REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Larbi Ben M'Hidi Oum El Bouaghi

# Cours Traitement du langage naturel

## Chapitre 3
## Représentation du texte

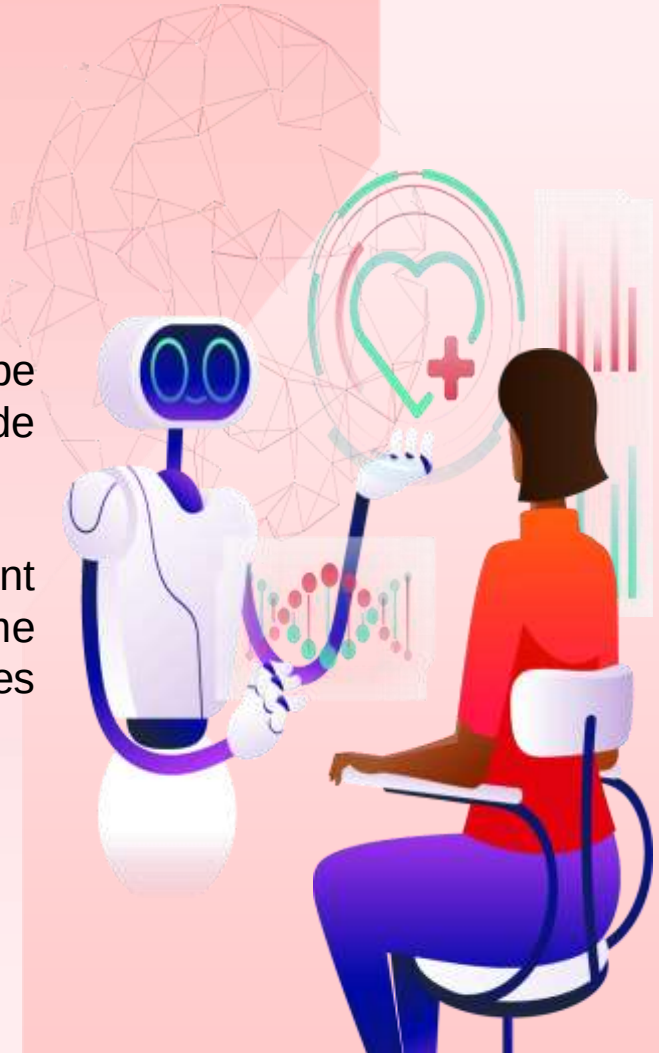
Dr.Belhouchette K

# Contenu du troixiéme chapitre

# Introduction

- la **représentation du texte** constitue en TLN une étape fondamentale pour permettre aux ordinateurs de comprendre et de manipuler des données textuelles.

- Les ordinateurs, étant incapables de traiter directement du texte brut, ont besoin de le convertir en une forme numérique qui pourra être exploitée par des modèles d'apprentissage automatique.



# Pourquoi la représentation du texte est-elle cruciale ?

- Les tâches de TLN, telles que la **classification de texte**, l'**analyse de sentiments**, ou encore la **traduction automatique**, nécessitent toutes une forme de représentation numérique du texte.

- Sans cela, il serait impossible de comparer, d'analyser ou d'extraire des informations significatives à partir de documents textuels.

- La **représentation du texte** permet donc de capturer les caractéristiques essentielles du contenu tout en facilitant l'interprétation par les algorithmes de machine learning.

# Les principales techniques de représentation du texte

techniques de représentation du texte

- Le sac de mots (BoW)
- TF-IDF
- Word embeddings

# Les principales techniques de représentation du texte

techniques de représentation du texte

- Le sac de mots (BoW)
- TF-IDF
- Word embeddings

# Les principales techniques de représentation du texte

## 1. Le sac de mots (BoW)

- Le modèle **Bag of Words (BoW)** est une méthode simple mais puissante pour la représentation des documents textuels en vue de traitement automatique.
- Il repose sur la création d'un vecteur numérique représentant un texte en se basant uniquement sur la fréquence des mots qu'il contient.
- Ce modèle ignore l'ordre des mots et les relations syntaxiques entre eux.

# Les principales techniques de représentation du texte

## 1. Le sac de mots (BoW) : Principe de base

- Transformer chaque document en un vecteur de caractéristiques, où chaque dimension représente un mot unique du vocabulaire global du corpus.
- La valeur de chaque dimension du vecteur correspond à la fréquence d'apparition de ce mot dans le document concerné.
- Ce modèle est principalement utilisé pour des tâches telles que la classification de texte ou la recherche d'information.

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


- Cette étape consiste à diviser le texte brut en unités de base appelées tokens.
- Ces tokens peuvent être des mots individuels ou des séquences de mots adjacents, appelées n-grams.
- Un n-gram est une sous-séquence de n mots consécutifs extraits du texte.

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


**Phrase** : Les étudiants étudient la programmation informatique.

**Tokenisation (en mots individuels)** :

Tokens : ["Les", "étudiants", "étudient", "la", "programmation", "informatique"]

**Bi-grams** : ["Les étudiants", "étudiants étudient", "étudient la", "la programmation", "programmation informatique"]

**Trigrams** (n-grams de taille 3) :
Trigrams : ["Les étudiants étudient", "étudiants étudient la", "étudient la programmation", "la programmation informatique"]

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


Après la **tokenisation** des textes, l'étape suivante est la **création du vocabulaire**

Le vocabulaire représente l'ensemble des mots distincts extraits du corpus de documents

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


**Tokenisation :** Découper les documents en tokens (mots ou n-grams).
**Extraction des mots distincts :** Identifier tous les mots uniques à travers l'ensemble des documents.
**Création du vocabulaire** : Construire une liste des mots uniques présents dans le corpus.
Chaque mot du vocabulaire devient une **dimension** dans le vecteur de caractéristiques.

Exemple : Si le vocabulaire contient 5 mots distincts, chaque document sera représenté par un vecteur de taille 5, où chaque dimension correspond à la présence (ou à la fréquence) d'un mot spécifique.

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


- La taille du vocabulaire dépend du corpus et de sa diversité.
- Elle influence directement :**La dimensionnalité** des vecteurs de caractéristiques et Les **besoins en mémoire** et la **complexité des calculs**.
- Dans des corpus volumineux, le vocabulaire peut contenir des centaines de milliers de mots, rendant la représentation plus complexe.

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


Texte 1 : "Les pommes sont rouges et sucrées" Texte 2 : "Les oranges sont oranges et sucrées"

1. Tokenisation :

Texte 1 : ["Les", "pommes", "sont", "rouges", "et", "sucrées"]

Texte 2 : ["Les", "oranges", "sont", "oranges", "et", "sucrées"]

2. Extraction des mots distincts :

Mots distincts : ["Les", "pommes", "sont", "rouges", "et", "sucrées", "oranges"]

3. Construction du vocabulaire :

Vocabulaire : ["Les", "pommes", "sont", "rouges", "et", "sucrées", "oranges"]

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


Une fois le vocabulaire établi, chaque document est converti en un vecteur où chaque dimension correspond à un mot du vocabulaire.

