

Feature engineering

Formatrice: Mously DIAW



INTRODUCTION

Avantages du NLP

01.

Efficacité et précision accrues de la documentation

02.

Permet l'utilisation de chatbots pour l'assistance client

04.

Possibilité de créer automatiquement un résumé de contenu textuel volumineux et complexe

05.

L'analyse des sentiments est plus simple

04.

Permet aux assistants personnels comme Alexa d'interpréter les mots parlés

05.

Des informations analytiques avancées qui étaient auparavant hors de portée

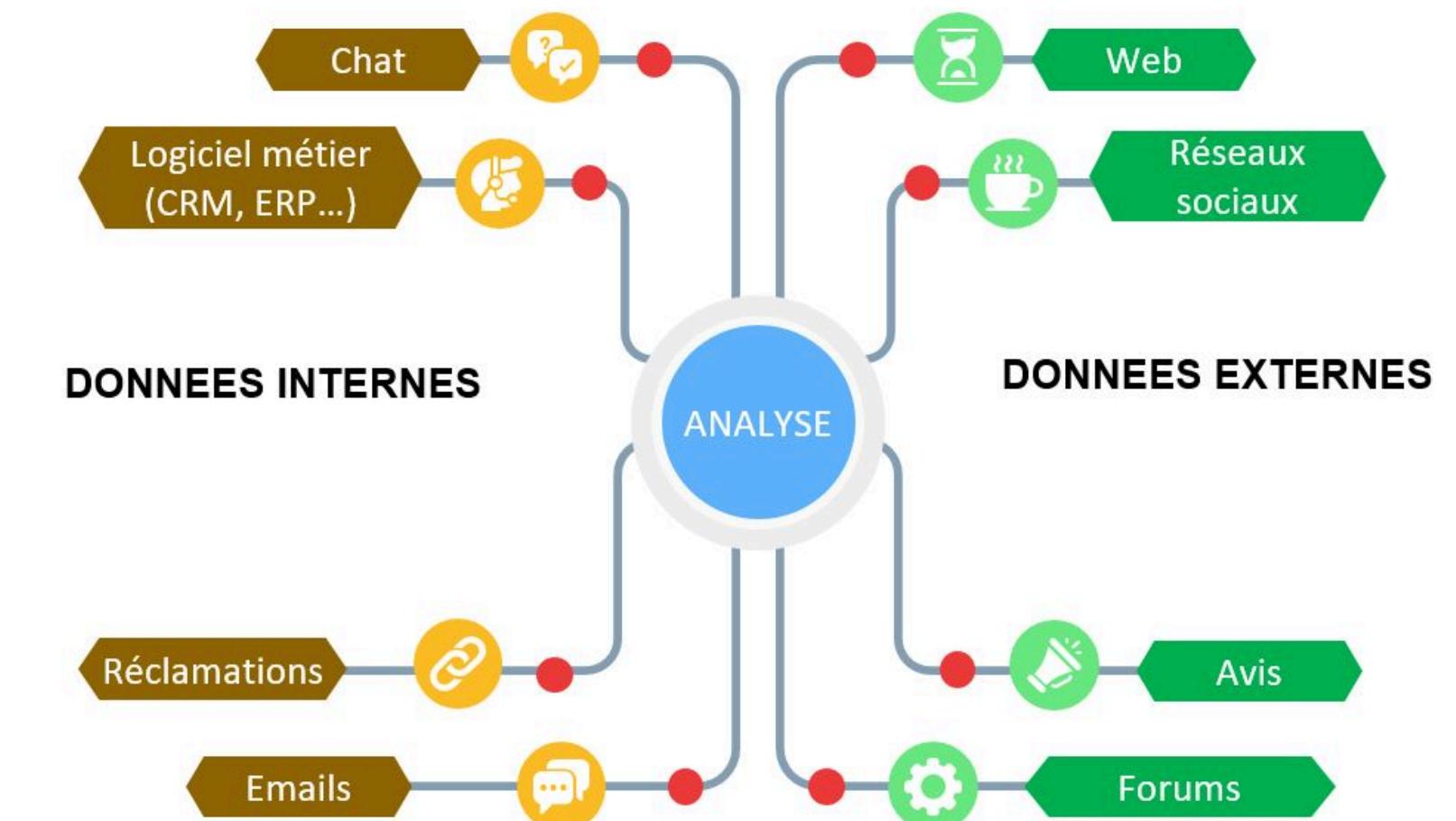
COLLECTE DES DONNÉES

Collecte des données

Sources de Données Textuelles

Les données NLP proviennent de différentes sources, dont :

- **Sites web et blogs** : Articles d'actualité, forums, avis de consommateurs.
- **Réseaux sociaux** : Tweets, posts Facebook, commentaires YouTube.
- **Bases de données ouvertes (OpenData)** : Wikipédia, Project Gutenberg, OpenSubtitles.
- **Documents professionnels** : Emails, contrats, rapports d'entreprise.
- **Données générées par l'utilisateur** : FAQ, transcriptions d'appels.



Collecte des données

Méthodes de Collecte

- **Scraping web** : Utilisation de bibliothèques comme BeautifulSoup (Python) ou Scrapy pour extraire du texte depuis des sites web.
- **APIs** : Twitter API, Google News API pour récupérer des données en temps réel.
- **Corpora préexistants** : Utilisation de jeux de données standard comme IMDB Reviews, CoNLL-2003, Common Crawl.
- **OCR** (Reconnaissance Optique de Caractères) : Conversion d'images ou de documents scannés en texte exploitable.



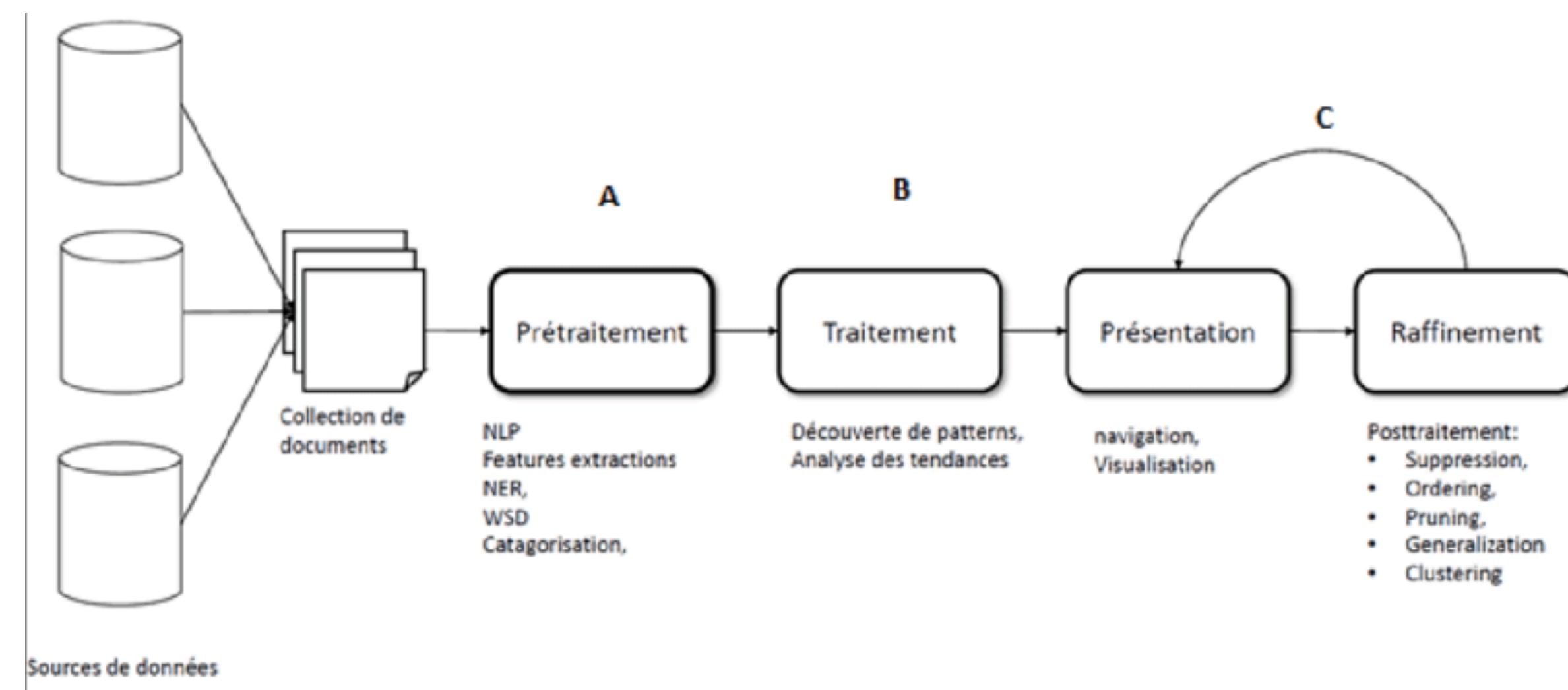
PRÉTRAITEMENT DES DONNÉES



Pré-traitement des données textuelles

Le prétraitement NLP correspond à la préparation d'un texte brut en vue de son analyse par un programme ou un modèle d'apprentissage automatique. Le prétraitement NLP est nécessaire pour préparer le texte dans un format que les modèles d'apprentissage en profondeur peuvent plus facilement analyser.

