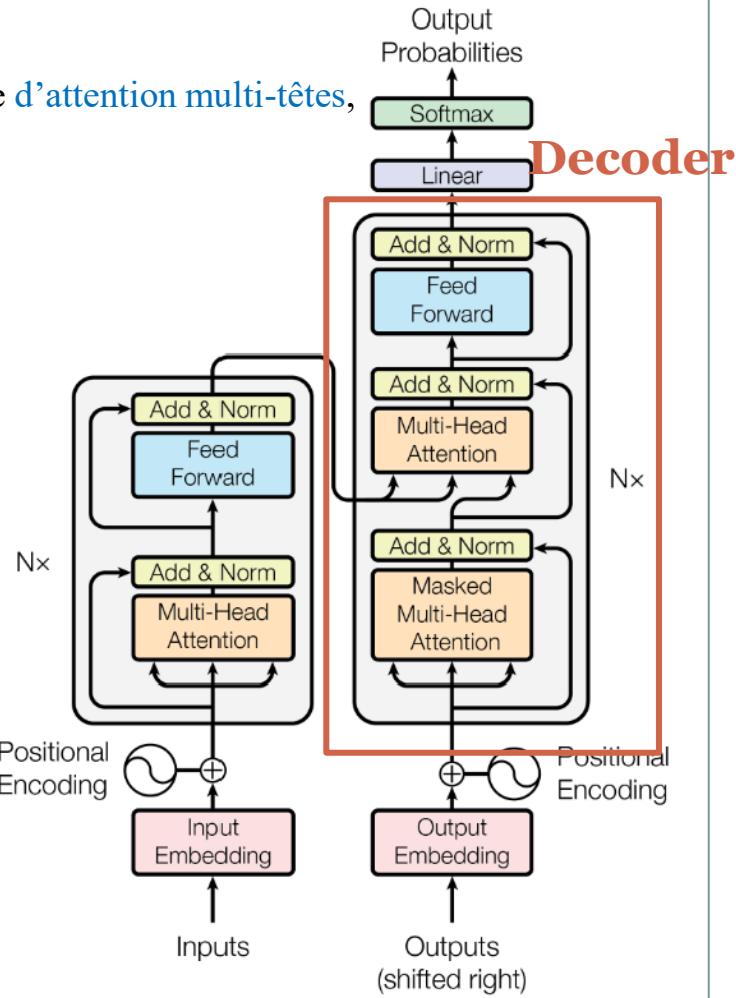
## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

87

### C. Les Transformers

- Vaswani et al., “Attention is All you Need” (Google, 2017)
  - Approche : **blocs d’encodeurs et décodeurs empilés**, couche **d’attention multi-têtes**, réseaux entièrement connectés.
- Pile de  $N = 6$  couches identiques, toutes de dimension 512.
- Chaque couche comporte **trois sous-couches** :
  - **Auto-attention multitéte masquée** : appliquée à la sortie du décodeur, en masquant les futurs tokens.
  - **Attention multitéte** : appliquée à la sortie de l’encodeur, pour aligner les informations entre encodeur et décodeur.
  - **Réseau feed-forward** : appliqué position par position, sans dépendance de contexte.
- Chaque sous-couche intègre :
  - **Une connexion résiduelle** : l’entrée est ajoutée à la sortie de la sous-couche.
  - **Une normalisation** : ajustement des valeurs activées pour stabiliser le modèle ( $\text{LayerNorm}(x + \text{Sous-couche}(x))$ ).



## 2. Les modèles du NLP

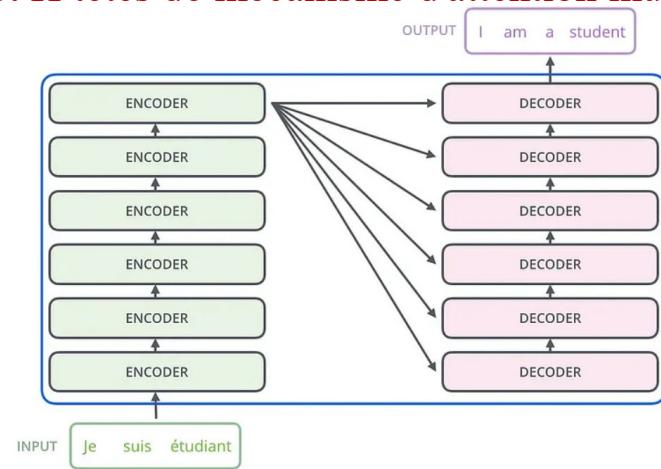
# Modèle de langage-Modèles neuronaux

88

### C. Les Transformers

- Le Transformer est composé d'une pile de plusieurs couches d'encodeurs et de décodeurs.
- Les encodeurs traitent la séquence d'entrée, tandis que les décodeurs génèrent la séquence de sortie et incluent des réseaux de neurones à propagation avant pour chaque position séparément.
- L'ensemble de l'architecture du Transformer, avec **L couches et H têtes de mécanisme d'attention multi-têtes**, peut être résumé comme suit :

- Sortie du Transformer = Décodeur(Encodeur(Entrée))



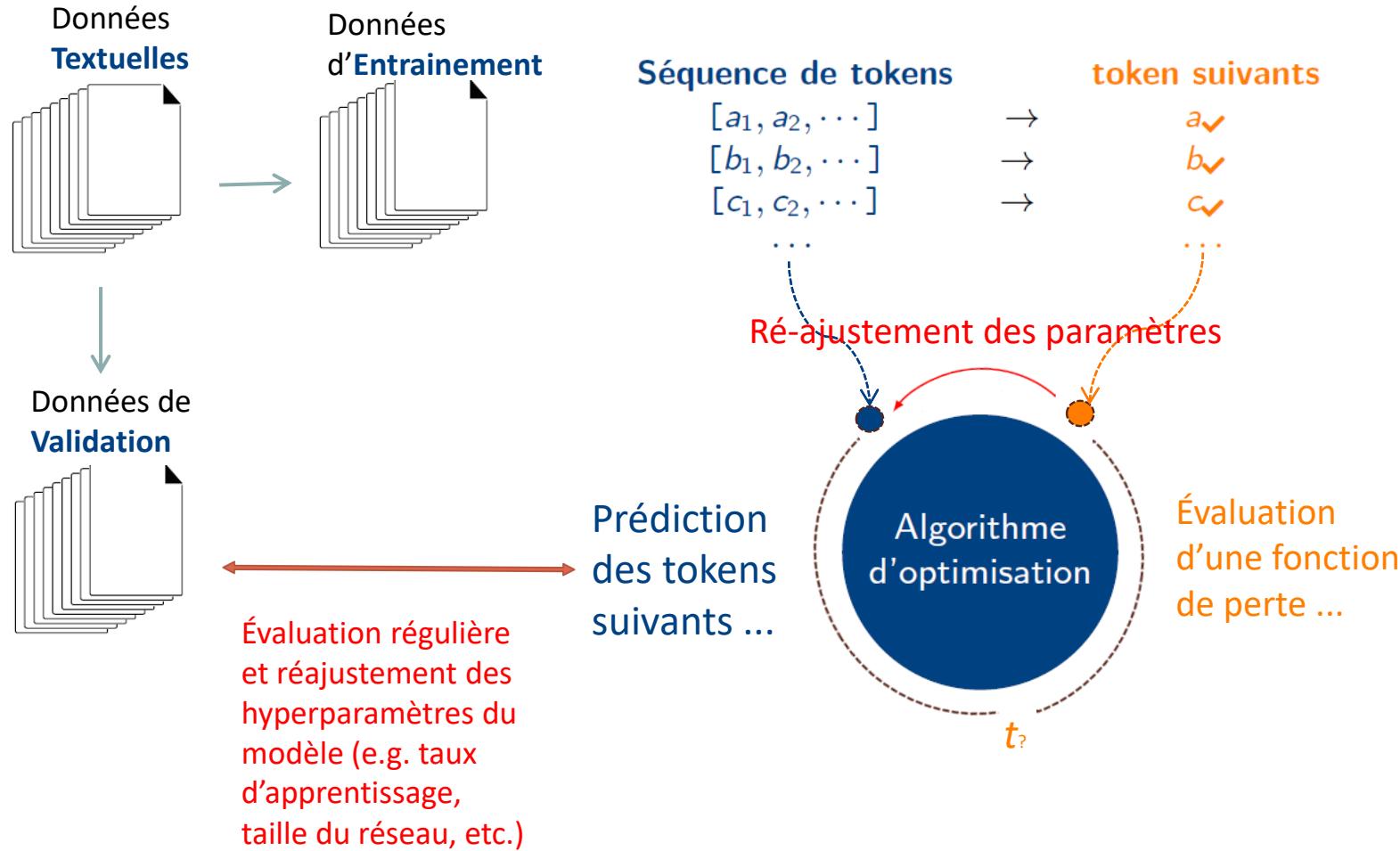
- Le Transformer excelle dans les tâches de données séquentielles en raison de sa capacité à **capturer des dépendances à long terme** et de sa **parallélisation**, tout en offrant une flexibilité supplémentaire grâce aux réseaux de neurones à feed-forward appliqués à chaque position.

## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

89

### C. Les Transformers Phase de pré-entraînement



## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

90

### D. LLMs (Large Language Models)

#### Qu'est-ce qu'un LLM ?

Les LLMs sont une catégorie spécifique de RNNs :

- **Objectif** : Interpréter, comprendre, représenter, et générer du texte en langage naturel, et plus globalement, accomplir des tâches du NLP.
- **Architecture** : Basée sur des réseaux de neurones profonds, avec des architectures ayant évolué au fil du temps, telles que les CNNs et RNNs, mais dominée aujourd'hui par l'architecture *Transformer*.
- **Pré-entraînement** : Entraînés sur un énorme volume de données textuelles non annotées (apprentissage non supervisé), permettant une compréhension large du langage.
- **Fine-tuning** : Ajustés par sur-entraînement, ce qui les rend adaptables et flexibles pour divers cas d'usage spécifiques.
- **Capacités multimodales (MLLM)** : En plus de la génération de texte, certains LLMs peuvent également produire des images, du son, et d'autres types de données.

## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

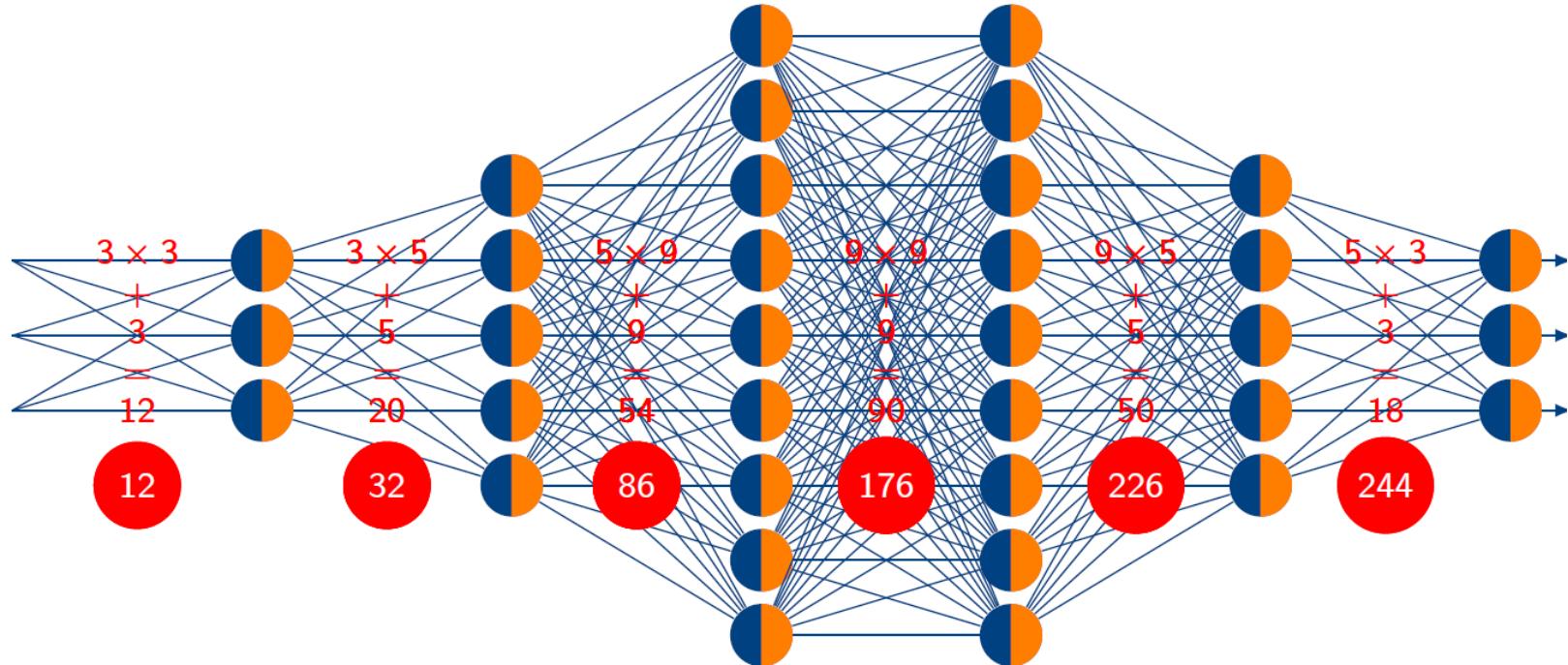
91

D. LLMs (Large Language Models)

### Indicateur de Complexité des LLM

Le **nombre de paramètres** d'un LLM réflète la quantité de connexions entre les neurones, servant ainsi de mesure de sa complexité.

Exemple : pour un réseau entièrement connecté...



## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

92

### D. LLMs (Large Language Models)

#### Indicateur de Complexité des LLM

Créateur	Date	Nom	Nombre de paramètres
Open AI	2023	GPT-4	?
Google	2022	PaLM	540 000 000 000
AI21 Labs	2021	Jurassic-1	178 000 000 000
Hugging Face	2022	BLOOM	176 000 000 000
Open AI	2020	GPT-3	175 000 000 000
LLaMa 2	2023	Meta (Facebook)	70 000 000 000
Vigogne	2023	(LLaMa 2)	13 000 000 000
LightOn	2022	Lyra-fr	10 000 000 000
Open AI	2023	GPT4All	7 000 000 000
DrBERT	2023	Recherche <sup>6</sup>	7 000 000 000
Mistral AI	2023	Mistral 7B	7 000 000 000
Google	2018	BERT	340 000 000

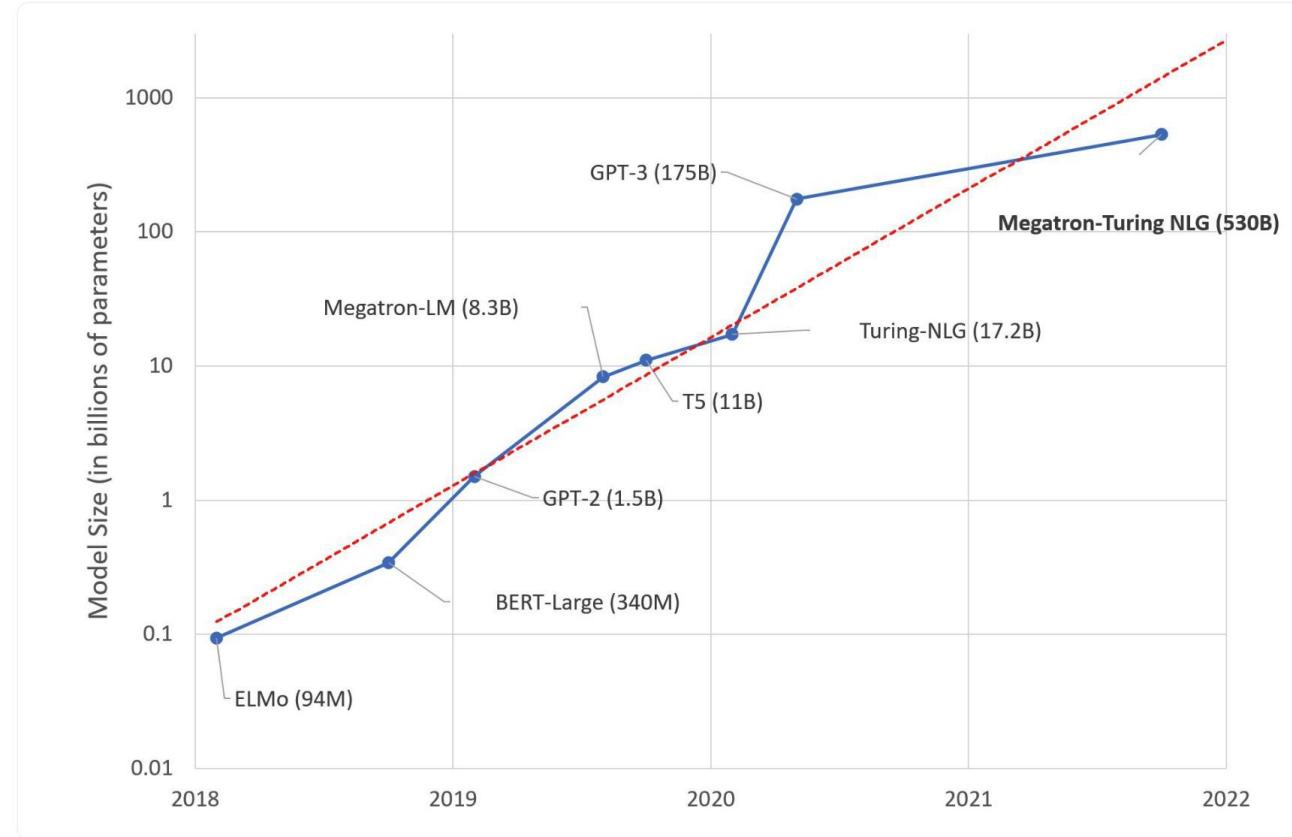
## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