La valeur de chaque dimension est la fréquence d'apparition du mot correspondant dans le document.

Par exemple, pour un vocabulaire de 3 mots : ["chat", "chien", "manger"], un document contenant "chat" et "manger" sera représenté par [1, 0, 1] (1 fois "chat", 0 fois "chien", 1 fois "manger").

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


Texte 1 : "Les pommes sont rouges et sucrées" Texte 2 : "Les oranges sont oranges et sucrées"

1. Tokenisation :

Texte 1 : ["Les", "pommes", "sont", "rouges", "et", "sucrées"]

Texte 2 : ["Les", "oranges", "sont", "oranges", "et", "sucrées"]

2. Extraction des mots distincts :

Mots distincts : ["Les", "pommes", "sont", "rouges", "et", "sucrées", "oranges"]

3. Construction du vocabulaire :

Vocabulaire : ["Les", "pommes", "sont", "rouges", "et", "sucrées", "oranges"]

4. Représentation des documents :

Vecteur du Texte 1 : ["Les", "pommes", "sont", "rouges", "et", "sucrées"] → [1, 1, 1, 1, 1, 1, 0]

Vecteur du Texte 2 : ["Les", "oranges", "sont", "oranges", "et", "sucrées"] → [1, 0, 1, 0, 1, 1, 1]

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :


- Après avoir vectorisé tous les documents, on obtient une **matrice de caractéristiques** où chaque ligne représente un document et chaque colonne correspond à un mot du vocabulaire.

- Les valeurs dans cette matrice sont les fréquences des mots dans les documents
- Chaque ligne représente un texte et chaque colonne représente un mot du vocabulaire
- Les éléments de la matrice sont les valeurs des vecteurs (0 ou 1), indiquant la présence ou l'absence d'un mot dans chaque texte

- La matrice est souvent creuse, car la plupart des documents ne contiennent qu'un sous-ensemble du vocabulaire total.

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :

Matrice des caractéristiques

| Vocabulaire | Les | Pommes | Sont | rouges | Et | Sucrées | oranges |
|---|---|---|---|---|---|---|---|
| Texte 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Texte 2 | 1 | 0 | 1 | 0 | 1 | 1 | 2 |

# Les principales techniques de représentation du texte

## 1. Étapes de la création du modèle sac de mots :

```
from sklearn.feature_extraction.text import CountVectorizer
```

**CountVectorizer**: Classe de la bibliothèque sklearn qui permet de transformer un texte en une matrice de fréquence des mots (Bag of Words).

# Les principales techniques de représentation du texte

techniques de représentation du texte

- Le sac de mots (BoW)
- **TF-IDF**
- Word embeddings

# Les principales techniques de représentation du texte

## Le modèle TF-IDF :

- Le modèle TF-IDF (Term Frequency - Inverse Document Frequency) est une méthode largement utilisée pour représenter les documents textuels de manière numérique, dans le cadre du traitement du langage naturel.

- Il est particulièrement utile pour évaluer l'importance des mots dans un corpus de textes, en tenant compte à la fois de leur fréquence dans un document donné et de leur rareté dans l'ensemble du corpus.

# Les principales techniques de représentation du texte

## Composantes du modèle TF-IDF :

**1. Term Frequency (TF)** : Cette mesure quantifie la fréquence d'apparition d'un mot dans un document.

L'idée est qu'un mot est plus important s'il apparaît fréquemment dans le document.

La formule de TF pour un terme t dans un document d est la suivante :

$$\mathrm{TF}(t,d) = \frac{\text{Nombre d'occurrences de } t \text{ dans } d}{\text{Nombre total de mots dans } d}$$

Cela permet de calculer la proportion de la fréquence du mot par rapport à la taille du document.

# Les principales techniques de représentation du texte

## Composantes du modèle TF-IDF :

**2. Inverse Document Frequency (IDF)** : Cette mesure évalue l'importance d'un mot dans l'ensemble du corpus.

Un mot qui apparaît dans de nombreux documents est considéré comme moins informatif.

La formule de l'IDF pour un terme t est :

$$\mathrm{IDF}(t) = \log\left(\frac{N}{\text{Nombre de documents contenant le terme } t}\right)$$

Où N est le nombre total de documents dans le corpus. Si un terme apparaît dans tous les documents, son IDF sera faible, ce qui indique qu'il n'apporte pas beaucoup d'information.

# Les principales techniques de représentation du texte

## Composantes du modèle TF-IDF :

**3. TF-IDF** : La combinaison de ces deux mesures donne le score TF-IDF d'un terme dans un document.

La formule est la suivante : $$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

- Ce score représente l'importance d'un terme t dans un document d, en tenant compte de la fréquence du terme dans le document et de sa rareté dans l'ensemble du corpus.
- Plus un terme apparaît fréquemment dans un document et moins il est courant dans le reste du corpus, plus son score TF-IDF sera élevé.

# Les principales techniques de représentation du texte

## Utilisation du modèle TF-IDF :

**Filtrage de mots-clés** : TF-IDF est utilisé pour extraire les mots-clés importants d'un texte, en supprimant les termes trop fréquents (comme les mots de fonction) et en mettant en avant ceux qui sont spécifiques au contenu.

# Les principales techniques de représentation du texte

**Utilisation du modèle TF-IDF :**

**Représentation de documents** : Dans la modélisation de texte, chaque document peut être représenté comme un vecteur dans un espace de caractéristiques, où chaque dimension correspond à un terme du vocabulaire et la valeur de la dimension est le score TF-IDF du terme pour ce document.

# Les principales techniques de représentation du texte

**Utilisation du modèle TF-IDF :**

**Recherche d'informations** : TF-IDF est couramment utilisé dans les moteurs de recherche pour évaluer la pertinence des documents par rapport à une requête.

# Les principales techniques de représentation du texte

**Exemple illustratif du modèle TF-IDF :**

Supposons un corpus de trois articles scientifiques :

**Document 1** : "Les réseaux de neurones sont utilisés pour la reconnaissance d'images."
**Document 2** : "Les algorithmes génétiques optimisent les solutions pour les problèmes complexes."
**Document 3** : "La reconnaissance d'images repose sur les réseaux convolutifs et d'autres modèles avancés."

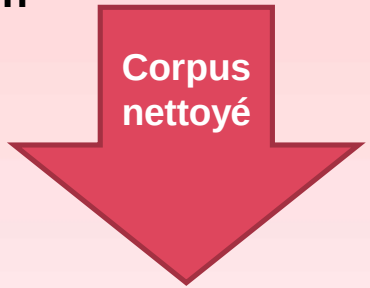
Nous allons utiliser **TF-IDF** pour extraire les mots-clés pertinents de ces documents.