Il existe plusieurs méthodes de prétraitement NLP utilisées conjointement. Les principales sont les suivantes :

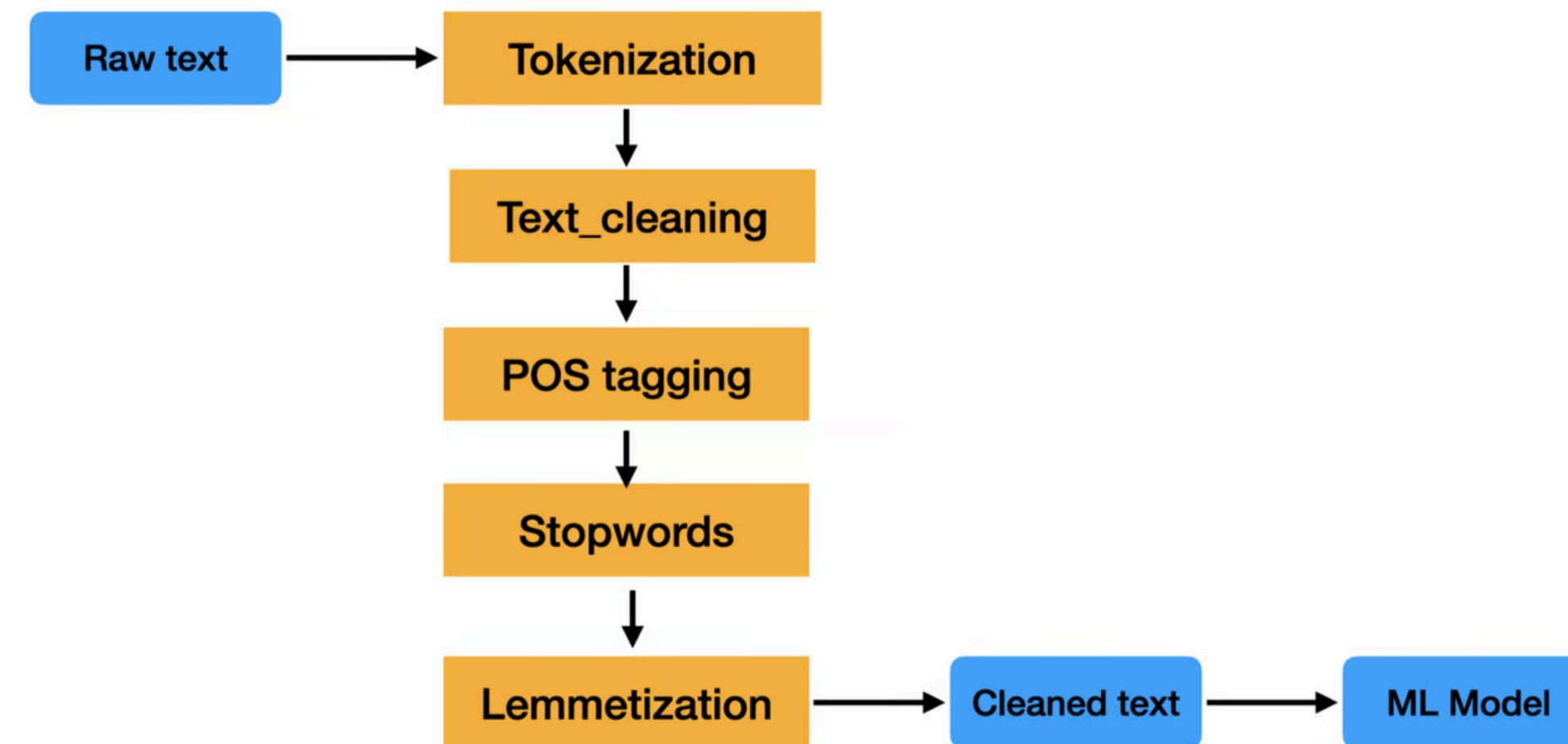


Pipeline de prétraitement de texte

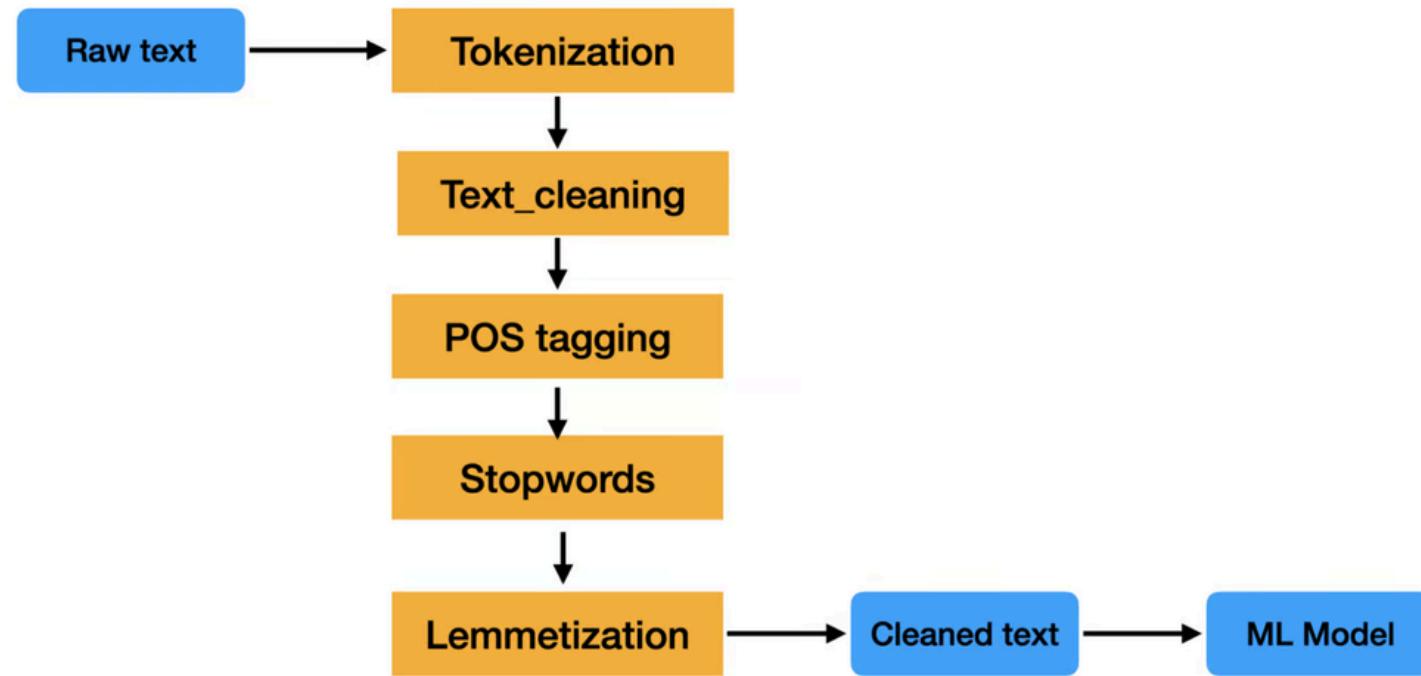
Le prétraitement des données textuelles est une étape importante dans toute tâche de traitement du langage naturel. Il permet de nettoyer et de préparer les données textuelles en vue d'un traitement ou d'une analyse ultérieurs.

Un pipeline de prétraitement de texte est une série d'étapes de traitement appliquées aux données textuelles brutes afin de les préparer à être utilisées dans des tâches de NLP.

Les étapes d'un pipeline de prétraitement de texte peuvent varier, mais elles comprennent généralement des tâches telles que la **tokenisation**, la **suppression** des **mots vides (stopwords)**, le **stemming** et la **lemmatisation**.



Pipeline de prétraitement de texte



Tokenization

La tokenisation est le processus de découpage d'un texte en unités appelées tokens. Un token peut être un mot, un caractère ou une sous-unité de mot.

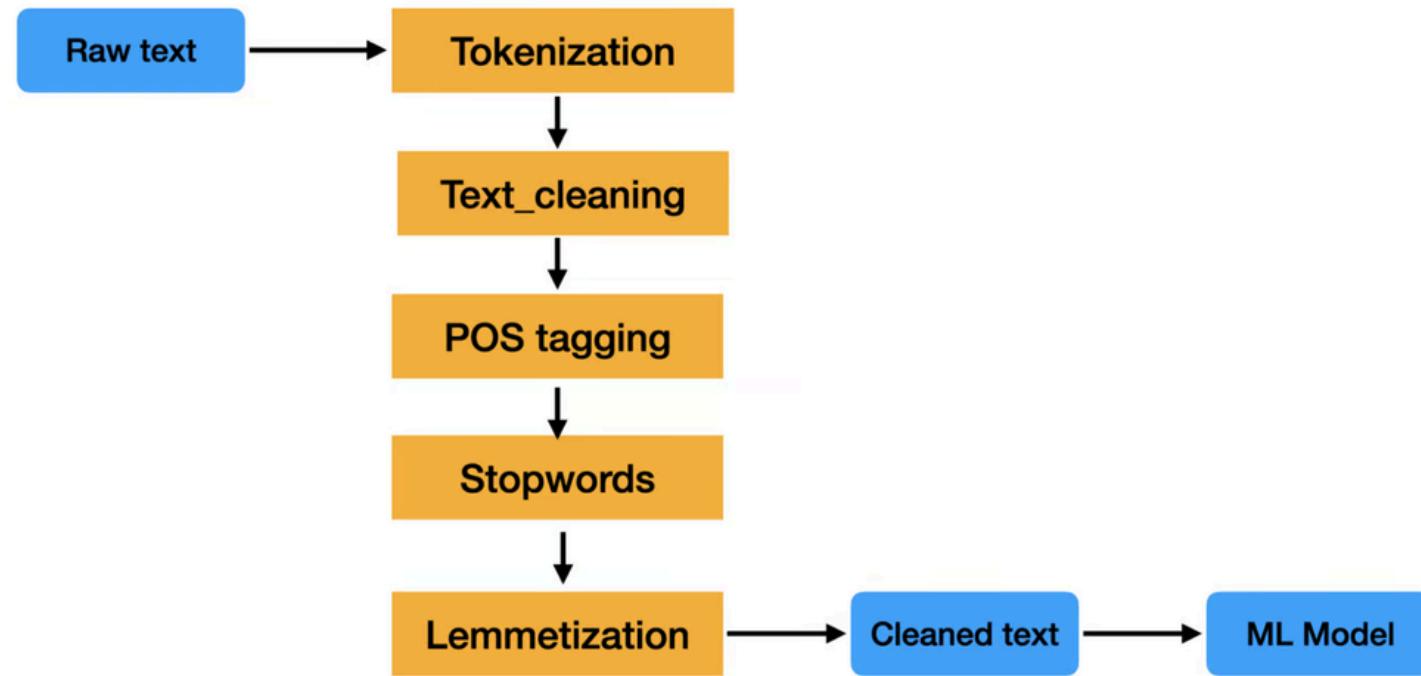
Exemple :

- Texte : "J'aime le NLP !"
- Tokenisation en mots : ["J'", "aime", "le", "NLP", "!"]
- Tokenisation en caractères : ["J", "'", "a", "i", "m", "e", " ", "l", "e", "!", "N", "L", "P", "!"]