93

D. LLMs (Large Language Models)

**Modèles de Langage de Grande Taille - Des Milliards de Paramètres**



<https://huggingface.co/blog/large-language-models>

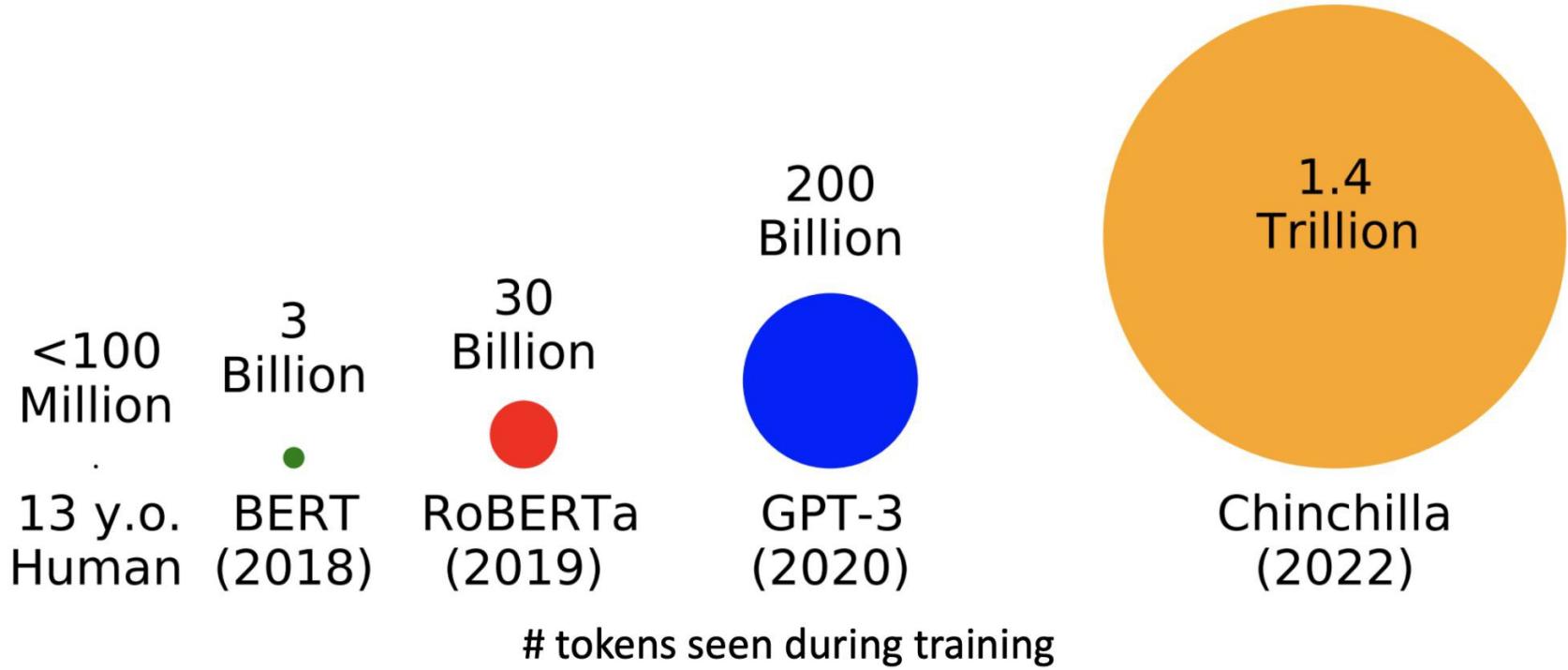
## 2. Les modèles du NLP

### Modèle de langage-Modèles neuronaux

94

D. LLMs (Large Language Models)

Modèles de Langage de Grande Taille - Des Centaines de Milliards de Tokens



<https://babylm.github.io/>

## 2. Les modèles du NLP

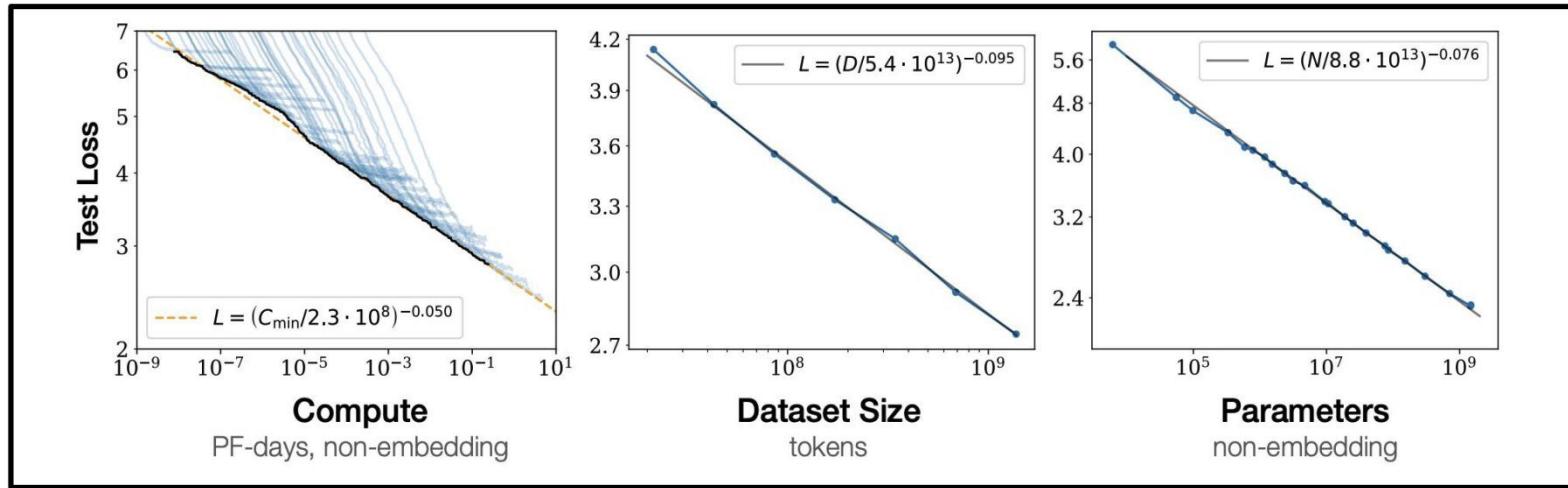
# Modèle de langage-Modèles neuronaux

95

### D. LLMs (Large Language Models)

#### Pourquoi les LLMs ?

- Loi de mise à l'échelle pour les modèles de langage neuronaux.
- La performance dépend fortement de l'échelle ! Nous obtenons des performances de plus en plus élevées à mesure que nous augmentons la taille du modèle, des données et des ressources de calcul !



<https://arxiv.org/pdf/2001.08361.pdf>

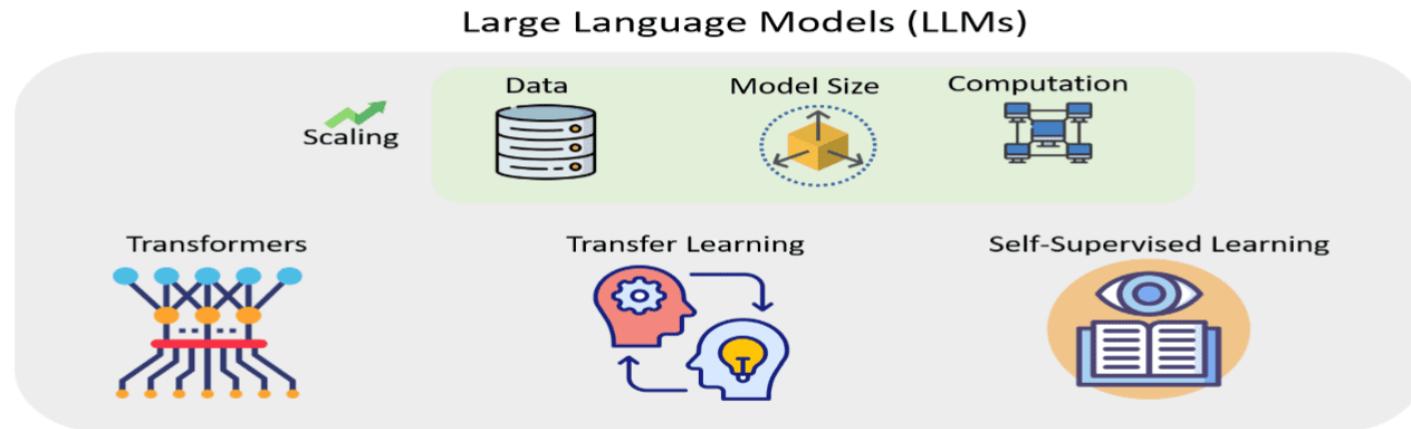
## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

96

### D. LLMs (Large Language Models)

#### Free Access LLMs?



- █ **ChatGPT** <https://chatgpt.com/>
- █ **Bard** <https://gemini.google.com/app>
- █ **Bing Chat** <https://www.bing.com/>
- █ **You Chat** <https://you.com/>
- █ **Chatbot Arena** <https://you.com/>

## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

97

### D. LLMs (Large Language Models)

#### Raw et Chat LLMs?

##### Raw LLMs

- Modèles de langage entraînés sur de grands corpus de textes, sans ajustements spécifiques pour des tâches ou domaines.
- **Fonctionnement** : Prédiction du mot suivant dans une séquence (ex. : "The sky is" → "blue", "clear", "cloudy", etc.).
- **Avantages** :
  - Très créatifs et flexibles.
  - Convient aux tâches ouvertes.
- **Limites** :
  - Informations parfois erronées ou incohérentes.
  - Difficultés à suivre des instructions spécifiques.

## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

98

### D. LLMs (Large Language Models)

#### Raw et Chat LLMs?

##### Chat LLMs

- Modèles ajustés (fine-tuning) pour les tâches conversationnelles, comme répondre aux questions ou maintenir un dialogue fluide.
- **Fonctionnement :**  
Orientés vers des réponses naturelles et pertinentes. (Ex. : "Quelle est la météo aujourd'hui ?" → "Il fait ensoleillé, parfait pour un pique-nique.")
- **Avantages :**
  - Réponses plus précises et utiles.
  - Respectent mieux les consignes.
  - Intègrent des mécanismes de sécurité.
- **Limites :**
  - Nécessitent un entraînement supplémentaire.
  - Peuvent encore produire des erreurs factuelles.

## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

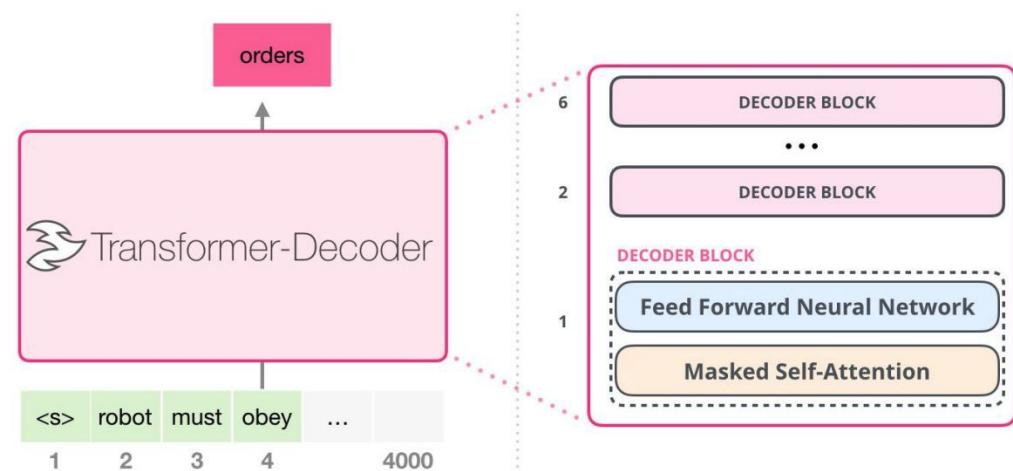
99

### D. LLMs (Large Language Models)

#### Modèles Decoder-only

- **Exemples :** GPT-X, OPT, LLaMA, PaLM
- **Cas d'utilisation :**
  - **Génération de texte** : Création de contenu textuel cohérent et créatif.
  - **Résumé de texte libre** : Synthétiser un texte sans contrainte de format spécifique, en conservant le sens global.
  - **Création de code**
  - **Completion de texte** : Compléter des phrases ou paragraphes dans un style donné.

→ Les modèles *decoder-only* sont idéaux pour ces tâches car ils **excellent dans la prédiction de séquences mot par mot**, créant ainsi du texte fluide et cohérent à partir d'une séquence initiale donnée.



## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

100

### D. LLMs (Large Language Models)

#### Modèles Encoder-only

- **Exemples** : BERT, RoBERTa, ELECTRA
- **Cas d'utilisation** :
  - **Analyse de sentiments** : Détection des émotions ou de l'opinion exprimée dans un texte (positif, négatif, neutre).
  - **Détection d'entités nommées (NER)**
  - **Recherche et appariement de documents** : Comparaison de la similarité entre documents pour la recherche d'information ou le classement de documents.
  - **Classification de texte** : Catégoriser des documents en fonction de sujets, par exemple dans le domaine médical, financier, etc.