# Les principales techniques de représentation du texte

## Exemple illustratif du modèle TF-IDF :

### Étape 1 : Prétraitement

1.Convertir en **minuscules**
2.Supprimer les **mots de fonction**: "les", "sont", "pour"
3.Supprimer la **ponctuation**




**Document 1** : "réseaux neurones reconnaissance images"
**Document 2** : "algorithmes génétiques optimisent solutions problèmes complexes"
**Document 3** : "reconnaissance images réseaux convolutifs modèles avancés"

# Les principales techniques de représentation du texte

**Exemple illustratif du modèle TF-IDF :**

**Étape 2 : Calcul du TF (Term Frequency)**

$$\mathrm{TF}(t,d) = \frac{\text{Fréquence de } t \text{ dans } d}{\text{Nombre total de mots dans } d}$$

# Étape 2 : Calcul du TF (Term Frequency)

| Mot | TF (Doc 1) | TF (Doc 2) | TF (Doc 3) |
|---|---|---|---|
| **Réseaux** | 1⁄4=0.25 | 0 | 1 / 6≈0.166 |
| **neurones** | 1⁄4=0.25 | 0 | |
| **reconnaissance** | 1⁄4=0.25 | 0 | 1 / 6≈0.166 |
| **images** | 1⁄4=0.25 | 0 | 1 / 6≈0.166 |
| **algorithmes** | 0 | 1 / 6≈0.166 | |
| **génétiques** | 0 | 1 / 6≈0.166 | |
| **optimisent** | 0 | 1 / 6≈0.166 | |
| **solutions** | 0 | 1 / 6≈0.166 | |
| **problèmes** | 0 | 1 / 6≈0.166 | |
| **complexes** | 0 | 1 / 6≈0.166 | |
| **convolutifs** | 0 | 0 | 1 / 6≈0.166 |
| **modèles** | 0 | 0 | 1 / 6≈0.166 |
| **avancés** | 0 | 0 | 1 / 6≈0.166 |

# Les principales techniques de représentation du texte

## Exemple illustratif du modèle TF-IDF :

### Étape 3 : Calcul de l'IDF (Inverse Document Frequency)

$$\mathrm{IDF}(t) = \log\left(\frac{N}{d}\right)$$

Avec :

N=3 est le nombre total des documents,
d est le nombre de documents contenant le mot

# Étape 3 : Calcul de l'IDF (Inverse Document Frequency)

| Mot | Nombre de documents contaenat le mot(d) | IDF (log(3/d) |
|---|---|---|
| **Réseaux** | 2 | log(3/2)≈0.176 |
| **neurones** | 1 | log(3/1)≈0.477 |
| **reconnaissance** | 2 | log(3/2)≈0.176 |
| **images** | 2 | log(3/2)≈0.176 |
| **algorithmes** | 1 | log(3/1)≈0.477 |
| **génétiques** | 1 | log(3/1)≈0.477 |
| **optimisent** | 1 | log(3/1)≈0.477 |
| **solutions** | 1 | log(3/1)≈0.477 |
| **problèmes** | 1 | log(3/1)≈0.477 |
| **complexes** | 1 | log(3/1)≈0.477 |
| **convolutifs** | 1 | log(3/1)≈0.477 |
| **modèles** | 1 | log(3/1)≈0.477 |

# Les principales techniques de représentation du texte

**Exemple illustratif du modèle TF-IDF :**

**Étape 4 : Calcul du TF-IDF**

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

# Étape 4 : Calcul du TF-IDF

| Mot | TF-IDF (Doc 1) | TF-IDF (Doc 2) | TF-IDF (Doc 3) |
|---|---|---|---|
| **Réseaux** | 0.25×0.176=0.044 | 0 | 0.166×0.176=0.029 |
| **neurones** | 0.25×0.477=0.119 | 0 | 0 |
| **reconnaissance** | 0.25×0.176=0.044 | 0 | 0.166×0.176=0.029 |
| **images** | 0.25×0.176=0.044 | 0 | 0.166×0.176=0.029 |
| **algorithmes** | 0 | 0.166×0.477≈0.079 | 0 |
| **génétiques** | 0 | 0.166×0.477≈0.079 | 0 |
| **optimisent** | 0 | 0.166×0.477≈0.079 | 0 |
| **solutions** | 0 | 0.166×0.477≈0.079 | 0 |
| **problèmes** | 0 | 0.166×0.477≈0.079 | 0 |
| **complexes** | 0 | 0.166×0.477≈0.079 | 0 |
| **convolutifs** | 0 | 0 | 0.166×0.477≈0.079 |
| **modèles** | 0 | 0 | 0.166×0.477≈0.079 |
| **avancés** | 0 | 0 | 0.166×0.477≈0.079 |

# Les principales techniques de représentation du texte

## Le modèle TF-IDF :

- Un mot avec une **TF-IDF élevée** est à la fois :Fréquent dans le document analysé **TF élevé**
- Rare dans les autres documents **IDF élevé**

# Les principales techniques de représentation du texte

techniques de représentation du texte

- Le sac de mots (BoW)
- TF-IDF
- **Word embeddings**

# Les principales techniques de représentation du texte

## Word embeddings

Contrairement aux méthodes classiques comme **Bag of Words (BoW)** ou **TF-IDF**, qui utilisent des représentations discrètes et ne capturent pas le sens des mots, les **word embeddings** apprennent des représentations sémantiques basées sur le contexte des mots dans de grands corpus de textes.

# Les principales techniques de représentation du texte

## Word embeddings

- Les **word embeddings** reposent sur l'idée que les mots ayant des significations similaires apparaissent souvent dans des contextes similaires.

- Ils permettent d'effectuer des calculs vectoriels pour capturer des relations sémantiques et syntaxiques entre les mots.

# Les principales techniques de représentation du texte

## Word embeddings

**Exemple Comparaison entre fruits et animaux**

Dans un espace vectoriel à 3 dimensions généré par un modèle d'embedding,

voici comment les mots "pomme", "orange", "chien" et "chat" pourraient être représentés pour illustrer leur proximité sémantique :

**"pomme"** : [0.9, 0.2, 0.1]
**"orange"** : [0.8, 0.3, 0.0]
**"chien"** : [0.1, 0.7, 0.8]
**"chat"** : [0.2, 0.8, 0.9]

# Les principales techniques de représentation du texte

## Word embeddings

**Analyse :**

1.Les vecteurs de **"pomme"** et **"orange"** sont proches, car ils partagent une sémantique similaire (des fruits)

2.Les vecteurs de **"chien"** et **"chat"** sont également proches, car ils représentent des animaux domestiques.