Utilité :

- Facilite l'analyse syntaxique et sémantique.
- Première étape du traitement du langage naturel

Pipeline de prétraitement de texte



Cleaning (Nettoyage du Texte)

📌 **Définition :** Le nettoyage de texte consiste à éliminer les éléments indésirables pour rendre les données exploitables par un modèle NLP.

📌 Opérations courantes :

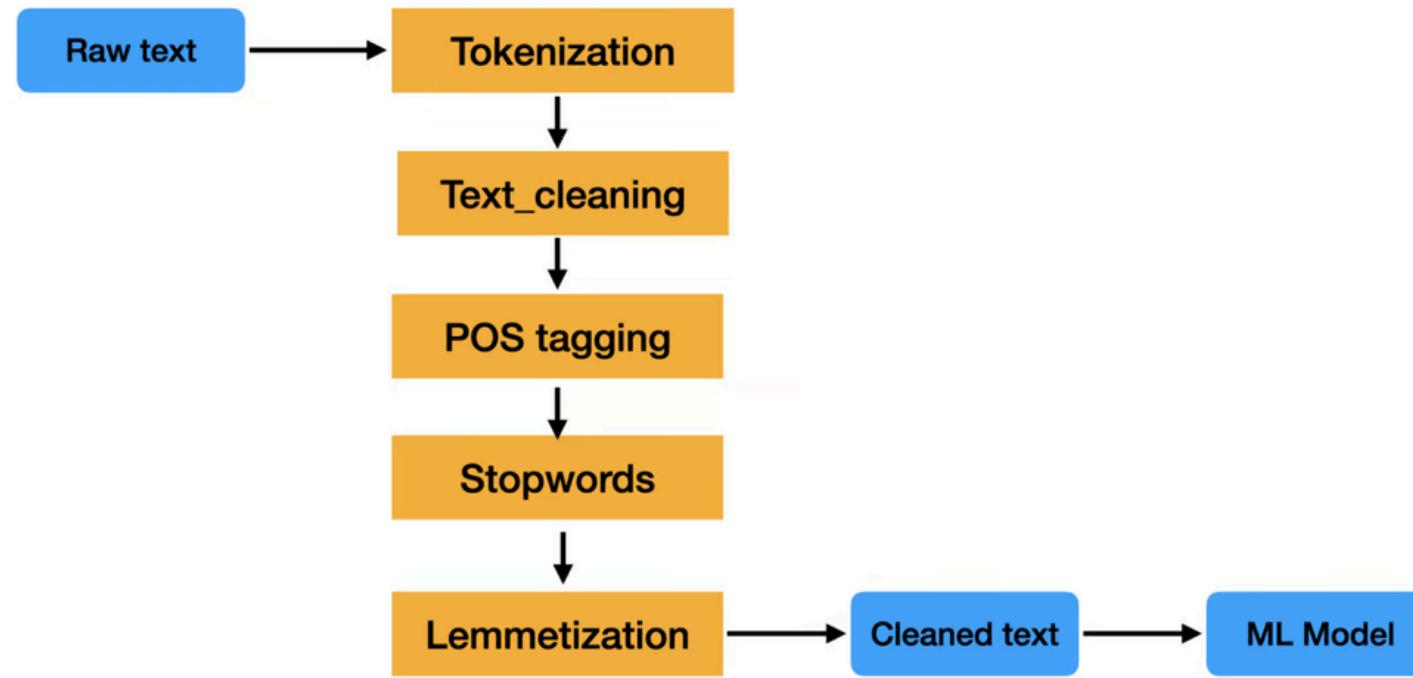
- ✓ Suppression des caractères spéciaux (@, #, \$, %...).
- ✓ Suppression des nombres si non pertinents.
- ✓ Uniformisation (tout en minuscule).
- ✓ Suppression des espaces inutiles et des sauts de ligne.

📌 Exemple :

Texte brut : "Bonjour !!! Comment ça va ??? 😊"

Après nettoyage : "bonjour comment ça va"

Pipeline de prétraitement de texte



POS Tagging (Étiquetage des parties du discours)

✖ **Définition :** L'étiquetage morphosyntaxique (Part-Of-Speech Tagging) consiste à attribuer à chaque mot son **catégorie grammaticale** (verbe, nom, adjectif, etc.).

✖ **Exemple** (avec spaCy ou NLTK) :

Phrase : "Le chien court vite."

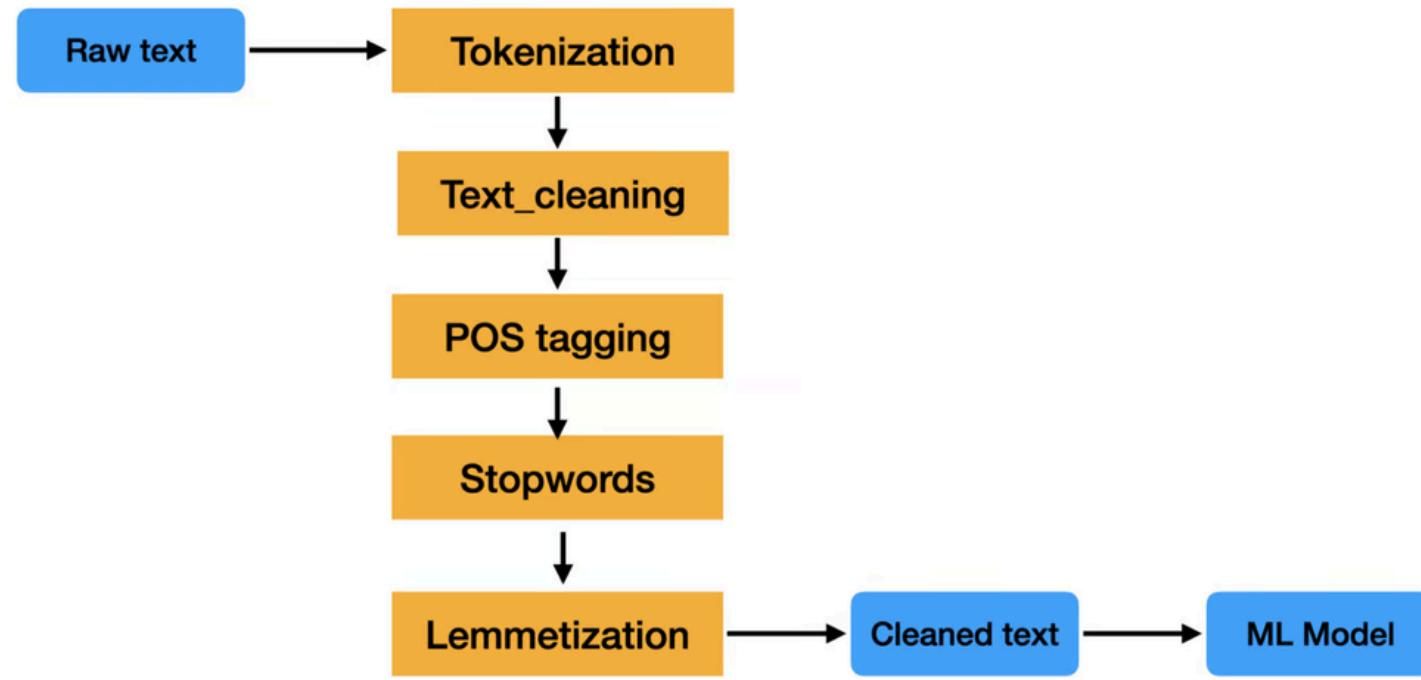
POS tagging :

- "Le" → Déterminant (DET)
- "chien" → Nom (NOUN)
- "court" → Verbe (VERB)
- "vite" → Adverbe (ADV)

✖ **Utilité :**

- Compréhension grammaticale d'une phrase.
- Meilleure analyse de sens pour les modèles NLP.

Pipeline de prétraitement de texte



Stopwords

✖ **Définition :** Les stopwords sont des mots très fréquents dans une langue qui n'apportent pas d'information utile pour certaines analyses (ex. "le", "de", "et", "est", "à"...).

✖ **Exemple :**

Phrase d'origine : "Le chat dort sur le canapé."