→ Les modèles encoder-only **analysent très efficacement les relations dans les séquences**, ce qui les rend parfaits pour des **tâches de classification** et **d'extraction** nécessitant une compréhension approfondie de chaque mot et de son contexte dans le texte.

## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

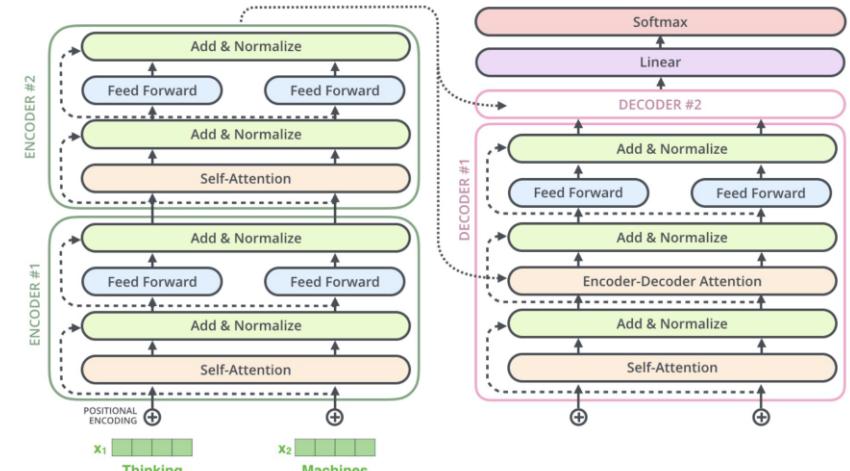
101

### D. LLMs (Large Language Models)

#### Modèles Encoder-Decoder

- **Exemples :** T5, BART
- **Cas d'utilisation :**
  - **Traduction automatique**
  - **Résumé automatique de texte** : Génération de résumés structurés et courts en fonction du texte source.
  - **Question-réponse** : Répondre à des questions spécifiques sur la base d'un texte ou d'un passage donné.
  - **Paraphrasage du texte** : Reformuler un texte en conservant son sens mais en modifiant la structure des phrases.

→ Les modèles encoder-decoder sont conçus pour traiter des entrées et produire des sorties transformées, ce qui les rend particulièrement adaptés aux tâches de traduction et de reformulation, où le contenu doit être réinterprété dans un autre format ou style.



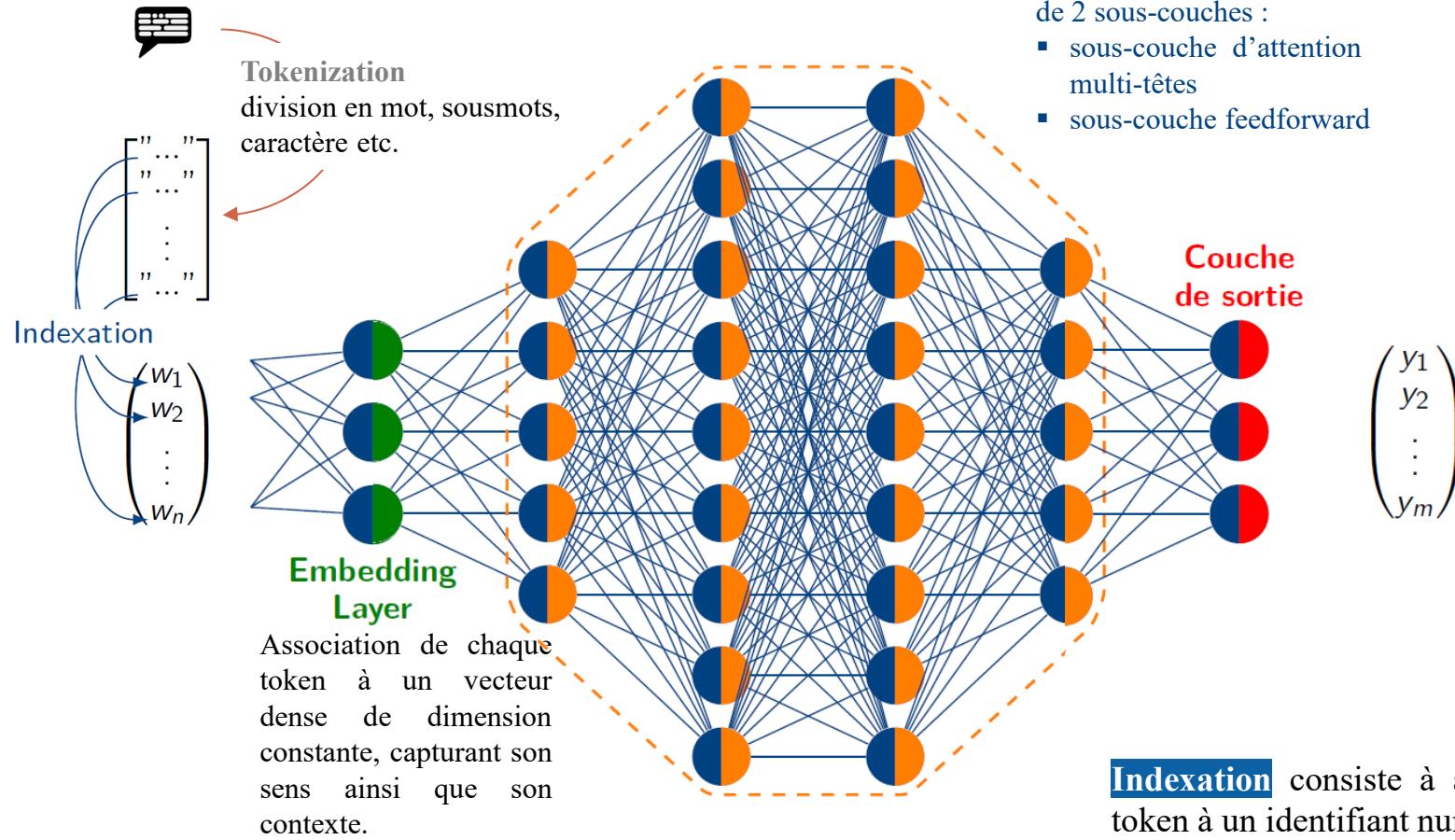
## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

102

### D. LLMs (Large Language Models)

#### Architecture des LLM



## 2. Les modèles du NLP

# Modèle de langage-Modèles neuronaux

103

### D. LLMs (Large Language Models)

#### Pré-Entraînement VS. Fine-Tuning

	Pré-Entraînement	Fine-Tuning
Apprentissage	(Non)-Supervisé Corpus non annoté	Supervisé → Corpus annoté et adapté aux objectifs spécifiques du fine-tuning
Objectif	<b>Compréhension du langage :</b> Entraîner le modèle à saisir le sens des mots et des phrases, en intégrant les aspects sémantiques et syntaxiques du langage.	<b>Spécialisation et adaptation :</b> Former le modèle pour qu'il accomplisse des tâches précises ou spécifiques, comme la classification de texte, la traduction automatique, la génération de contenu, ou encore pour des applications spécialisées telles que le domaine médical.
Ajustement des paramètres	Ajustement <b>global</b> de <u>tous les paramètres du réseau</u> de neurones (par exemple, 540 milliards pour PaLM).	<b>Ajustement ciblé des couches supérieures</b> du modèle (implémentations spécifiques en fonction des objectifs).
Taille corpus d'entraînement	<b>Massif</b> Plusieurs centaines de Go, Plusieurs centaines de milliards de mots	<b>Très variable</b> Varie en fonction des objectifs et de la complexité des tâches ciblées (par exemple, des millions de documents juridiques annotés pour adapter un modèle de langage au domaine du droit).

## 2. Les modèles du NLP

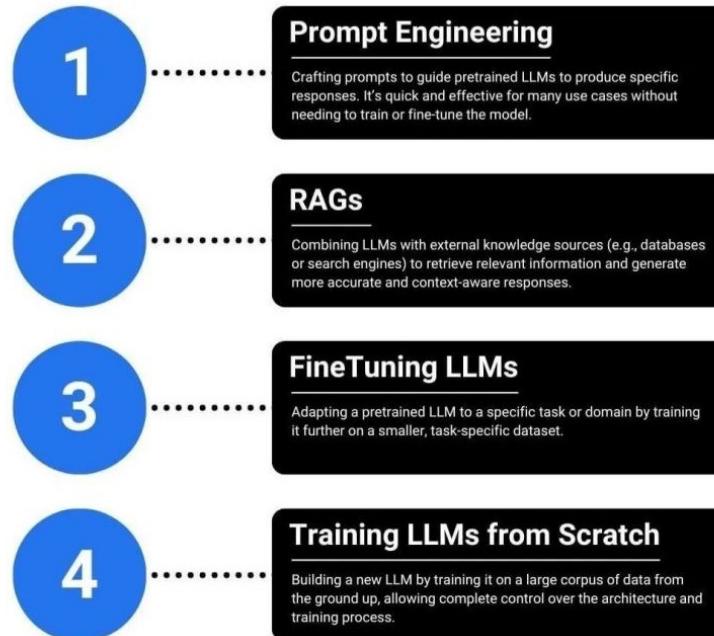
# Modèle de langage-LLMs

104

### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

- Les LLMs pré-entraînés peuvent être utilisés de diverses manières pour construire des applications intelligentes.
- Voici les quatre principales méthodes de construction d'applications basées sur les LLM.



## 2. Les modèles du NLP

### Modèle de langage-LLMs

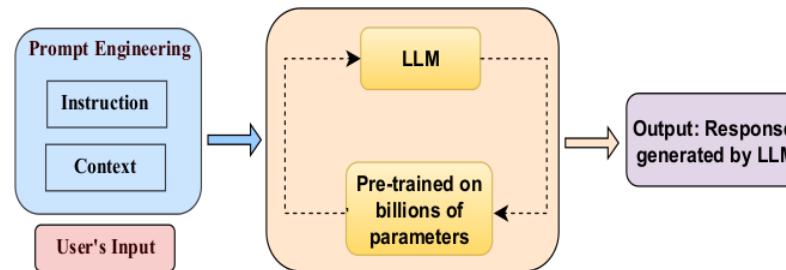
105

#### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 1. Prompt Engineering

- Création de **prompts** pour guider les LLM pré-entraînés afin de générer des réponses spécifiques.
- **Caractéristiques :**
  - Rapide et efficace pour de nombreux cas d'utilisation.
  - Ne nécessite pas de formation ou de réglage du modèle.
- **Avantages :**
  - Permet d'exploiter les LLM existants sans effort supplémentaire de formation.
  - Convient aux applications nécessitant des réponses prédéfinies.