3.Les distances entre un fruit "pomme«  et un animal "chien«  sont plus grandes, car ils n’ont pas de lien sémantique direct.

Pour mesurer ces relations, une métrique comme la **distance cosinus** peut être utilisée

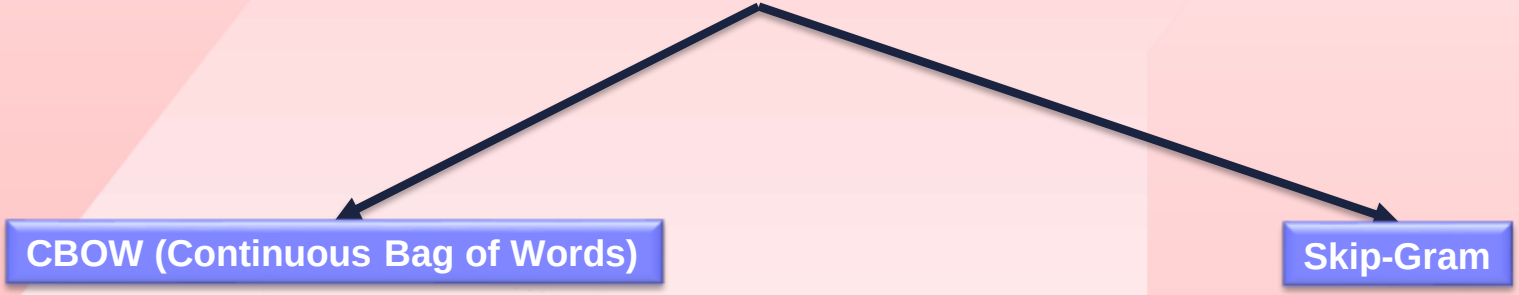
# Les principales techniques de représentation du texte

## Techniques principales du Word embeddings

### 1. Word2Vec

- Word2Vec est une méthode populaire développée par Mikolov et al. (2013) pour apprendre des représentations vectorielles des mots.

- Word2Vec repose sur un **réseau de neurones peu profond** et utilise l'une des deux architectures suivantes :



# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

**CBOW (Continuous Bag of Words)**

Dans cette approche, l'objectif est de prédit un mot en fonction de son contexte.

**Exemple** : Imaginons que nous analysons des rapports financiers et que nous avons une phrase comme :
« L'entreprise a annoncé une augmentation de ses revenus au dernier trimestre »

Ici, le mot cible pourrait être "**augmentation**", et le modèle utiliserait les mots [L'entreprise, a, une, de, ses, revenus, au, dernier, trimestre] (en excluant parfois les mots vides) pour prédire ce mot.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

CBOW (Continuous Bag of Words)

**Comment cela fonctionne ?**

1.Le modèle prend les mots environnants ("L'entreprise", "revenus", "trimestre")

2.Il essaie d'inférer que le mot cible central probable est **« augmentation »**

3.Il ajuste ses poids internes pour mieux prédire ce mot à l’avenir.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

**CBOW (Continuous Bag of Words)**

- Cela fait de CBOW une approche plus rapide et adaptée aux corpus de taille plus petite.
- Elle tend également à créer des représentations plus générales des mots fréquents.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

**CBOW (Continuous Bag of Words)**

1. **Prétraitement du texte** :

   - Tokenisation du texte
   - Filtrage des mots peu fréquents (en utilisant un seuil de fréquence minimale)
   - Création d'un vocabulaire et de mappings entre les mots et leurs indices

2. **Génération des données d'entraînement** :
   - Pour chaque mot du texte, on extrait son contexte : les mots dans une fenêtre autour de lui
   - On crée des paires (contexte, mot cible) pour l'entraînement

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

CBOW (Continuous Bag of Words)

3. **Propagation avant (forward pass)** :

- On prend les mots contextuels comme entrée. Par exemple, si le contexte est un ensemble de mots autour du mot cible, on passe ces mots à travers un **embedding layer** pour obtenir leurs représentations vectorielles.

- Calcul des embeddings du contexte : On fait la moyenne des vecteurs d'embedding des mots contextuels

- Prédiction : On utilise cette moyenne pour prédire la probabilité des mots cibles. Dans CBOW, on essaie de prédire un mot à partir de son contexte en utilisant un produit scalaire entre la représentation du contexte et les poids des embeddings du mot cible.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

CBOW (Continuous Bag of Words)

**4. Rétropropagation (backpropagation)** :

- La rétropropagation est la méthode utilisée pour ajuster les poids du modèle afin de minimiser l'erreur, ou la fonction de perte, obtenue lors de la propagation avant

**5. Entraînement itératif** :

- On répète les étapes 3 et 4 pour toutes les paires (contexte, mot cible)
- On effectue plusieurs époques sur l'ensemble des données
- On réduit progressivement le taux d'apprentissage

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

CBOW (Continuous Bag of Words)

**6. Utilisation du modèle entraîné** :

- Les embeddings de mots sont stockés
- On peut calculer la similarité entre mots en utilisant la similarité cosinus entre leurs vecteurs
- On peut visualiser les relations entre mots dans un espace vectoriel réduit

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

**CBOW (Continuous Bag of Words)**

D: > TLN > python > CBOW > corpus

```
1  from gensim.models import Word2Vec
2
3  corpus = [["je", "suis", "fan", "de", "nlp"],
4            ["word2vec", "est", "puissant"],
5            ["cbow", "est", "un", "modele", "simple"]]
6  model = Word2Vec(sentences=corpus, vector_size=50, window=5, min_count=1, workers=4, sg=0)
7  print(model.wv["nlp"])
8
```

Entraîner Word2Vec avec CBOW (sg=0 signifie CBOW, sg=1 signifie Skip-gram)

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

Skip-Gram

- L'approche Skip-Gram est l'inverse de CBOW
- Au lieu de prédire le mot cible à partir du contexte, elle prédit les mots de contexte à partir d'un mot central.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

Skip-Gram

Exemple : Analyse financière d'entreprise

Prenons la phrase suivante issue d'un rapport annuel :

« L'entreprise a enregistré une forte croissance de son chiffre d'affaires ce trimestre »

Si nous utilisons Skip-Gram, nous allons choisir un mot central, par exemple "croissance", et essayer de prédire ses mots voisins potentiels :

- "forte"
- "enregistré"
- "chiffre d'affaires"
- "trimestre"

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

Skip-Gram

- Skip-Gram fonctionne mieux pour les mots rares, car il a tendance à mieux capturer des représentations spécifiques aux mots peu fréquents, en particulier dans les grands corpus.