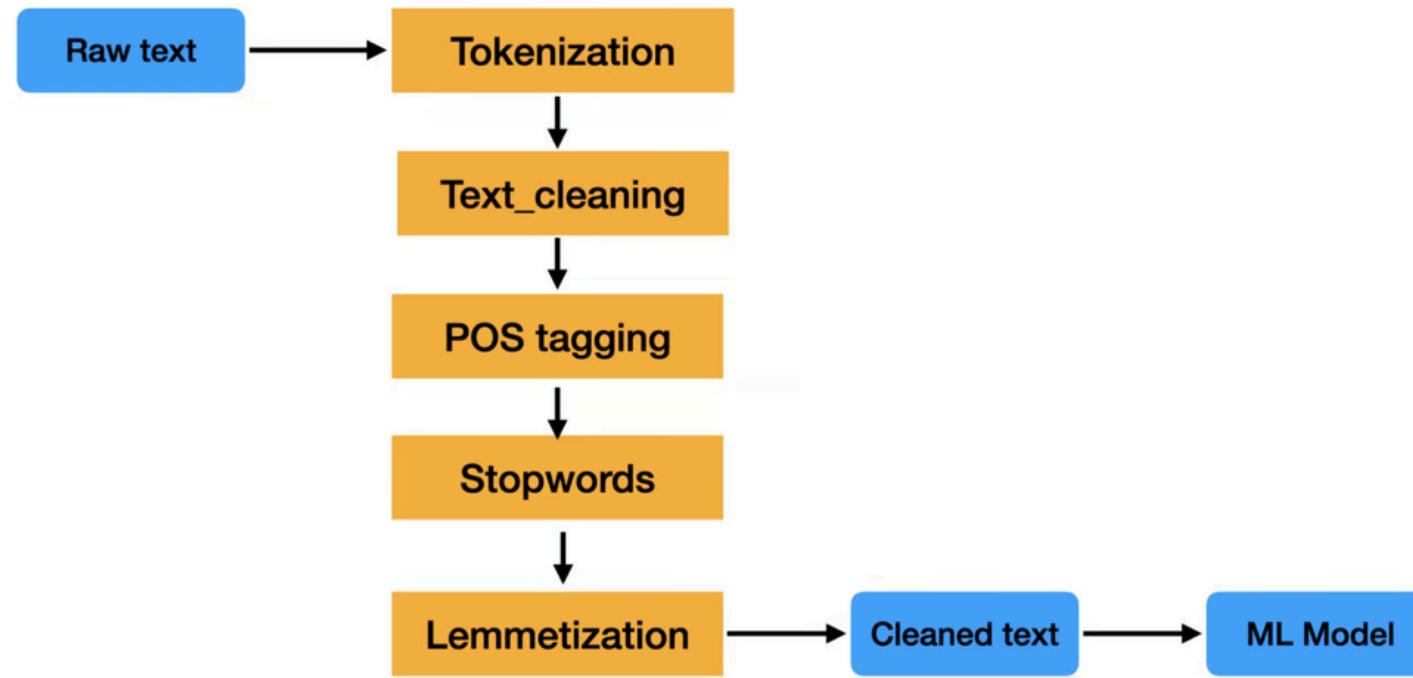
Après suppression des stopwords : "chat dort canapé"

✖ **Utilité :**

- Réduction de la taille des données.
- Concentration sur les mots clés importants.

⚠ **Attention :** Dans certains cas (comme l'analyse syntaxique), il ne faut pas supprimer les stopwords !

Pipeline de prétraitement de texte



Stemming

✖ **Définition :** Le stemming consiste à réduire un mot à sa racine en supprimant les suffixes et préfixes. Cette approche est brute et parfois inexacte.

✖ Exemple :

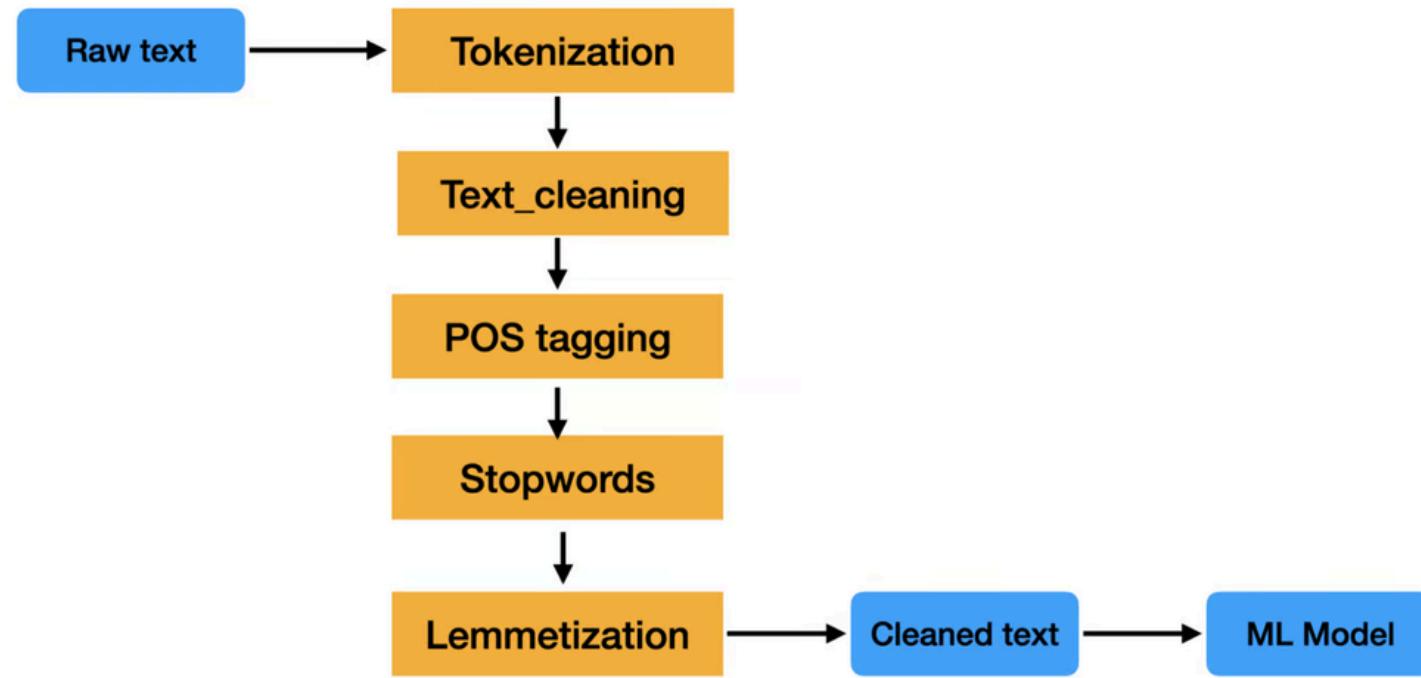
- "manger", "mangé", "mangeons" → "mang"
- "courir", "court", "courra" → "cour"

✖ Utilité :

- Réduit la complexité du texte.
- Utile pour des tâches comme la recherche d'information.

⚠ **Limite :** Le stemming ne tient pas compte de la grammaire, il peut produire des racines incorrectes.

Pipeline de prétraitement de texte



Lemmatization

📌 **Définition :** La lemmatisation ramène un mot à sa forme canonique (lemme) en prenant en compte sa catégorie grammaticale.

📌 Exemple :

- "manger", "mangé", "mangeons" → "manger"
- "courir", "court", "courra" → "courir"

📌 Utilité :

- Plus précis que le stemming.
- Respecte la grammaire et le sens des mots.

📌 Différence avec le stemming :

Méthode	Résultat pour "courait"	Résultat pour "manges"
Stemming	cour	mang
Lemmatisation	courir	manger

Pipeline de prétraitement de texte: résumé

Concept	Définition
Tokenization	Découpage du texte en unités (mots, phrases, caractères).
Cleaning	Nettoyage du texte (suppression des caractères spéciaux, uniformisation, etc.).
POS Tagging	Attribution de la catégorie grammaticale des mots (nom, verbe, adjectif...).
Stopwords	Mots fréquents souvent supprimés pour alléger l'analyse.
Stemming	Réduction d'un mot à sa racine (approche brute).
Lemmatisation	Réduction d'un mot à sa forme canonique (plus précis que le stemming).

RÉPRESENTATION VECTORIELLE



Techniques de représentation vectorielle des données textuelles

En traitement du langage naturel (NLP), la représentation vectorielle des données textuelles est une étape cruciale pour permettre aux modèles de comprendre et de manipuler le langage humain. Voici un aperçu des principales techniques utilisées pour transformer le texte en vecteurs numériques.

- One-Hot Encoding
- Bag of Words (BoW)
- Term Frequency-Inverse Document Frequency (TF-IDF)
- Représentation Dense (Word Embeddings)
- Représentation Contextuelle (Transformers)