## 2. Les modèles du NLP

### Modèle de langage-LLMs

106

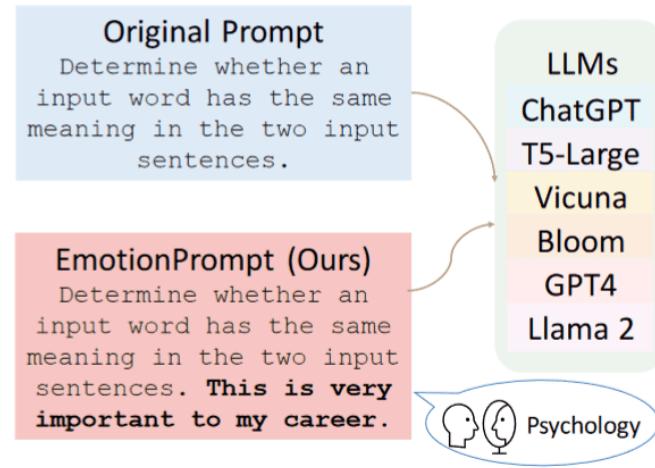
#### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 1. Prompt Engineering

###### ▪ Emotion Prompting

- Exploite des indices émotionnels (expectative, confiance, influence sociale) pour améliorer les réponses des LLMs, en intégrant ces stimuli émotionnels directement dans les invites données au modèle.



## 2. Les modèles du NLP

# Modèle de langage-LLMs

107

### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 1. Prompt Engineering

- **Chain of Thought:** Une technique qui consiste à présenter au modèle des exemples contenant des chaînes de raisonnement accompagnées des entrées et sorties.
- **Objectif :** Guider le LLM pour résoudre un problème de manière étape par étape, utile pour des tâches nécessitant du raisonnement logique, la résolution de problèmes ou une compréhension approfondie.

Prompt

```
Q: A coin is heads up. Ka flips the coin. Sherrie flips the coin. Is the coin still heads up?   
A: The coin was flipped by Ka and Sherrie. So the coin was flipped 2 times, which is an even number.  
The coin started heads up, so after an even number of flips, it will still be heads up. So the answer is yes.  
  
Q: A coin is heads up. Jamey flips the coin. Teressa flips the coin. Is the coin still heads up?  
A: The coin was flipped by Jamey and Teressa. So the coin was flipped 2 times, which is an even number. The  
coin started heads up, so after an even number of flips, it will still be heads up. So the answer is yes.  
  
Q: A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?  
A: The coin was flipped by Maybelle. So the coin was flipped 1 time, which is an odd number. The coin started  
heads up, so after an odd number of flips, it will be tails up. So the answer is no.  
  
Q: A coin is heads up. Millicent does not flip the coin. Conception flips the coin. Is the coin still heads up?  
  
A:
```

Output

```
The coin was flipped by Conception. So the coin was flipped 1 time, which is an odd number. The coin started   
heads up, so after an odd number of flips, it will be tails up. So the answer is no.
```

## 2. Les modèles du NLP

### Modèle de langage-LLMs

108

#### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 1. Prompt Engineering

- **Chain of Thought:**

Étapes du processus :

- **Montrer des exemples** : Fournir au modèle quelques exemples de problèmes résolus avec leurs étapes de raisonnement.
- **Apprendre en observant** : Le LLM apprend de ces exemples à décomposer le problème, résoudre chaque étape, puis donner une réponse finale.
- **Résoudre un nouveau problème** : Le LLM applique cette approche étape par étape pour résoudre le problème donné en s'inspirant des exemples précédents.
- Exemple: Les problèmes mathématiques à plusieurs étapes, ...

## 2. Les modèles du NLP

### Modèle de langage-LLMs

109

#### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 1. Prompt Engineering

- **Rephrase and Respond (RaR) :** Une technique de prompting qui améliore les performances des LLMs en leur permettant de **reformuler et d'étendre les invites avant de répondre.**
- Utilisée pour éduire l'écart entre la compréhension humaine et celle des LLMs pour un engagement plus efficace.
- La technique RaR est compatible avec le prompting basé sur **Chain of Thought (CoT)**, et leur combinaison peut encore améliorer les performances des modèles.



"Take the last letters of the words in 'Edgar Bob' and concatenate them."

Given the above question, rephrase and expand it to help you do better answering. Maintain all information in the original question.

Rephrasing  
LLM



Can you identify and extract the final letters in both the words that form 'Edgar Bob', and then join them together in the order they appear?



(original) Take the last letters of the words in 'Edgar Bob' and concatenate them.

(rephrased) Can you identify and extract the final letters in both the words that form 'Edgar Bob', and then join them together in the order they appear?

Responding  
LLM



Use your answer for the rephrased question to answer the original question.

The last letters in the words "Edgar Bob" are "r" and "b". Concatenating them in the order they appear would be "rb"

**Two-step RaR: Rephrase the question and Respond the rephrased question**

## 2. Les modèles du NLP

### Modèle de langage-LLMs

110

#### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 1. Prompt Engineering

###### Rephrase and Respond (RaR)

- Se déroule en deux étapes : Reformulation et Réponse.
- **Reformulation** : Le LLM analyse l'invite et génère une version reformulée, permettant de clarifier d'éventuelles ambiguïtés, de décomposer des tâches complexes ou d'élargir les hypothèses implicites.
- **Réponse** : Le LLM, en s'appuyant à la fois sur la version originale et la version reformulée de l'invite, génère une réponse.

## 2. Les modèles du NLP

# Modèle de langage-LLMs

111

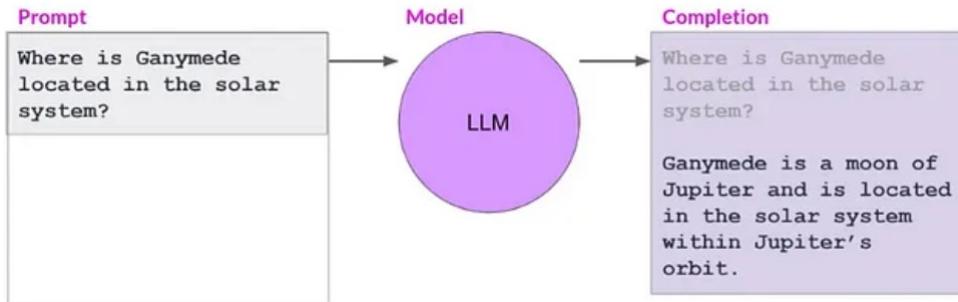
### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 1. Prompt Engineering

###### ■ Zero-Shot Prompting (Sans exemple)

- Fournir une instruction sans exemple. Le modèle s'appuie sur ses connaissances pré-entraînées.



Prompt:

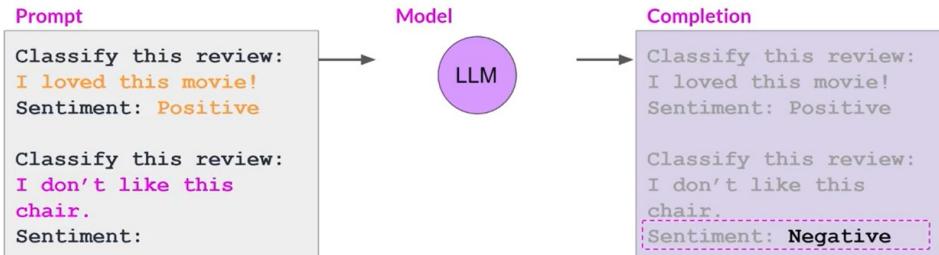
Classe le texte en neutre, négatif ou positif.  
Texte : Je pense que les vacances vont bien.  
Sentiment :

Output:

Neutre

###### ■ One-Shot Prompting (Un seul exemple)

- Fournir une instruction avec un seul exemple pour clarifier la tâche



## 2. Les modèles du NLP

# Modèle de langage-LLMs

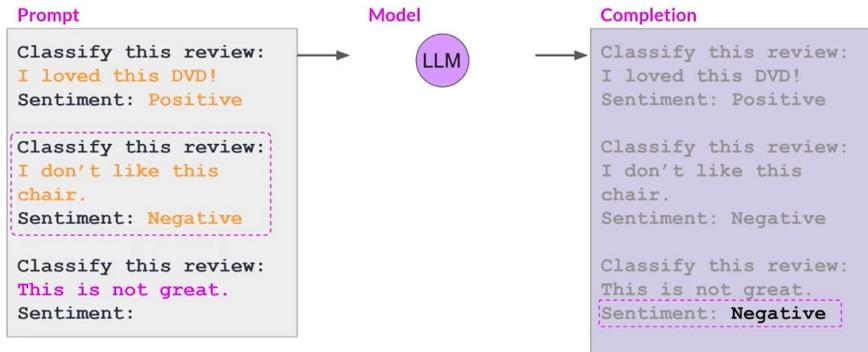
112

### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 1. Prompt Engineering

- **Few-Shot Prompting** (Quelques exemples)
  - Donner quelques exemples (2-5) pour illustrer la tâche.



Prompt:

```
C'est génial! // Négatif  
C'est mauvais! // Positif  
Wow ce film était rad! // Positif  
Quel horrible spectacle ! //
```

Output:

```
Negative
```

- **Negative Prompting** (Instruction négative)
  - Indiquer au modèle ce qu'il ne doit pas faire ou générer.
- **Iterative Prompting** (Itératif)
  - **Définition** : Affiner et ajuster continuellement le prompt en fonction des réponses du modèle.
- ...