- Elle peut aussi être plus lente à entraîner que CBOW à cause de sa nature d'apprentissage basée sur de nombreux exemples.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

Skip-Gram

```
D: > TLN > python > SKIP GRAM > ...
1  from gensim.models import Word2Vec
2
3  corpus = [["je", "suis", "fan", "de", "nlp"],
4            ["word2vec", "est", "puissant"],
5            ["cbow", "est", "un", "modele", "simple"]]
6  model = Word2Vec(sentences=corpus, vector_size=50, window=5, min_count=1, workers=4, sg=1)
7  print(model.wv["nlp"])
```

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 1. Word2Vec

**Propriétés**

- **Entraînement** : Les modèles Word2Vec utilisent une fonction d'activation de type sigmoïde et un algorithme d'optimisation comme stochastic gradient descent (SGD) pour minimiser une fonction de coût comme l'erreur de prédiction de contexte
- **Dimensionnalité** : Les embeddings générés par Word2Vec sont souvent de faible dimension, comme 100 à 300 dimensions.
- **Limite** : Word2Vec ignore l'ordre des mots dans le contexte, ce qui peut limiter sa capacité à capturer des relations plus complexes entre les mots.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. GloVe (Global Vectors for Word Representation)

- GloVe a été développé par Stanford en 2014 pour améliorer les approches comme Word2Vec en utilisant les statistiques globales du corpus.

- Il s'appuie sur des réseaux de neurones pour apprendre les relations entre mots, GloVe est basé sur la **factorisation de matrices** et exploite les cooccurrences globales des mots dans un corpus.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. GloVe (Global Vectors for Word Representation)

POURQUOI GLOVE ?

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. GloVe (Global Vectors for Word Representation)

- L’idée principale derrière **GloVe** est de capturer l'information statistique du corpus de manière globale.

- Les auteurs de GloVe (Pennington et al., 2014) ont remarqué qu'en combinant les co-occurrences de mots dans un grand corpus de texte, il est possible d’apprendre une bonne représentation des mots.

- GloVe est donc basé sur l'idée que les relations entre les mots peuvent être déduites de leurs co-occurrences dans le corpus, et que ces co-occurrences reflètent la sémantique.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

matrice de co-occurrence

- GloVe commence par créer une **matrice de co-occurrence** à partir d'un grand corpus de texte.

- Chaque élément de cette matrice représente la fréquence avec laquelle deux mots apparaissent ensemble dans une certaine fenêtre de contexte.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

### matrice de co-occurrence

Prenant l'exemple : chien mange une pomme

Alors, la matrice de co-occurrence sera construite en comptabilisant les occurrences de chaque mot avec chaque autre mot.

Pour ce corpus, nous pourrions obtenir une partie de la matrice comme suit

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

matrice de co-occurrence

| | chien | Mange | Une | pomme |
|---|---|---|---|---|
| **chien** | 0 | 1 | 1 | 0 |
| **mange** | 1 | 0 | 1 | 0 |
| **une** | 1 | 1 | 0 | 1 |
| **pomme** | 0 | 0 | 1 | 0 |

Les éléments de cette matrice représentent combien de fois deux mots apparaissent dans la même fenêtre de contexte.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

Factorisation de la Matrice de Co-occurrence

- Une fois que cette matrice de co-occurrence est construite, GloVe procède à une **factorisation** de cette matrice pour obtenir des vecteurs de mots.

- L'objectif est de trouver une factorisation de la matrice X qui minimise une fonction de coût.

- Mathématiquement, pour chaque paire de mots (i,j), GloVe cherche des vecteurs vi et vj tels que :

$$X_{ij} \approx v_i^T v_j$$

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

Grâce à GloVe, nous pouvons regrouper les termes générer par la matrice pour et améliorer :

- Les moteurs de recherche: suggestions intelligentes
- L'analyse de sentiment : détection de termes positifs/négatif
- Les recommandations documentaires : mots proches sémantiquement

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

D: > TLN > python > glov > ...

```python
from glove import Corpus, Glove
corpus = [["je", "suis", "fan", "de", "nlp"],
          ["word2vec", "est", "puissant"],
          ["glove", "est", "basé", "sur", "les", "cooccurrences"]]
corpus_model = Corpus()
corpus_model.fit(corpus, window=5)
glove = Glove(no_components=50, learning_rate=0.05)
glove.fit(corpus_model.matrix, epochs=20, no_threads=4, verbose=True)
glove.add_dictionary(corpus_model.dictionary)
print(glove.word_vectors[glove.dictionary["nlp"]])

```

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

`corpus_model.fit(corpus, window=5)` : construit la matrice de co-occurrence en utilisant une fenêtre de taille 5 autour de chaque mot

Cela crée une matrice de co-occurrence X où :

X[i, j] = combien de fois le mot i apparaît avec le mot j dans la fenêtre donnée.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

`Glove(no_components=50, learning_rate=0.05)` crée un modèle GloVe avec :

`no_components=50` : La dimension des embeddings (chaque mot sera représenté par un vecteur de taille 50)

`learning_rate=0.05` : Le taux d'apprentissage utilisé pour ajuster les vecteurs lors de l'entraînement.

`glove.fit()` applique une factorisation de matrice sur la matrice de co-occurrence créée avec `corpus_model.fit()`

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 2. Etape de fonctionnement de GloVe

`no_threads (int):` Nombre de threads utilisés pour l'entraînement

`verbose (bool):` Affiche des messages pendant l'entraînement si True, sinon reste silencieux

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 3. FastText

- FastText, développé par Facebook AI, est une extension de Word2Vec qui améliore la représentation des mots en utilisant des sous-mots (subwords).

- Contrairement à Word2Vec (CBOW & Skip-Gram) et GloVe, FastText ne représente pas un mot comme une seule unité, mais comme une somme de plusieurs morceaux de mots (n-grammes).

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 3. FastText

- Par exemple, dans les mots suivants :**écran** et **tablette**
- FastText décompose ces mots en sous-mots (n-grams) tels que "**écr**", "**cran**", "**tab**", "**blet**", "**ette**".
- Ce qui permet de mieux gérer les mots rares ou inconnus, comme "**smartphone**" qui pourrait être décomposé en "**smar**", "**mart**", "**phon**", "**hone**".
- Si un mot n'a jamais été vu dans le corpus d'entraînement, FastText pourra quand même générer un vecteur de représentation basé sur les sous-mots.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 3. FastText

D: > TLN > python > fastText > ...

```python
from gensim.models import FastText
corpus = [["je", "suis", "fan", "de", "nlp"],
          ["fasttext", "est", "très", "puissant"],
          ["il", "gère", "les", "mots", "rares"]]

model = FastText(sentences=corpus, vector_size=50, window=5, min_count=1, workers=4)

print(model.wv["nlp"])
print(model.wv.most_similar("nlp"))

```

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 3. FastText

```
Entraîner FastText
model = FastText(sentences=corpus
```

`vector_size=50:` chaque sous-mot FastText sera représenté par un vecteur de taille **50**.