One-Hot Encoding

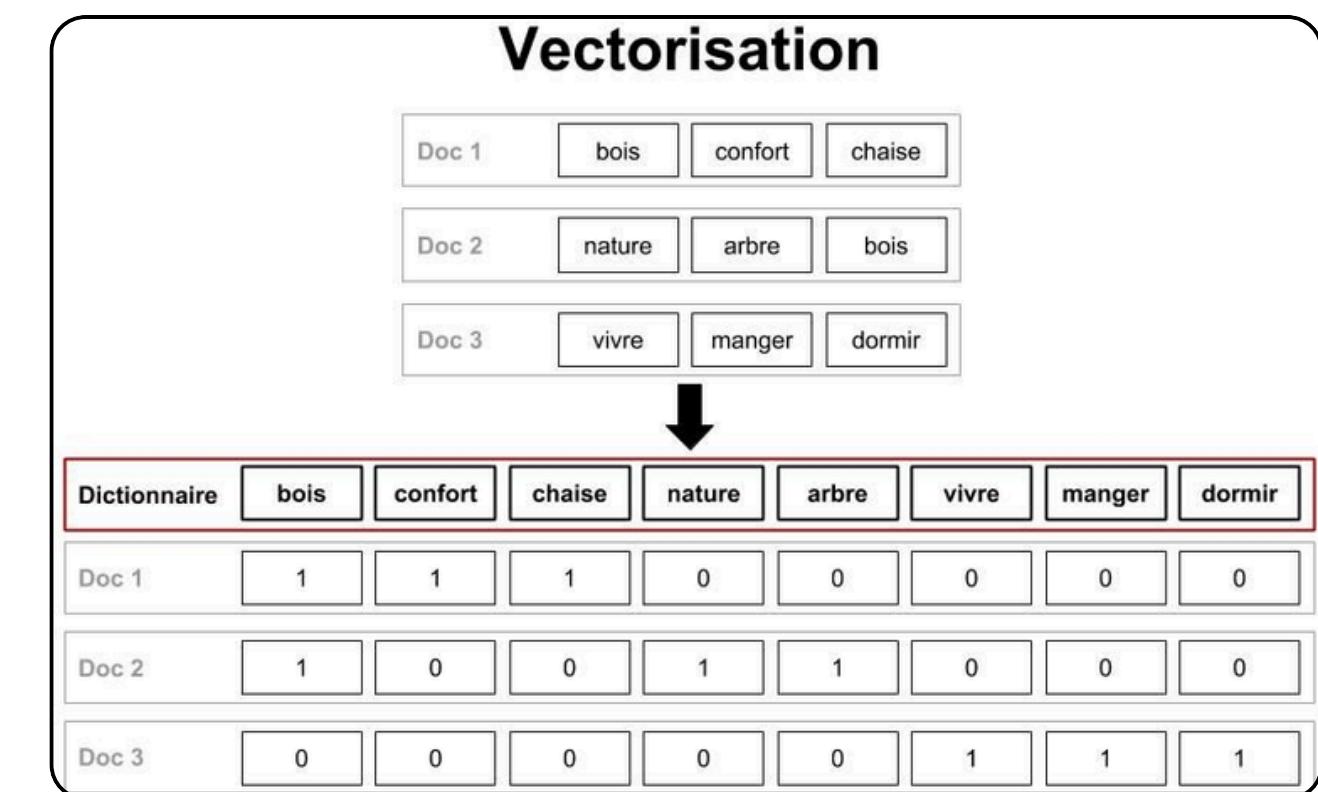
Chaque mot est représenté par un **vecteur binaire de taille égale au vocabulaire** total. Le vecteur a une valeur de 1 à la position correspondant au mot et 0 partout ailleurs.

- **Exemple :**

- Vocabulaire : ["chat", "chien", "oiseau"]
- Représentation :
 - "chat" → [1, 0, 0]
 - "chien" → [0, 1, 0]
 - "oiseau" → [0, 0, 1]

- **Limites :**

- La dimension du vecteur est égale à la taille du **vocabulaire**, ce qui peut être **très grand**.
- **Ne capture pas les relations sémantiques** entre les mots.
- Comme la plupart des documents utilisent généralement un très petit sous-ensemble des mots présents dans le corpus, la matrice résultante aura de **nombreuses valeurs égales à zéro** (généralement plus de 99 % d'entre elles: **matrice sparse**).



Représentation par Fréquence: Bag of Word (BoW)

Un document est représenté par un vecteur où chaque dimension correspond à la **fréquence d'un mot** dans le vocabulaire.

- **Exemple :**

- Document : "Le chat mange le poisson."
- Vocabulaire : ["le", "chat", "mange", "poisson"]
- Représentation : [2, 1, 1, 1] (car "le" apparaît 2 fois, les autres mots 1 fois).

- **Limites :**

- Ignore l'ordre des mots.
- Ne capture pas le sens des mots.

Document D1		<i>The child makes the dog happy</i>					
		the: 2, dog: 1, makes: 1, child: 1, happy: 1					
Document D2		<i>The dog makes the child happy</i>					
		the: 2, child: 1, makes: 1, dog: 1, happy: 1					
		child	dog	happy	makes	the	BoW Vector representations
D1		1	1	1	1	2	[1,1,1,1,2]
D2		1	1	1	1	2	[1,1,1,1,2]

TF-IDF (Term Frequency-Inverse Document Frequency)

Améliore le BoW en pondérant les mots par leur importance dans le document par rapport à l'ensemble du corpus.

- **Formule:**

- TF (Term Frequency) : Fréquence du mot dans le document.
- IDF (Inverse Document Frequency) : Logarithme de l'inverse de la fréquence du mot dans le corpus.
- $TF-IDF = TF * IDF$

$$TF(w) = \frac{\text{Nombre d'occurrences du mot } w \text{ dans un document}}{\text{Nombre total de mots dans le document}}$$
$$IDF(w) = \log \left(\frac{\text{Nombre total de documents}}{\text{Nombre de documents contenant le mot } w} \right)$$
$$TF - IDF(w) = TF(w) \times IDF(w)$$

- **Exemple :**

- **Document** : "Le chat mange le poisson."
- **Corpus** : ["Le chat mange.", "Le poisson est frais."]
- **Représentation** : [0.0, 0.58, 0.58, 0.58] (les mots rares comme "chat", "mange", "poisson" ont un poids plus élevé).

Plus un **mot est fréquent dans un document spécifique mais rare dans le corpus global**, plus sa **valeur TF-IDF sera élevée**.

- **Avantages :**

- Réduit l'importance des mots trop fréquents.
- Améliore la pertinence pour des tâches comme la recherche d'information.

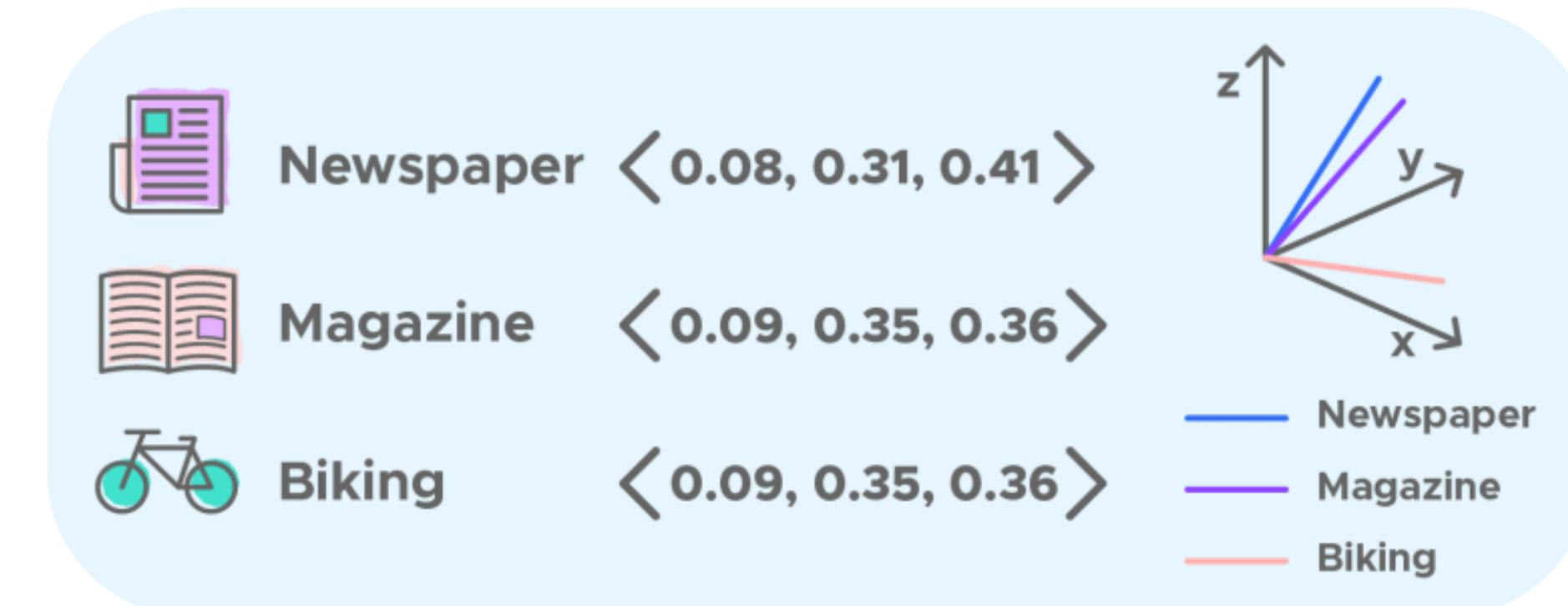
- **Limites :**

- Toujours basé sur des fréquences
- Ne capture pas le sens des mots (ex. "voiture" et "automobile" sont vus comme différents)

Plongements de mots (Word Embeddings)

Contrairement à BoW et TF-IDF, qui traitent les mots comme indépendants, les word embeddings apprennent des représentations continues et denses qui capturent les relations sémantiques entre les mots.

- Word2Vec
- GloVe (Global Vectors for Word Representation)
- FastText (Facebook AI)



Word Embeddings: Word2Vec (1/4)

- **Principe** : Représente chaque mot par un vecteur dense basé sur son contexte.
- **Deux architectures** :
 - **CBOW** (Continuous Bag of Words) : Prédit un mot à partir de son contexte.
 - **Skip-gram** : Prédit le contexte à partir d'un mot.
- **Exemple** : Word2Vec capture les relations sémantiques entre les mots. Par exemple:
 - Roi - Homme + Femme ≈ Reine
 - France - Paris + Berlin ≈ Allemagne

Avantages :

- Capture les relations sémantiques entre les mots.
- Réduit la dimensionnalité par rapport à BoW.

Limites :

- Besoin de beaucoup de données pour être efficace.
- Ne gère pas bien la polysémie (ex. "banc" peut être un siège ou une institution financière).