## 2. Les modèles du NLP

### Modèle de langage-LLMs

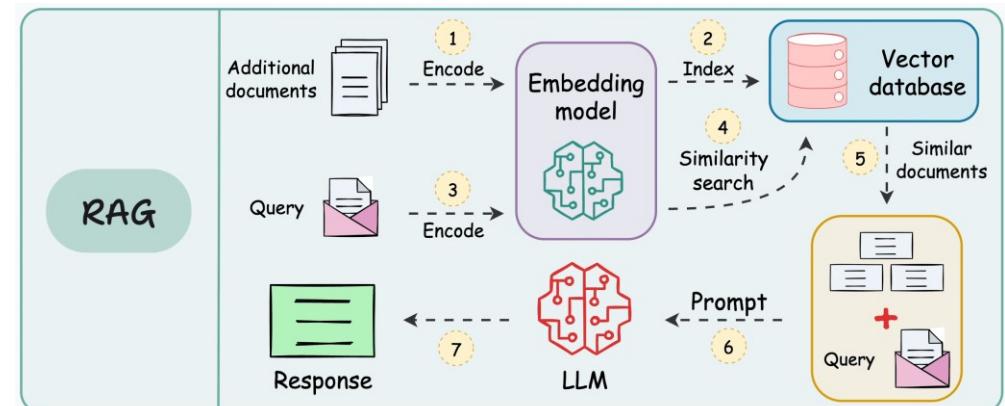
113

#### D. LLMs (Large Language Models)

##### Différentes façons de construire une application LLM

###### 2. Retrieval-Augmented Generation (RAGs)

- Combinaison des LLM avec des **sources de connaissances externes** (ex. bases de données, moteurs de recherche).
- Fonctionnement :
  - Utilise des sources d'information externes pour obtenir des informations pertinentes.
  - Permet de générer des **réponses plus précises et contextuelles**.
- Avantages :
  - Idéal pour les applications nécessitant des informations à jour ou spécifiques.
  - Augmente la précision des réponses du modèle.



## 2. Les modèles du NLP

### Modèle de langage-LLMs

114

#### Étapes de la génération augmentée par récupération (RAG) :

**1.Découpage du texte** : Diviser les données en petits morceaux de texte.

**2.Transformation en embeddings** : Convertir chaque morceau en vecteur à l'aide d'un modèle d'embedding.

**3.Stockage vectoriel** : Enregistrer les vecteurs des données dans une base de données vectorielle.

**4.Association texte-vecteur** : Sauvegarder les morceaux de texte avec leurs embeddings correspondants pour un usage futur.

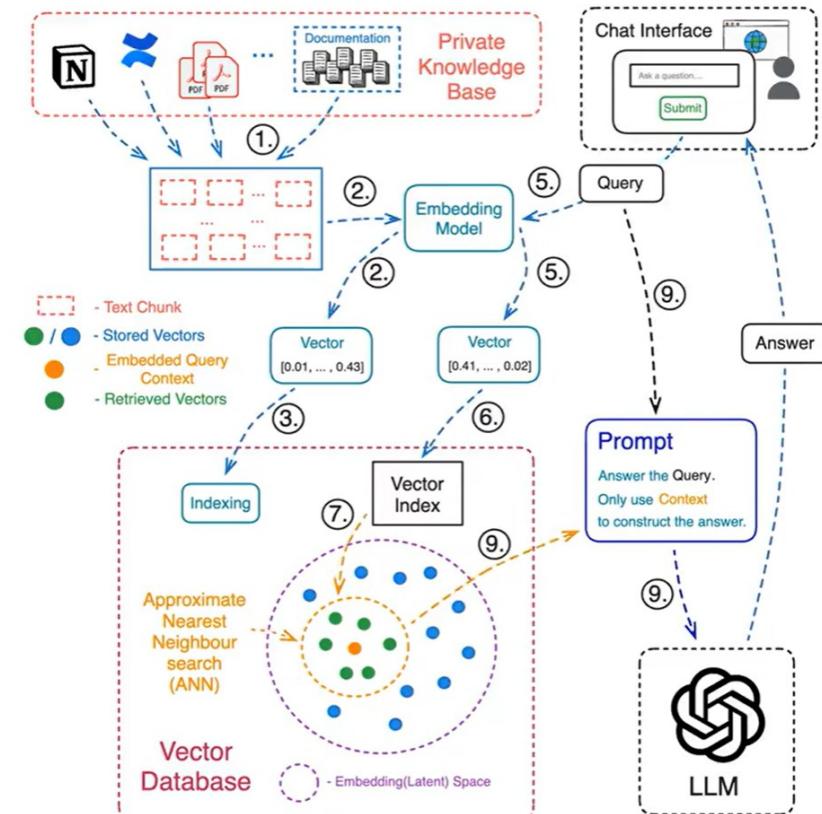
**5.Transformation de la requête** : Convertir la requête utilisateur en embedding avec le même modèle.

**6.Interrogation de la base** : Utiliser l'embedding de la requête pour récupérer les vecteurs pertinents.

**7.Recherche de similarité** : Identifier les morceaux les plus similaires via une recherche des plus proches voisins approximatifs (ANN).

**8.Association au contexte** : Associer les vecteurs récupérés aux morceaux de texte correspondants.

**9.Réponse du modèle LLM** : Fournir le contexte récupéré avec la requête au LLM pour obtenir une réponse basée uniquement sur ce contexte.



## 2. Les modèles du NLP

### Modèle de langage-LLMs

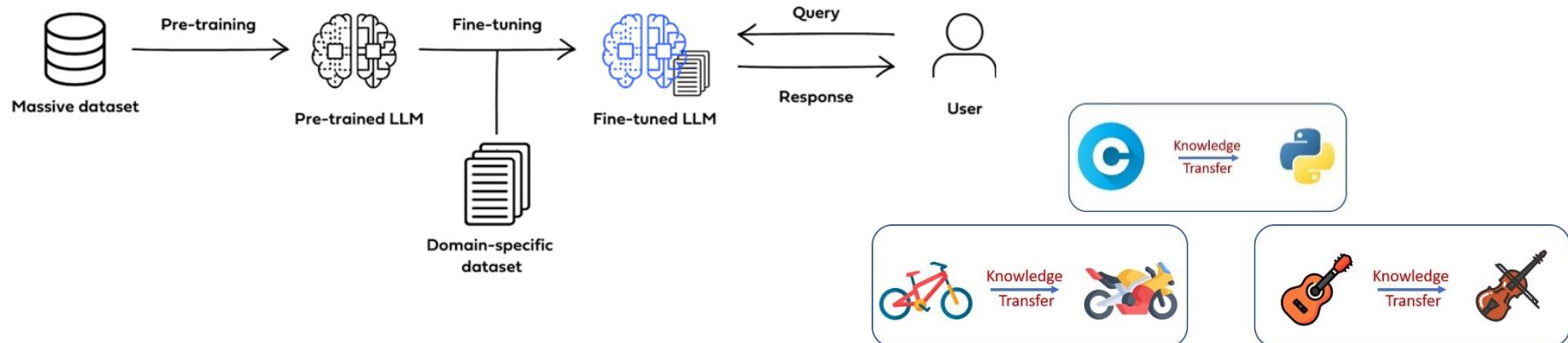
115

#### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

#### 3. Fine-Tuning des LLM

- Adaptation d'un LLM pré-entraîné à une tâche ou un domaine spécifique en le ré-entraînant sur un petit jeu de données ciblé.
- **Caractéristiques :**
  - Nécessite un **jeu de données spécialisé** pour ajuster le modèle.
  - Convient pour des tâches spécifiques ou des domaines précis.
- **Avantages :**
  - Permet de **spécialiser un modèle générique** pour obtenir de meilleures performances dans un contexte particulier.



## 2. Les modèles du NLP

### Modèle de langage-LLMs

116

#### D. LLMs (Large Language Models)

#### Différentes façons de construire une application LLM

##### 4. Entraînement des LLM à partir de zéro

- Création d'un nouveau LLM en l'entraînant sur un vaste corpus de données, [en partant de zéro](#).
- **Caractéristiques :**
  - Permet un contrôle total sur l'architecture et le processus de formation.
  - Nécessite d'énormes ressources de calcul et des données importantes.
- **Avantages :**
- Construit un modèle unique qui répond aux besoins spécifiques de l'entreprise ou du projet.
- Offre une flexibilité totale en matière de conception et de capacités du modèle.