`window=5:` la taille de la fenêtre de contexte autour du mot cible

`min_count=1:` le nombre minimum d'occurrences qu'un mot doit avoir dans le corpus pour être pris en compte

`workers=4:` le nombre de threads (ou processus parallèles) à utiliser pour accélérer l'entraînement.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 3. FastText

**Pourquoi FastText est-il plus puissant ?**

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 3. FastText

FastText est conçu pour résoudre deux problèmes majeurs de Word2Vec et GloVe :

1.Gestion des mots rares ou inconnus : si un mot n’a jamais été vu dans le corpus d’entraînement, Word2Vec et GloVe ne peuvent pas lui attribuer un vecteur. FastText, en revanche, génère une représentation en se basant sur les sous-mots.

2.Meilleure capture de la morphologie des mots

Exemple : Les mots "entreprise" et "entrepreneur" ont une racine commune "entrepr-«

FastText comprend que ces mots sont liés grâce aux n-grammes.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

Prenons le mot "banc" :

- Dans *"Je suis assis sur un banc"*, il signifie un siège.
- Dans *"Le banc de poissons"*, il désigne un groupe de poissons.
- Dans *"Un banc d'essai"*, il signifie un dispositif de test.

Word2Vec, GloVe et FastText donneront le même vecteur à "banc" dans toutes ces phrases.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

**Donc c’est quoi la solution?**

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

# Les approches contextuelles

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles

Les méthodes classiques de Word Embedding comme Word2Vec, GloVe et FastText ont une limitation majeure **: elles attribuent un seul vecteur par mot, sans tenir compte du contexte.**

**Cela pose problème pour les mots polyvalents (polysemous)**

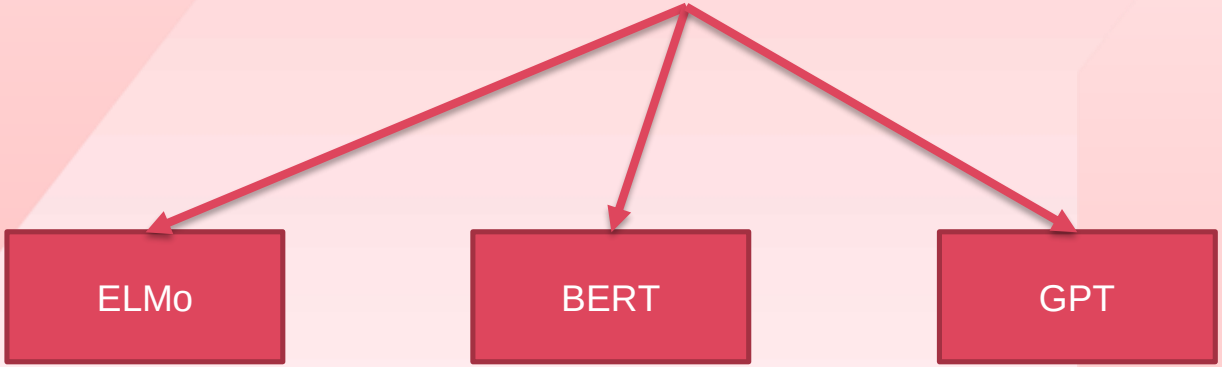
# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

Les approches contextuelles génèrent des embeddings qui varient en fonction du contexte dans lequel un mot apparaît.

Le but est de différencier des mots ayant des significations multiples.



# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

**ELMo (Embeddings from Language Models**

- Est un modèle d'embeddings de mots basé sur des réseaux neuronaux profonds, développé par AllenNLP en 2018

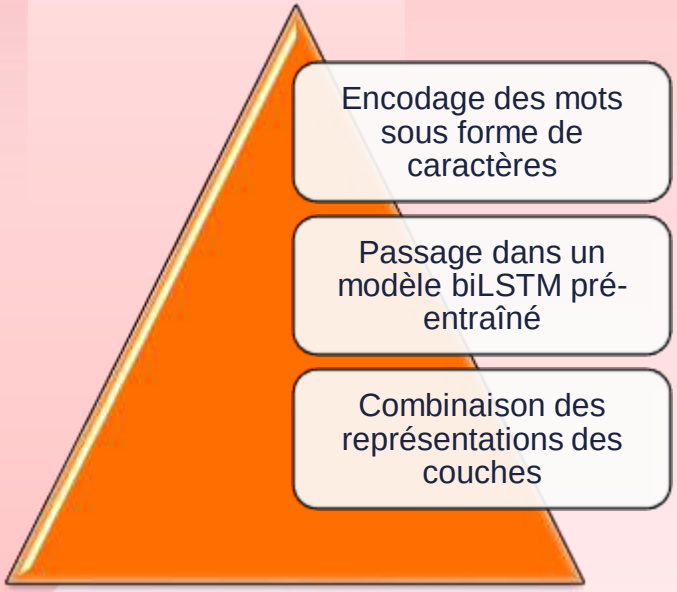
- ELMo repose sur un modèle **bidirectionnel de LSTM** (biLSTM) entraîné sur une tâche de modélisation du langage (Language Model). Il fonctionne en trois grandes étapes

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

ELMo (Embeddings from Language Models




**Embedding layer basée sur les caractères**: permet de capturer des **informations morphologiques**

Etapes de ELMo

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 4. Approches contextuelles (BERT, ELMo, GPT)

ELMo (Embeddings from Language Models

2. Passage dans un modèle biLSTM pré-entraîné

- LSTM (Long Short-Term Memory) est un type de réseau de neurones récurrent (RNN) qui est largement utilisé pour traiter des données séquentielles, comme du texte, de la parole ou des séries temporelles

- Il est capable de retenir des informations importantes pendant de longues périodes, ce qui le rend adapté pour des tâches où le contexte sur des distances longues est important, comme dans le traitement du langage naturel

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 4. Approches contextuelles (BERT, ELMo, GPT)

ELMo (Embeddings from Language Models

2. Passage dans un modèle biLSTM pré-entraîné

- Un réseau LSTM bidirectionnel empilé (stacked biLSTM) analyse la séquence de texte
- Le premier LSTM lit le texte de gauche à droite : modélisation du langage directe
- Le second LSTM lit le texte de droite à gauche : modélisation du langage inversée

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 4. Approches contextuelles (BERT, ELMo, GPT)

ELMo (Embeddings from Language Models

### 3. Combinaison des représentations des couches

- Chaque mot obtient un embedding de 3 niveaux :

1. Un premier niveau = embedding de base (caractères)
2. Un second niveau = sortie du 1er LSTM (gauche → droite)
3. Un troisième niveau = sortie du 2e LSTM (droite → gauche)

- Une combinaison pondérée de ces couches donne l'embedding final contextuel.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 4. Approches contextuelles (BERT, ELMo, GPT)

ELMo (Embeddings from Language Models

Soient les mots **banc** dans les deux phrases suivantes :

- Le **banc** du parc est en bois
- Il a trouvé un **banc** de poissons dans la rivière

- Dans le premier contexte, **banc** fait référence à un objet sur lequel on peut s'asseoir.