Word Embeddings: Word2Vec (2/4)

Continuous Bag of Words (CBOW)

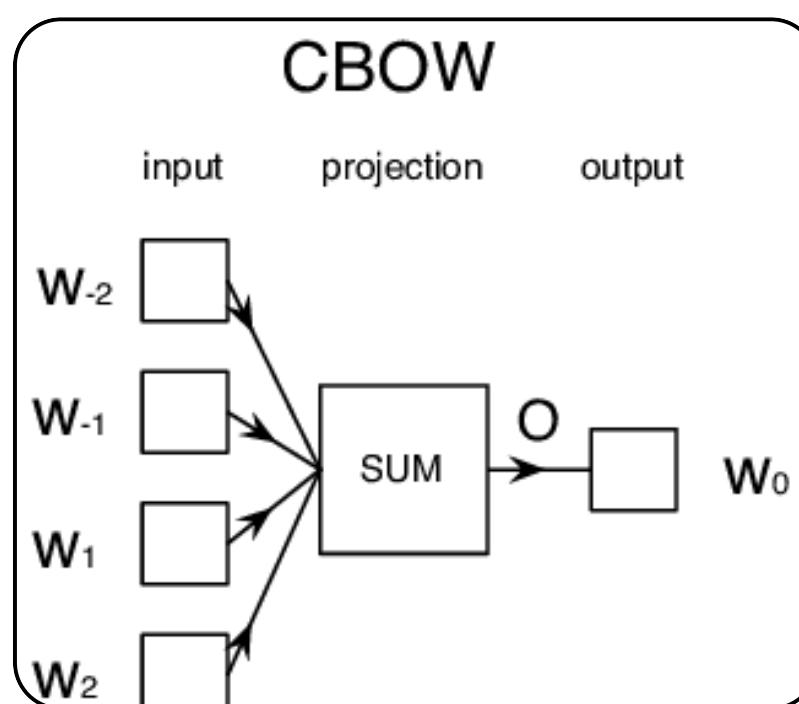
Dans cette approche, le modèle prédit le mot cible à partir d'une fenêtre de contexte fixe de mots environnants. Par exemple, étant donné le contexte de mots autour d'un mot manquant (par exemple, "Le chat est assis sur le ___"), CBOW essaie de prédire "tapis".

Avantages :

- Efficace pour les petits corpus.
- Moins coûteux en calcul que Skip-Gram.

Limites :

- Moins performant pour les mots rares, car le contexte moyen peut être dominé par des mots fréquents.



Formule :

Le modèle maximise la probabilité :

$$P(w_t | w_{t-k}, w_{t-k+1}, \dots, w_{t+k})$$

où w_t est le mot cible et w_{t-k}, \dots, w_{t+k} sont les mots du contexte.

Word Embeddings: Word2Vec (2/4)

Skip-Gram

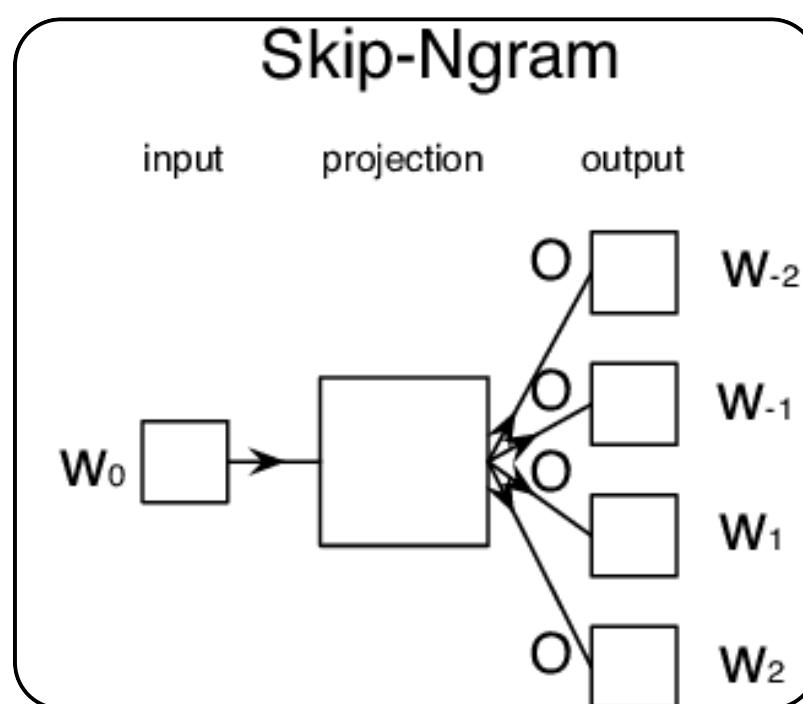
Il s'agit de l'inverse de CBOW, où le modèle utilise un seul mot pour prédire le contexte environnant. Par exemple, avec le mot "chat", le modèle essaiera de prédire des mots comme "Le", "est", "assis", "sur", "le", dans la fenêtre de contexte.

Avantages :

- Mieux adapté pour les mots rares, car il se concentre sur un seul mot à la fois.
- Performances supérieures sur les grands corpus.

Limites :

- Plus coûteux en calcul que CBOW, surtout pour les grands corpus.



Formule :

Le modèle maximise la probabilité :

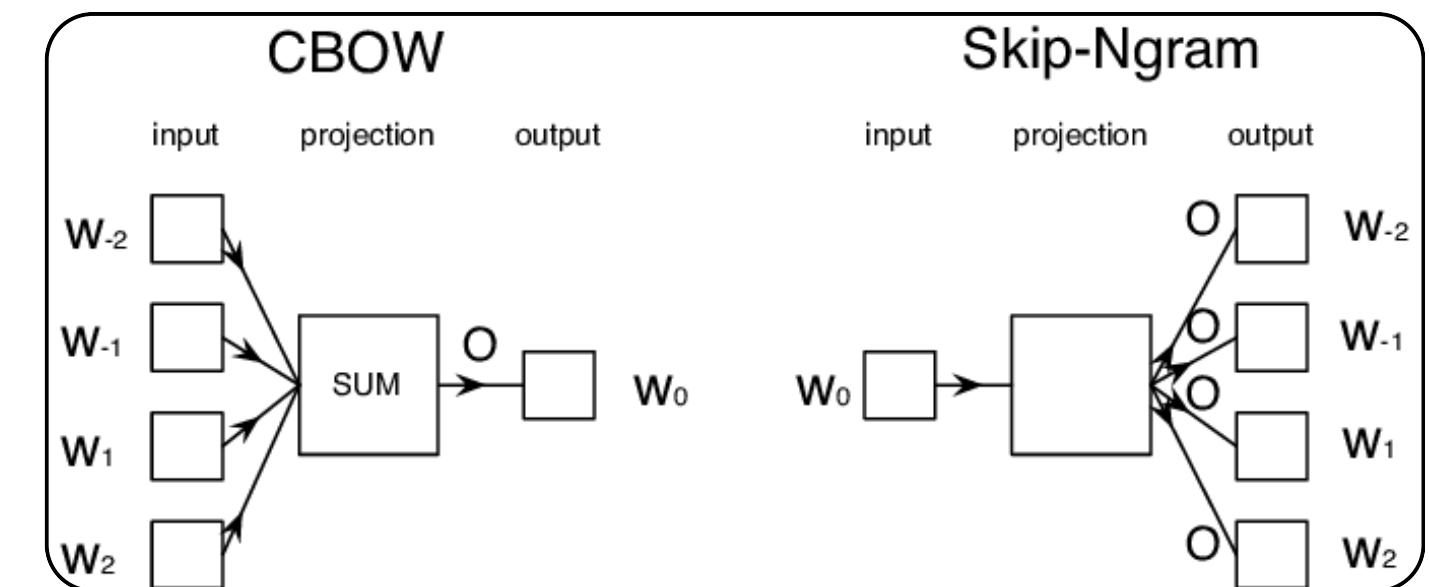
$$P(w_{t-k}, w_{t-k+1}, \dots, w_{t+k} | w_t)$$

où w_t est le mot cible et w_{t-k}, \dots, w_{t+k} sont les mots du contexte.

Word Embeddings: Word2Vec (4/4)

Comparaison entre CBOW et Skip-Gram

Aspect	CBOW	Skip-Gram
Objectif	Prédit le mot cible à partir du contexte.	Prédit le contexte à partir du mot cible.
Performance	Rapide, efficace pour les petits corpus.	Lent, mais performant pour les grands corpus.
Mots rares	Moins performant.	Mieux adapté.
Usage typique	Corpus petits à moyens.	Corpus grands ou mots rares.



Word Embeddings: GloVe (Global Vectors for Word Representation)

Principe : Contrairement à Word2Vec qui apprend à partir de séquences de mots, GloVe apprend à partir des cooccurrences des mots dans un grand corpus. La **matrice de co-occurrence** compte **combien de fois chaque paire de mots apparaît ensemble** dans un contexte donné (par exemple, une fenêtre de mots)

Exemple de relations sémantiques :

- Roi - Homme + Femme ≈ Reine
- France - Paris + Londres ≈ Royaume-Uni

✓ Avantages :

- Capture mieux les relations globales entre mots.
- Les vecteurs appris reflètent directement les relations de co-occurrence, ce qui facilite l'interprétation
- Plus performant sur certaines tâches de classification et de recherche d'information.

✳ Limites :

- Nécessite une grande matrice de cooccurrence.
- Moins flexible que Word2Vec pour les nouvelles phrases.

Formule :

GloVe minimise la fonction de coût suivante :

$$J = \sum_{i,j=1}^V f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

où :

- X_{ij} : Nombre de co-occurrences entre le mot i et le mot j .
- w_i : Vecteur du mot i .
- \tilde{w}_j : Vecteur du mot j (vecteur de contexte).
- b_i, \tilde{b}_j : Biais pour les mots i et j .
- $f(X_{ij})$: Fonction de pondération pour réduire l'influence des co-occurrences rares ou fréquentes.