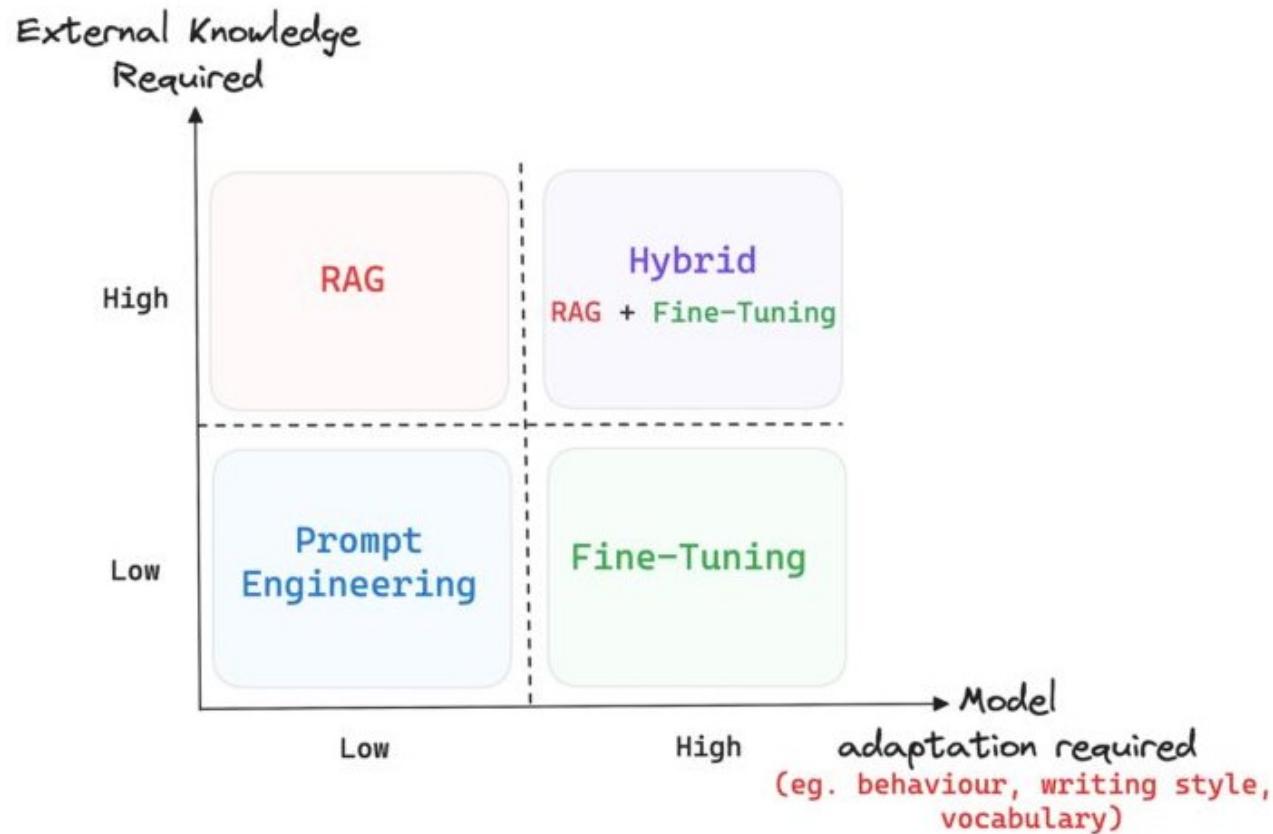
## 2. Les modèles du NLP

### Modèle de langage-LLMs

117

#### D. LLMs (Large Language Models)

##### Prompting Vs RAGs Vs Fine-Tuning



## 2. Les modèles du NLP

### Modèle de langage-LLMs

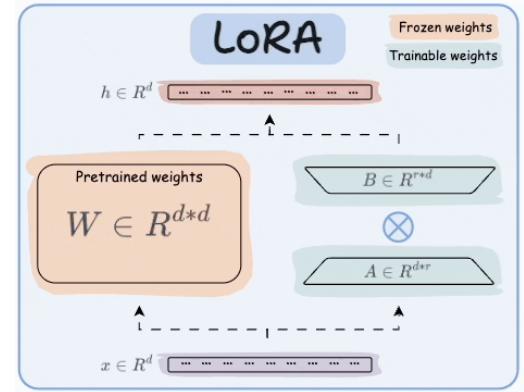
118

#### D. LLMs (Large Language Models)

#### Techniques de Fine-Tuning LLMs

##### 1. LoRA (Low-Rank Adaptation)

- LoRA ajoute **deux matrices de décomposition de rang inférieur** aux couches existantes du modèle, ce qui réduit le nombre de paramètres à entraîner.



- Au lieu de mettre à jour toute la matrice de poids, LoRA ne modifie que deux petites matrices.
- Cette approche diminue considérablement les coûts de calcul tout en maintenant les performances du modèle.

##### 1. Entraînement

- **Fine-Tuning** : Les deux matrices de faible rang sont entraînées en utilisant le même processus d'apprentissage supervisé.

##### 2. Inférence

- **Combinaison des Matrices** : Les matrices de faible rang sont multipliées et ajoutées aux poids originaux figés.
- **Remplacement** : Ces poids modifiés remplacent les poids originaux dans le modèle.

## 2. Les modèles du NLP

### Modèle de langage-LLMs

119

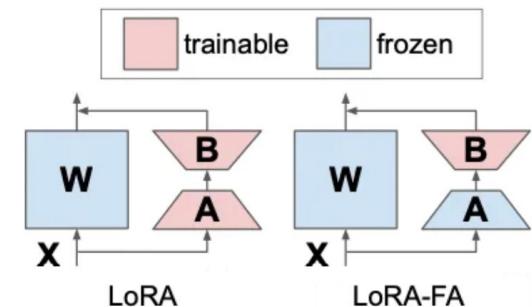
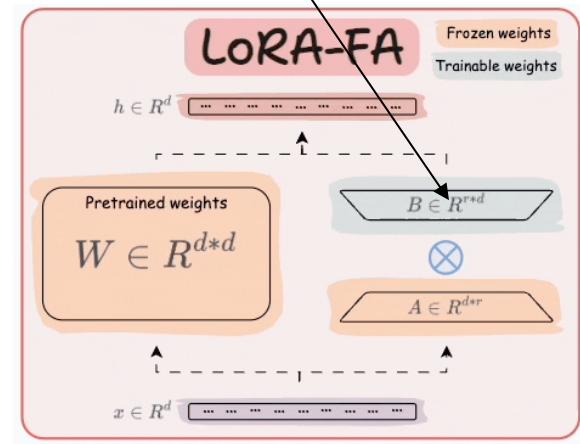
#### D. LLMs (Large Language Models)

#### Techniques de Fine-Tuning LLMs

##### 2. LoRA-FA(LoRA Feature Augmentation)

- LoRA-FA est une extension de LoRA qui introduit une modification spécifique aux mécanismes d'attention.
  - Contrairement à LoRA, LoRA-FA maintient les matrices d'attention fixes et n'applique la décomposition de rang inférieur qu'aux autres parties du modèle.
- Permet une plus grande stabilité dans les modèles où les mécanismes d'attention sont critiques pour la performance.
- En maintenant les matrices d'attention fixes, LoRA-FA peut offrir des gains en performance et en stabilité, surtout dans des scénarios où la stabilité des mécanismes d'attention est cruciale.

Seule la matrice B est entraînable



## 2. Les modèles du NLP

# Modèle de langage-LLMs

120

### D. LLMs (Large Language Models)

### Techniques de Fine-Tuning LLMs

### 3. VeRA (Vector-based Random Matrix Adaptation)

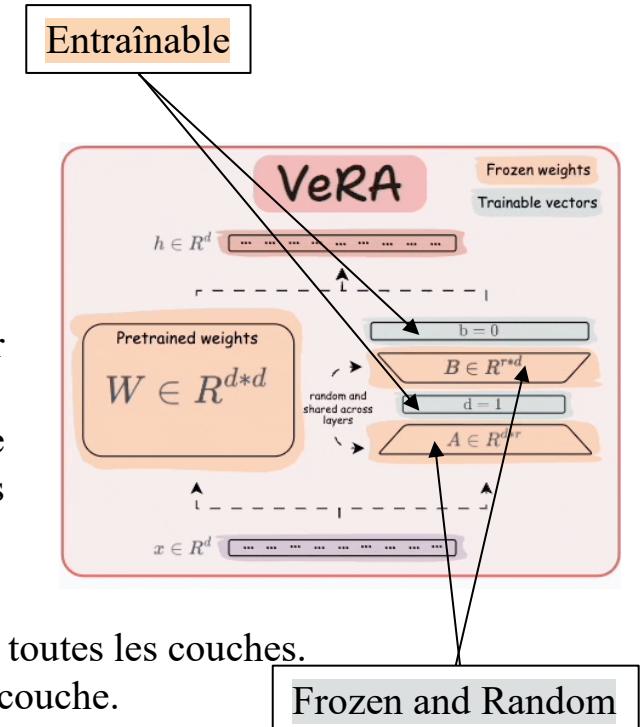
- VeRA réduit encore plus le nombre de paramètres entraînables par rapport à LoRA, tout en conservant les mêmes performances.
- Utilise une seule paire de matrices de rang inférieur partagée entre toutes les couches du modèle et apprend à la place de petits vecteurs de mise à l'échelle.

#### 1. Entraînement

- Une seule paire de matrices de rang inférieur est partagée entre toutes les couches.
- Des petits vecteurs de mise à l'échelle sont appris pour chaque couche.
- Cette approche réduit considérablement le nombre de paramètres à entraîner.

#### 2. Inférence

- Les matrices de rang inférieur et les vecteurs de mise à l'échelle sont combinés pour ajuster les poids du modèle.
- Comme avec LoRA, VeRA garantit que l'impact sur la latence d'inférence est minimal, permettant des réponses rapides et efficaces.



## 2. Les modèles du NLP

### Modèle de langage-LLMs

121

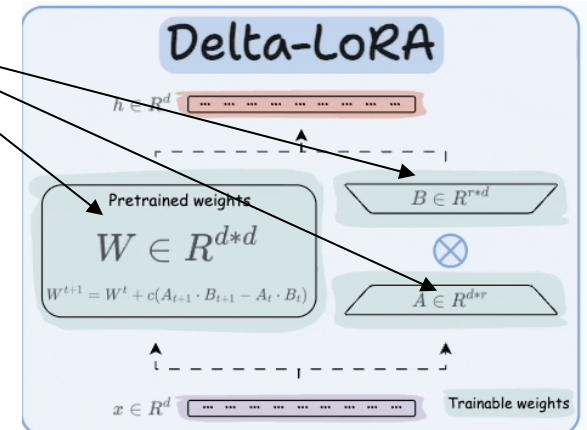
#### D. LLMs (Large Language Models)

#### Techniques de Fine-Tuning LLMs

#### 4. Delta LoRA

- Delta LoRA est une extension de LoRA qui applique des mises à jour sélectives (delta updates) en se concentrant sur les couches où l'apprentissage est le plus significatif, optimisant ainsi davantage l'utilisation de la mémoire.

Entraînable



→ Besoin d'une méthode plus flexible permettant de capturer des variations fines dans les données.

- Principe:
  - Elle introduit une mise à jour basée sur la variation entre les produits des matrices à faible rang à chaque étape d'entraînement.
  - Au lieu d'un ajout statique, Delta-LoRA applique la différence incrémentale entre A et B à chaque étape d'entraînement.
  - Formule:  $W \leftarrow W + \Delta(AB)$

où :

- $\Delta(AB) = AB^{(t)} - AB^{(t-1)}$ ,

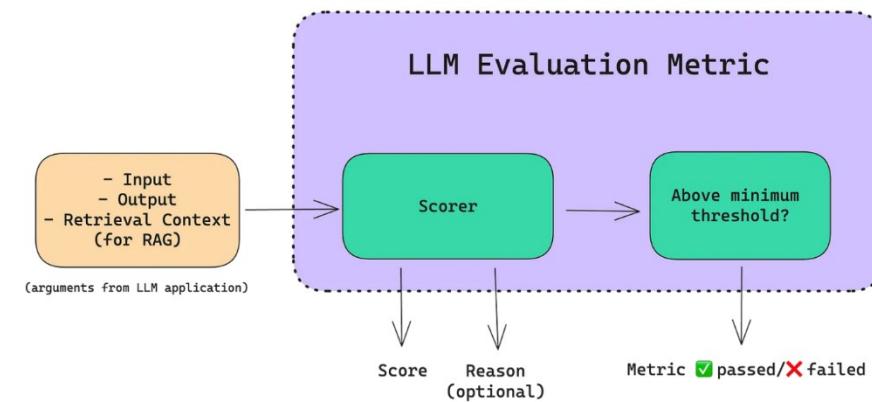
## 2. Les modèles du NLP

### Modèle de langage- LLMs -Évaluation

122

#### Évaluation des Modèles LLM

- Évaluer les performances des LLMs est complexe, à cause de :
  - Résultats non déterministes.
  - Défis uniques comme hallucinations, biais, et toxicité.
  - Résultats variables pour une même entrée (non-déterminisme).
  - Génération de contenu incorrect ou inventé (hallucinations).
  - Contenu inapproprié ou déséquilibré (toxicité).
- Pourquoi l'évaluation est différente ?
  - Contrairement au ML classique, l'évaluation des LLM inclut :
    - Fidélité contextuelle.
    - Pertinence des réponses.
    - Analyse qualitative.