- Dans le second, il fait référence à un groupe de poissons.

- **ELMo** génère des vecteurs différents pour **banc** dans ces deux contextes, car il tient compte de l'ensemble de la phrase.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 4. Approches contextuelles (BERT, ELMo, GPT)

ELMo (Embeddings from Language Models

D: > TLN > python > 11 > ...

```python
import allennlp
print(allennlp.__version__)

from allennlp.commands.elmo import ElmoEmbedder

elmo = ElmoEmbedder()
sentence = ["I", "love", "deep", "learning"]
embeddings = list(elmo.embed_sentences([sentence]))

# Embedding du mot "deep"
word_embedding = embeddings[0][:, 2, :]
print(word_embedding.shape)
```

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

ELMo (Embeddings from Language Models

la classe ElmoEmbedder:  permet de charger le modèle ELMo et de générer des embeddings de mots.

ElmoEmbedder: initialise le modèle pré-entraîné d'ELMo. Par défaut, il télécharge les poids du modèle ELMo de AllenNLP.

Génération des embeddings par elmo.embed_sentences([sentence]) :

- Prend une liste de phrases (ici une seule phrase)
- Retourne un tensor 3D contenant les embeddings de chaque mot
- Chaque mot est représenté par trois niveaux d'embeddings (sorties des couches ELMo)

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

4. Approches contextuelles (BERT, ELMo, GPT)

BERT (Bidirectional Encoder Representations from Transformers)

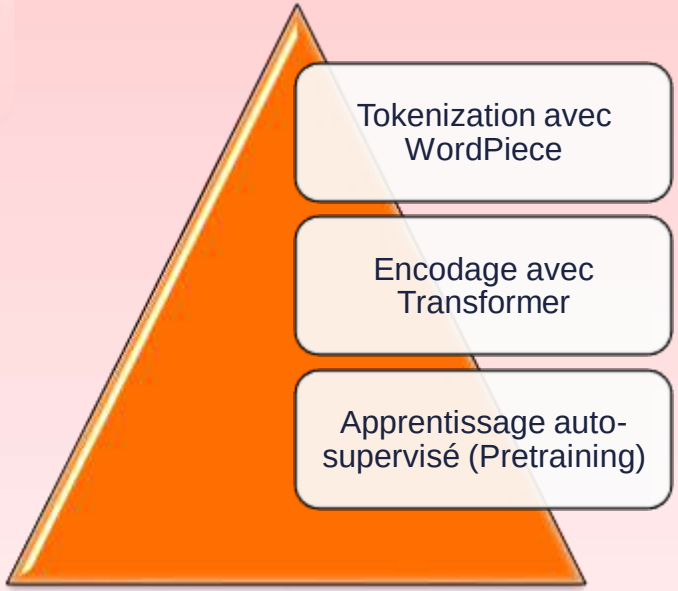
Contrairement aux modèles comme Word2Vec et ELMo, BERT est basé sur une architecture Transformer et est bidirectionnel, ce qui signifie qu'il prend en compte tout le contexte gauche et droit pour chaque mot.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

4. Approches contextuelles (BERT, ELMo, GPT)

BERT (Bidirectional Encoder Representations from Transformers)




Etapes de BERT

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 4. Approches contextuelles (BERT, ELMo, GPT)

BERT (Bidirectional Encoder Representations from Transformers)

### 1. Tokenization avec WordPiece

- Des mots sont découpés en sous-mots

playing → "play", "ing"

- Deux tokens spéciaux sont ajoutés :

[CLS] : Représente toute la phrase (utilisé pour la classification).
[SEP] : Sépare les phrases dans les tâches à deux phrases (comme la question-réponse).

# Les principales techniques de représentation du texte

**Approches principales du Word embeddings**

4. Approches contextuelles (BERT, ELMo, GPT)

2. Encodage avec Transformer

BERT (Bidirectional Encoder Representations from Transformers)

- BERT utilise **plusieurs couches de Transformers** pour traiter le texte
- Chaque mot est représenté par un **embedding contextuel**, dépendant de la phrase entière.

# Les principales techniques de représentation du texte

**Approches principales du Word embeddings**

4. Approches contextuelles (BERT, ELMo, GPT)

BERT (Bidirectional Encoder Representations from Transformers)

3. Apprentissage auto-supervisé (Pretraining)

BERT est pré-entraîné avec deux tâches :

**Masked Language Model (MLM):** Certains mots sont masqués , et le modèle doit les prédire.

Comme: "I love [MASK] learning" → BERT prédit "deep".

**Next Sentence Prediction (NSP):** Le modèle apprend si une phrase suit logiquement une autre.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

BERT (Bidirectional Encoder Representations from Transformers)

Le mot "charge" peut avoir plusieurs significations :

"Le directeur financier gère la charge des coûts fixes." → Charge = coûts
"L'ingénieur est en charge du projet de refonte." → Charge = responsabilité

✅ Avec BERT, le vecteur de "charge" sera différent selon le contexte, contrairement à Word2Vec qui lui donnerait toujours le même vecteur.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

BERT (Bidirectional Encoder Representations from Transformers)

```python
from transformers import BertTokenizer, BertModel
import torch

tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertModel.from_pretrained("bert-base-uncased")
sentence = "I love deep learning"
inputs = tokenizer(sentence, return_tensors="pt", padding=True, truncation=True)
with torch.no_grad():
    outputs = model(**inputs)

sentence_embedding = outputs.last_hidden_state[:, 0, :]
print(sentence_embedding.shape)
```

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

4. Approches contextuelles (BERT, ELMo, GPT)

BERT (Bidirectional Encoder Representations from Transformers)

`Charger le tokenizer et le modèle BERT pré-entraîné :`

bert-base-uncased : est une version de BERT avec 12 couches et 110M de paramètres.

Extraction de l'embedding de la phrase :`sentence_embedding = outputs.last_hidden_state[:, 0, :]`

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 4. Approches contextuelles (BERT, ELMo, GPT)

BERT (Bidirectional Encoder Representations from Transformers)

**Avantages de BERT**

- Comprend les contextes complexes
- Modèles pré-entraînés disponibles (BERT-base, BERT-large)
- Améliore énormément la recherche sémantique et les chatbots

**Limites**

- Plus lourd à entraîner
- Nécessite des ressources GPU importantes

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

GPT (Generative Pre-trained Transformer)

- Est une famille de modèles de traitement du langage naturel développée par OpenAI. Contrairement à BERT, qui est conçu pour la compréhension du texte, GPT est principalement utilisé pour la génération de texte

- GPT est basé sur l'architecture Transformer, mais contrairement à BERT (qui est bidirectionnel), GPT est uniquement un modèle auto-régressif (unidirectionnel).