2. Fonction de Pondération $f(X_{ij})$

La fonction de pondération $f(X_{ij})$ est utilisée pour éviter que les mots très fréquents ou très rares ne dominent l'apprentissage. Elle est définie comme suit :

$$f(X_{ij}) = \begin{cases} \left(\frac{X_{ij}}{x_{\max}}\right)^\alpha & \text{si } X_{ij} < x_{\max} \\ 1 & \text{sinon} \end{cases}$$

où x_{\max} et α sont des hyperparamètres (par exemple, $x_{\max} = 100, \alpha = 0.75$).

Word Embeddings: FastText

FastText est une méthode développée par Facebook AI Research (FAIR) pour apprendre des représentations vectorielles de mots, similaire à Word2Vec, mais avec quelques améliorations importantes, notamment la capacité à gérer des mots rares ou inconnus.

Comment cela fonctionne :

- FastText **décompose les mots en n-grammes de caractères**. Par exemple, pour le mot "chat", FastText pourrait utiliser des n-grammes comme "<ch", "cha", "hat", "at>", où les signes "<" et ">" indiquent le début et la fin du mot
- Au lieu de représenter un mot par un seul vecteur, **FastText représente un mot par la somme des vecteurs** de ses **n-grammes de caractères**.
- FastText **utilise les architectures Skip-gram ou CBOW** pour prédire les contextes des n-grams de caractères

✓ Avantages :

- Meilleure gestion des mots rares et nouvelles langues.
- Capture des relations morphologiques (ex. pluriels, conjugaisons).

✗ Limites :

- Plus lent à entraîner que Word2Vec.
- Nécessite plus de stockage.

Word Embeddings: Word2Vec vs GloVe vs FastText

FastText est une méthode développée par Facebook AI Research (FAIR) pour apprendre des représentations vectorielles de mots, similaire à Word2Vec, mais avec quelques améliorations importantes, notamment la capacité à gérer des mots rares ou inconnus.

Comment cela fonctionne :

- FastText **décompose les mots en n-grammes de caractères**. Par exemple, pour le mot "chat", FastText pourrait utiliser des n-grammes comme "<ch", "cha", "hat", "at>", où les signes "<" et ">" indiquent le début et la fin du mot
- Au lieu de représenter un mot par un seul vecteur, **FastText représente un mot par la somme des vecteurs** de ses **n-grammes de caractères**.
- FastText **utilise les architectures Skip-gram ou CBOW** pour prédire les contextes des n-grams de caractères

Avantages :

- Meilleure gestion des mots rares et nouvelles langues.
- Capture des relations morphologiques (ex. pluriels, conjugaisons).

Limites :

- Plus lent à entraîner que Word2Vec.
- Nécessite plus de stockage.

WORD EMBEDDINGS: WORD2VEC VS GLOVE VS FASTTEXT

	Word2Vec	GloVe	FastText
Méthode	Réseau de neurones (prédiction locale)	Factorisation de matrice de co-occurrence	Modèle de prédiction avec n-grammes de caractères
Approche	Locale (contexte immédiat des mots). Représentation: Mots entiers.	Globale (co-occurrence sur tout le corpus). Représentation: Mots entiers.	Par sous-mots (n-grams de caractères).
Gestion des mots rares / inconnus	Moins efficace pour les mots rares (vocabulaire fixe)	Moins efficace pour les mots rares (vocabulaire fixe)	Très efficace pour les mots rares ou inconnus (grâce aux n-grams)
Complexité	Relativement simple et rapide	Nécessite de construire une matrice de co-occurrence	Plus complexe, coûte plus de ressources
Utilisation des relations sémantiques	Bonne capture des relations locales Capture morphologique: non	Excellente pour les relations globales (ex. analogies). Capture morphologique: non	Bonne capture des relations, particulièrement pour les mots avec des sous-mots communs. Capture morphologique: préfixes, suffixes, racines
Adapté pour	Grands corpus avec des mots fréquents	Corpus où les relations globales sont importantes	Langues morphologiquement complexes ou avec beaucoup de mots rares
Exemples d'utilisation	Recherche de similarité de mots, analyse de texte	Recherche de similarité, applications multilingues	Traduction automatique, analyse de texte dans des langues avec des mots rares.

Représentation contextuelle: ELMo, BERT, USE, GPT,...

- Les modèles de représentation contextuelle comme **BERT** (Bidirectional Encoder Representations from Transformers) et **USE** (Universal Sentence Encoder) ont transformé la manière dont nous comprenons et utilisons les représentations des mots et des phrases en NLP
- **Contrairement** à des méthodes comme **Word2Vec**, **GloVe** et **FastText** qui produisent des **représentations statiques** (c'est-à-dire les mêmes pour un mot donné quel que soit son contexte), ces modèles **génèrent des représentations dynamiques et contextuelles**, où le **sens d'un mot ou d'une phrase peut varier en fonction de son contexte**.

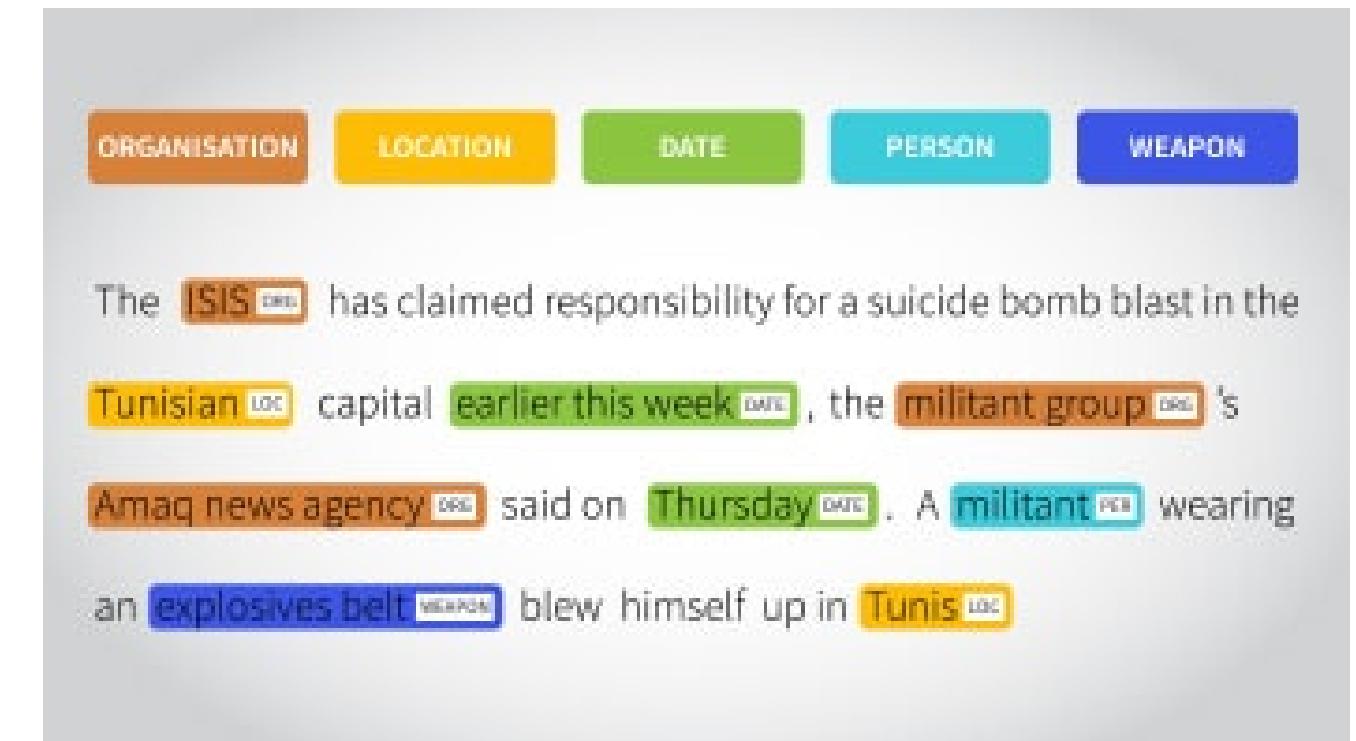
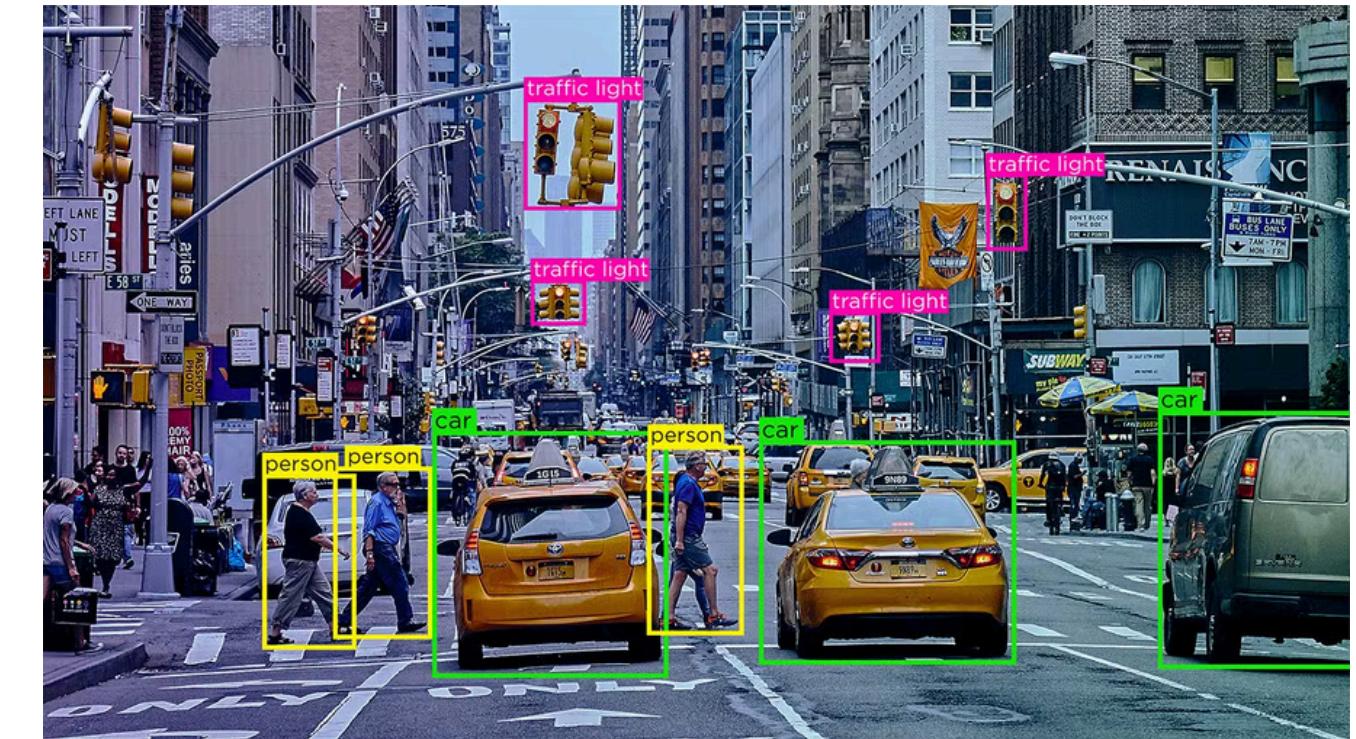
NLP: ANNOTATION ET ÉTIQUETAGE DES DONNÉES