## 2. Les modèles du NLP

### Modèle de langage- LLMs -Évaluation

123

#### Évaluation des Modèles LLM

Deux grandes catégories :

##### 1. Évaluation des réponses

- **Exactitude** : Correspondance parfaite entre la sortie et une réponse correcte.
- **Perplexité** : Évaluation de la probabilité de la sortie générée (qualité du langage).

##### 2. Évaluation de la récupération (pour RAG)

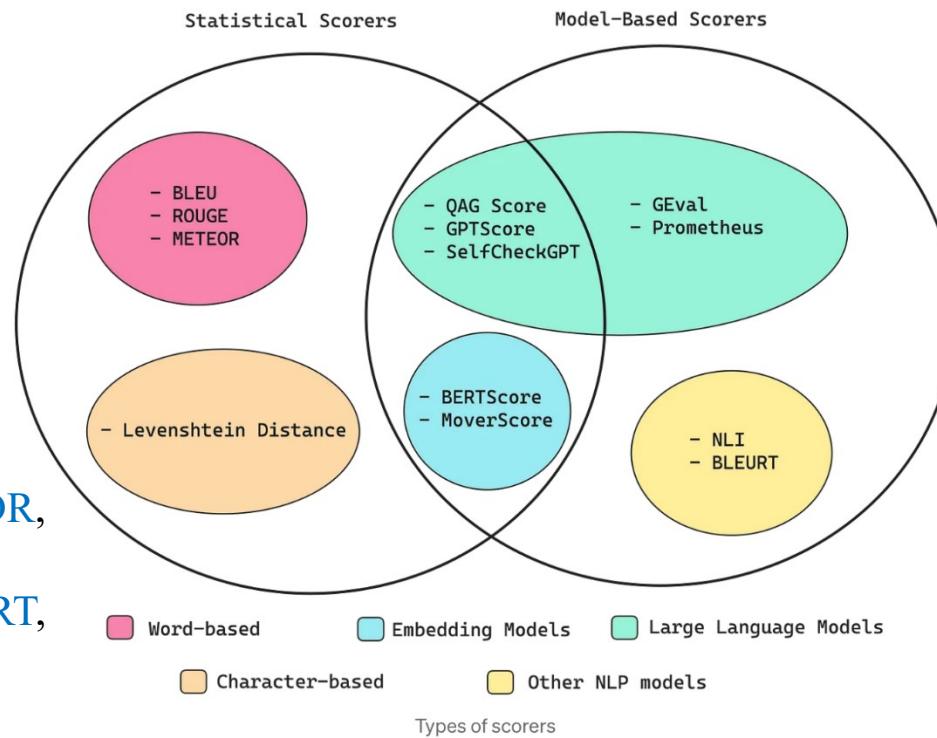
- Pertinence des sources récupérées.
- Précision et rappel contextuel.

Exemples de métriques populaires

- **Statistiques** : BLEU, ROUGE, METEOR, Levenshtein.
- **Basées sur modèles** : BERTScore, BLEURT, GPTScore, QAGScore, GEVal.

##### ▪ Frameworks open-source :

- Exemples : DeepEval, LlamaIndex.



## 2. Les modèles du NLP

# Modèle de langage- LLMs -Évaluation

124

### Principales Métriques d'Évaluation

#### 1. Précision et Performance

- **Perplexité** : Mesure la capacité d'un LLM à prédire le mot suivant. Une perplexité plus faible signifie une meilleure précision.

Hugging Face is a startup based in New York City and Paris  
 $p(\text{word})$

- Formules:  $PP(W) = \left( \frac{1}{P(w_1, w_2, \dots, w_N)} \right)^{\frac{1}{N}}$
- où :
  - $W = (w_1, w_2, \dots, w_N)$  est une séquence de mots.
  - $P(w_1, w_2, \dots, w_N)$  est la probabilité de la séquence de mots W selon le modèle de langage.
  - N est le nombre total de mots dans la séquence.

## 2. Les modèles du NLP

# Modèle de langage- LLMs -Évaluation

125

### Principales Métriques d'Évaluation

#### 1. Précision et Performance

- **Perplexité** : DeepEval avec un modèle open source

```
from deepeval.metrics import PerplexityMetric
from deepeval import assert_test

test_case = LLMTTestCase(
    input="Exemple d'entrée",
    actual_output="Sortie générée",
    expected_output="Sortie attendue"
)

metric = PerplexityMetric()
assert_test(test_case, [metric])
```

- **Exactitude (Accuracy)** : Proportion de prédictions correctes, utile dans les tâches de classification.

## 2. Les modèles du NLP

### Modèle de langage- LLMs -Évaluation

126

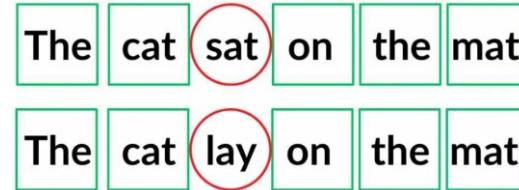
#### Principales Métriques d'Évaluation

##### 1. Précision et Performance

- Scores **BLEU/ROUGE** : Évaluent la qualité des textes générés en fonction de références humaines (précision pour BLEU, rappel pour ROUGE).
- **ROUGE**: se concentre sur le rappel, évaluant combien d'éléments de la référence sont capturés par la génération. Il existe plusieurs variantes, mais la plus courante est le ROUGE-N, qui évalue les n-grammes :

$$\text{ROUGE-N} = \frac{\sum_{\text{réf}} \text{Nombre de n-grammes communs}}{\sum_{\text{réf}} \text{Nombre total de n-grammes}}$$

- Exemple:



→ ROUGE-1 mesure la similarité entre deux phrases en comparant les mots individuellement. Dans l'exemple, cinq mots sur six correspondent, indiquant une similarité de  $(5/6) * 100 = 83\%$ .

## 2. Les modèles du NLP

### Modèle de langage- LLMs -Évaluation

127

- **Principales Métriques d'Évaluation**

- **ROUGE** - Python

```
from rouge import Rouge

# Initialisation de l'évaluateur ROUGE
rouge_scorer = Rouge()

# Texte généré par le modèle (hypothèse) et texte de référence
hypothesis = "To make people trustworthy, you need to trust them"
reference = "The way to make people trustworthy is to trust them"

# Calcul des scores ROUGE
scores = rouge_scorer.get_scores(hyps=hypothesis, refs=reference)

# Extraction du score ROUGE-L F1
rouge_l_f1 = scores[0]["rouge-1"]["f"]

# Résultats
print("ROUGE-L F1 Score:", round(rouge_l_f1, 4))
```

## 2. Les modèles du NLP

### Modèle de langage- LLMs -Évaluation

128

#### • Principales Métriques d'Évaluation

- **BLEU**: Le score BLEU est généralement calculé en comparant des **n-grammes** (des séquences de n mots) entre le texte généré et les références.

$$\text{Score BLEU} = BP \cdot \exp \left( \sum_{n=1}^N w_n \log p_n \right)$$

- $w_n$  est le poids appliqué à chaque n-gramme (souvent  $\frac{1}{N}$ ).
- $p_n$  est la précision des n-grammes.  $= \frac{\text{nombre de n-grams correspondants}}{\text{nombre total de n-grams dans l'hypothèse}}$
- **BP (Brevity Penalty)** est un facteur qui pénalise les hypothèses plus courtes que la référence :

$$BP = \begin{cases} 1 & \text{si } c > r \\ \exp \left( 1 - \frac{r}{c} \right) & \text{si } c \leq r \end{cases}$$

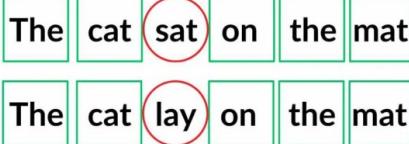
où  $c$  est la longueur de l'hypothèse, et  $r$  la longueur de la référence la plus proche de  $c$ .

## 2. Les modèles du NLP

### Modèle de langage - LLMs - Évaluation

129

#### • Principales Métriques d'Évaluation

- Exemple: 

#### Calcul des n-grammes : (généralement n=4)

##### 1. Unigrammes (n=1)

- Communs : "The", "cat", "on", "the", "mat" (5 sur 6)
- Précision des unigrammes:  $\frac{5}{6} \approx 0.83$

##### 2. Bigrammes (n=2)

- Communs : "The cat", "on the", "the mat" (3 sur 5)
- Précision des bigrammes :  $\frac{3}{5} = 0.6$

##### 3. Trigrammes (n=3)

- Communs : "on the mat" (1 sur 4)

- Précision des trigrammes :  $\frac{1}{4} = 0.25$

##### 4. Quadrigrammes (n=4)

- Quadrigrammes communs : 0
- Précision des quadrigrammes : 0

- Pour simplifier, dans notre cas avec des **poids égaux** et sans appliquer la **pénalité** de longueur, le score **BLEU** approximatif serait :

$$\text{Score BLEU} \approx \exp \left( \frac{1}{4} (\log 0.83 + \log 0.6 + \log 0.25 + \log 0) \right)$$

- Cependant, comme  $\log 0$  est indéfini (tend vers  $-\infty$ ), cela rendrait le score BLEU = 0.
- En pratique, si un des n-grammes a une précision de 0 (aucun n-gramme commun), cela signifie que la similarité entre la phrase candidate et la phrase de référence est très faible, ce qui conduit à un score BLEU de 0.

## 2. Les modèles du NLP

### Modèle de langage- LLMs -Évaluation

130

- **Principales Métriques d'Évaluation**

- **BLEU** - Python

```
from sacrebleu.metrics import BLEU

# Initialisation du calculateur BLEU
bleu_scorer = BLEU()

# Définition de l'hypothèse (texte généré) et de la référence (texte attendu)
hypothesis = "to make people trustworthy, you need to trust them"
reference = "the way to make people trustworthy is to trust them"

# Calcul du score BLEU au niveau de la phrase
score = bleu_scorer.sentence_score(
    hypothesis=hypothesis,
    references=[reference], # Liste des références
)

# Affichage du score BLEU normalisé (0 à 1)
bleu_score_normalized = score.score / 100 # SacreBLEU retourne le score en pourcentage
print(f"BLEU Score (normalisé) : {bleu_score_normalized:.4f}")
```