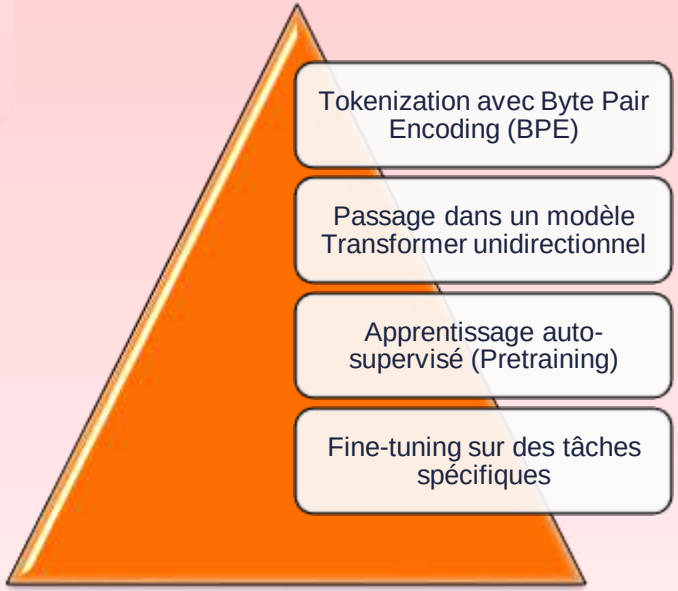
- Il prédit les mots un par un en utilisant uniquement le contexte gauche

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

GPT (Generative Pre-trained Transformer)




Les étapes clés du fonctionnement de GPT

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

## 4. Approches contextuelles (BERT, ELMo, GPT)

GPT (Generative Pre-trained Transformer)

### 1. Tokenization avec Byte Pair Encoding (BPE)

- Le texte est transformé en tokens à l'aide d'un algorithme appelé Byte Pair Encoding (BPE)

Exemple : "playing" → "play", "ing"

- Comme pour BERT, les tokens [CLS] et [SEP] ne sont pas nécessaires, car GPT ne fait pas de classification.

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

2. Passage dans un modèle Transformer unidirectionnel

GPT (Generative Pre-trained Transformer)

- Contrairement à BERT, GPT ne regarde que les tokens précédents
- Il prédit chaque mot en se basant uniquement sur le contexte gauche → droite.

# Les principales techniques de représentation du texte

**Approches principales du Word embeddings**

4. Approches contextuelles (BERT, ELMo, GPT)

3. Apprentissage auto-supervisé (Pretraining)

GPT (Generative Pre-trained Transformer)

- GPT est pré-entraîné avec une tâche simple : Language Modeling Auto-Régressif
- Prédiction du mot suivant dans une phrase

"I love deep" → Modèle prédit "learning"

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

4. Fine-tuning sur des tâches spécifiques

GPT (Generative Pre-trained Transformer)

- GPT peut être adapté pour des applications comme :La génération de texte (chatbots, rédaction assistée)
- La complétion de code
- Le résumé automatique
- La réponse aux questions

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

GPT (Generative Pre-trained Transformer)

```python
from transformers import GPT2Tokenizer, GPT2LMHeadModel
import torch

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

prompt = "Artificial intelligence is"

inputs = tokenizer(prompt, return_tensors="pt")

with torch.no_grad():
    outputs = model.generate(**inputs, max_length=50, num_return_sequences=1)
generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
print(generated_text)
```

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

### 4. Approches contextuelles (BERT, ELMo, GPT)

GPT (Generative Pre-trained Transformer)

- GPT-2 est une version populaire de GPT avec plusieurs tailles de modèles :
  - gpt2 → 117M de paramètres.
  - gpt2-medium → 345M de paramètres.
  - gpt2-large → 774M de paramètres.
  - gpt2-xl → 1,5B de paramètres.

- max_length=50 : Le texte généré sera de 50 tokens maximum
- num_return_sequences=1 : On génère une seule phrase

# Les principales techniques de représentation du texte

## Approches principales du Word embeddings

| | |
|---|---|
| **Word2Vec** | Rapide et efficace pour des représentations simples basées sur des contextes locaux. |
| **GloVe** | Capture des relations globales entre les mots dans un corpus donné |
| **FastText** | Améliore Word2Vec en intégrant des sous-mots pour mieux gérer les mots rares |
| **approches contextuelles** | Génèrent des embeddings dynamiques et sont particulièrement puissantes pour des tâches de compréhension contextuelle et de génération de texte. |

# Les principales techniques de représentation du texte

## Mesures de similarité

### Définition

Les mesures de similarité sont des outils mathématiques et informatiques utilisés pour quantifier la proximité ou la différence entre deux phrases ou chaînes de caractères

# Les principales techniques de représentation du texte

## Mesures de similarité

### Types de Mesures de Similarité

**1. Distance cosinus**

La distance cosinus mesure l'angle entre deux vecteurs dans un espace vectoriel.

Elle est basée sur le **cosinus** de l'angle entre les deux vecteurs, qui indique à quel point les deux vecteurs sont similaires.
La formule de la distance cosinus est :

$$d_{cosinus}(v1, v2) = 1 - \frac{v1 \cdot v2}{||v1|| \cdot ||v2||}$$

où :
v1·v2 est le produit scalaire des deux vecteurs.

||v1||et ||v2|| sont les normes (longueurs) des vecteurs.

Le résultat varie entre 0 et 2, où 0 signifie que les deux vecteurs sont identiques en termes de direction (très similaires), et 2 signifie qu'ils sont totalement opposés.

# Les principales techniques de représentation du texte

## Mesures de similarité

## Types de Mesures de Similarité

### 2. La distance euclidienne

La distance euclidienne mesure la "longueur" de la différence entre deux vecteurs dans un espace multidimensionnel.

Soit deux vecteurs **v1** et **v2** représentant les phrases 1 et 2, respectivement.
La formule de la distance euclidienne est :

$$d_{euclidienne}(v1, v2) = \sqrt{\sum_{i=1}^{n}(v1_i - v2_i)^2}$$

où **n** est la dimension du vecteur (nombre de caractéristiques), et $v1_i$ et $v2_i$ sont les valeurs des éléments correspondants des deux vecteurs.

# Les principales techniques de représentation du texte

## Conclusion

- Ce chapitre a présenté les principales méthodes de représentation de texte en TLN

- Nous avons vu les approches classiques comme **Bag of Words** et **TF-IDF**, puis les **word embeddings** (Word2Vec, GloVe) qui capturent des relations sémantiques. Enfin, nous avons abordé les modèles modernes tels que **ELMo** et **BERT**, qui produisent des représentations contextuelles et dynamiques.

- Ces avancées permettent une meilleure compréhension et traitement du langage naturel, ouvrant la voie à des applications de plus en plus précises et performantes.