Qu'est-ce que l'annotation de données ?

L'annotation des données est le processus d'attribution, de marquage ou d'étiquetage des données pour aider les algorithmes d'apprentissage automatique à comprendre et à classer les informations qu'ils traitent.

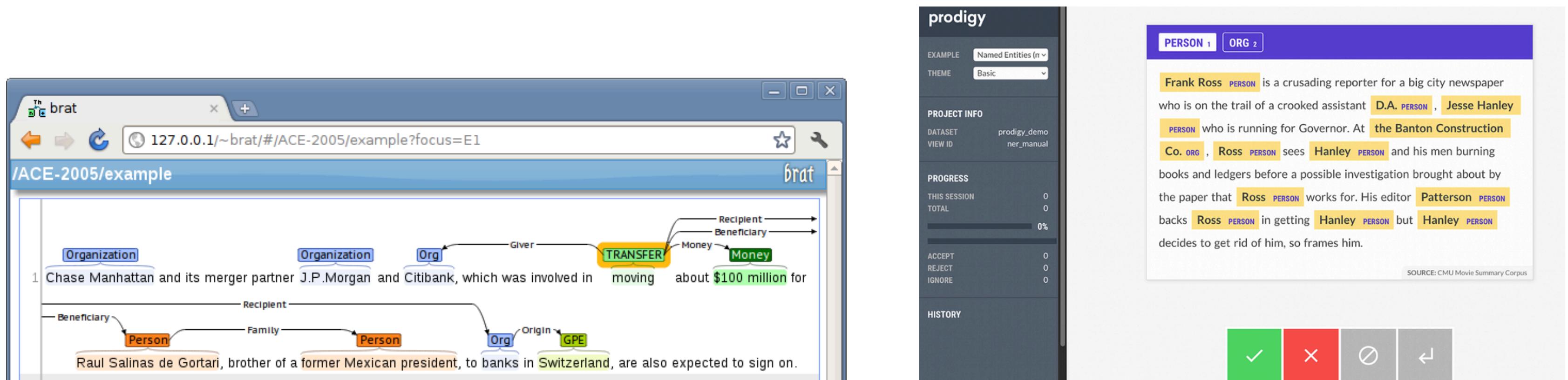
Ce processus est essentiel pour former des modèles d'IA, leur permettant de comprendre avec précision divers types de données, tels que des images, des fichiers audio, des séquences vidéo ou du texte. Cela peut inclure :

- **Annotation de texte** : Par exemple, l'annotation des entités nommées dans un texte (comme des noms de personnes, de lieux ou d'organisations) ou des relations entre différentes entités.
- **Annotation de sentiment** : L'étiquetage des opinions ou sentiments exprimés dans un texte (positif, négatif, neutre).
- **Annotation de syntaxe ou de dépendances** : Marquage des structures grammaticales ou des relations entre les mots dans une phrase.
- **Annotation de catégories ou de labels** : Par exemple, annoter un article de presse en fonction de sa catégorie (politique, sport, économie, etc.).



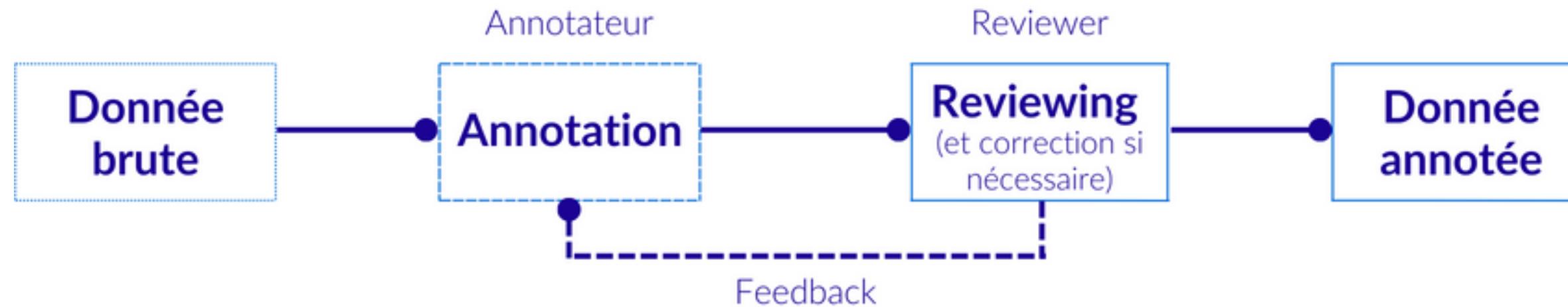
Techniques et outils pour l'annotation

- **Annotation manuelle** : Des annotateurs humains ajoutent des labels aux données, ce qui peut être coûteux en temps et en ressources.
- **Outils d'annotation** : Des outils comme BRAT (outil open source pour l'annotation de texte) ou Prodigy (outil interactif pour l'annotation de données) sont utilisés pour faciliter le processus.
- **Annotation semi-automatique** : Utilisation de modèles d'IA pour suggérer des annotations, que les annotateurs humains peuvent valider ou corriger



Importance de la qualité des annotations

- Des **annotations de qualité** sont essentielles pour obtenir des **modèles de NLP performants**.
- Des **incohérences ou des erreurs d'annotation** peuvent **entraîner une mauvaise performance des modèles**, d'où l'importance de former des annotateurs et de vérifier la qualité des annotations.



Exemple de workflow d'annotation des données avec review.

Outils et bibliothèques pour le NLP

- Il existe de nombreux outils et bibliothèques pour le NLP: NLTK, spaCy, Gensim, OpenNLP, TextBlob, Stanforw NLP, scikit-learn, CorenLP, PyTorch, TensorFlow et l'écosystème Hugging Face, etc.
- Outils d'annotation: Prodigy, BRAT (Brat Rapid Annotation Tool), Doccano



Ressources

<https://www.cegos.fr/ressources/mag/marketing-communication/les-4-sources-du-big-data>
<https://www.cloudflare.com/fr-fr/learning/ai/natural-language-processing-nlp/>
<https://www.datacamp.com/fr/tutorial/text-classification-python>
<https://blog.coddity.com/articles/natural-language-processing/>
<https://aiml.com/what-are-the-advantages-and-disadvantages-of-bag-of-words-model/>
<https://ayselaydin.medium.com/11-word2vec-approaches-word-embedding-in-nlp-538478c14b37>
<https://www.youtube.com/watch?app=desktop&v=UqRCEmrVlgQ>
<https://spotintelligence.com/2023/07/27/continuous-bag-of-words/>
https://fr.ryte.com/wiki/TF*IDF#:~:text=La%20formule%20TF*IDF%20permet,rapport%20au%20reste%20du%20texte
<https://datascientest.com/tf-idf-intelligence-artificielle>
<https://nlp-ensae.github.io/files/2-ML-FOR-NLP-2022.pdf>
<https://www.geeksforgeeks.org/continuous-bag-of-words-cbow-in-nlp/>
<https://www.ibm.com/think/topics/word-embeddings>
<https://www.intelligence-artificielle-school.com/ecolet/technologies/abert-modele-nlp/>
<https://lesdieuxducode.com/blog/2019/4/bert--le-transformer-model-qui-sentraine-et-qui-represente>
<https://www.intelligence-artificielle-school.com/ecolet/technologies/abert-modele-nlp/>
<https://fastercapital.com/fr/contenu/BERT-Comprendre-BERT--un-guide-complet.html>
https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fb053c1c4a845aa-Paper.pdf
<https://arxiv.org/pdf/1706.03762>
<https://blent.ai/blog/a/transformers-deep-learning>
<https://fr.shaip.com/blog/the-a-to-z-of-data-annotation/>
<https://www.peopleforai.com/fr/comment-maintenir-la-qualite-en-annotation-de-donnees/>
<https://konfuzio.com/fr/nlp-tools/>