Vector Space Model

- Besoin de convertir le texte en vecteurs numériques pour le passer aux modèles Machine Learning ……
- Le modèle d'espace vectoriel est utile pour le traitement des données textuelles, couramment utilisé en recherche d'information et en classement de documents.
- Aussi appelé modèle de vecteur terme, il représente les documents textuels sous forme de vecteurs numériques de termes spécifiques, qui constituent les dimensions des vecteurs.

Vector Space Model

- Mathématiquement, considérons un document D dans un espace vectoriel VS. Chaque document a autant de **dimensions que de termes uniques dans l'espace vectoriel.**

- L'espace vectoriel VS peut être représenté par l'ensemble des termes uniques présents dans tous les documents : VS = {W1,W2, ... ,Wn}.

- Un document D peut être représenté dans cet espace comme : D = {wD1,wD2, ... ,wDn}, où wDn est le poids du mot n dans le document D.

- Lors de l'extraction et de l'ingénierie des caractéristiques, il est crucial d'utiliser le même processus pour extraire les caractéristiques de nouveaux documents à prédire, sans reconstruire complètement l'algorithme basé sur ces nouveaux documents.

Modèles traditionnels du Feature Engineering

- Les stratégies traditionnelles pour l'ingénierie de fonctionnalités textuelles appartiennent au modèle "Sac de Mots" incluant TF, TF-IDF, N-grams, etc.
- Ces méthodes sont efficaces mais perdent des informations sémantiques, structurelles et contextuelles.
- Des modèles avancés abordent ces limitations, détaillés dans une section ultérieure.
- Ces modèles traditionnels sont basés sur des méthodes mathématiques et statistiques, appliquées à un corpus de textes.

Modèle Bag of Words

- Le modèle de l'espace vectoriel est la représentation la plus simple pour du texte non structuré.
- Il représente le texte sous forme de vecteurs numériques, où chaque dimension correspond à une caractéristique spécifique.
- Le modèle "Sac de Mots" présente chaque document texte sous forme de vecteur numérique, attribuant à chaque mot une dimension avec sa fréquence, occurrence (représentée par 1 ou 0), voire des valeurs pondérées.
- Ce modèle est nommé ainsi car chaque document est littéralement représenté comme un sac de ses propres mots, sans tenir compte de l'ordre des mots, des séquences ou de la grammaire.

Modèle Bag of Words : Exemple

- Corpus (Chaque ligne est un document séparé):

  It was the best of times,
  it was the worst of times,
  it was the age of wisdom,
  it was the age of foolishness

→ *Vocabulaire de 10 mots :* “it”, “was”, “the”, “best”, “of”, “times”, “worst”, “age”, “wisdom”, “foolishness”

→Créer les vecteurs associés aux documents :

"It was the best of times" = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

- Si un corpus de documents se compose de N mots uniques dans tous les documents → un vecteur à N dimensions pour chacun des documents.

Modèle Bag of Words

- La matrice de caractéristiques est traditionnellement représentée sous forme de matrice creuse (Sparse Matrix) en raison de l'augmentation considérable du nombre de caractéristiques avec chaque document, où chaque mot distinct devient une caractéristique.

- Chaque document est représenté par un vecteur (ligne) dans la matrice des caractéristiques et chaque colonne représente un mot unique en tant que caractéristique.

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green	ham	jumps	kings	lazy	love	quick	sausages	sky	toast	today
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
2	0	0	0	0	0	1	1	0	1	0	0	1	0	1	0	1	0	0	0	0
3	1	1	0	0	1	0	0	1	0	0	1	0	1	0	0	0	1	0	1	0
4	1	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	0	0	0
5	0	0	0	1	0	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0
6	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1
7	0	0	0	0	0	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0

Modèle Bag of Words : Lab

1. Importer les dépendances et les paramètres nécessaires :

import pandas as pd // La manipulation et l'analyse des données.
import numpy as np // Manipuler des matrices ou tableaux multidimensionnels
import re // Regular expression
import nltk // Natural Language Toolkit est une bibliothèque logicielle en Python permettant un traitement automatique des langues
import matplotlib.pyplot as plt // Tracer et visualiser des données sous forme de graphiques

Modèle Bag of Words : Lab

2. Constituer le corpus de texte

```
corpus = ['The sky is blue and beautiful.',
        'Love this blue and beautiful sky!',
        'The quick brown fox jumps over the lazy dog.',
        "A king's breakfast has sausages, ham, bacon, eggs, toast and beans",
        'I love green eggs, ham, sausages and bacon!',
        'The brown fox is quick and the blue dog is lazy!',
        'The sky is very blue and the sky is very beautiful today',
        'The dog is lazy but the brown fox is quick!'
]
labels = ['weather', 'weather', 'animals', 'food', 'food', 'animals', 'weather', 'animals']

corpus = np.*****(corpus)
corpus_df = pd.*****({'Document': corpus, 'Category': labels})
corpus_df = corpus_df[[**********]]
```


Modèle Bag of Words : Lab

Afficher corpus_df

index	Document	Category
0	The sky is blue and beautiful.	weather
1	Love this blue and beautiful sky!	weather
2	The quick brown fox jumps over the lazy dog.	animals
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food
4	I love green eggs, ham, sausages and bacon!	food
5	The brown fox is quick and the blue dog is lazy!	animals
6	The sky is very blue and the sky is very beautiful today	weather
7	The dog is lazy but the brown fox is quick!	animals

Modèle Bag of Words : Lab

3. Définir une fonction pour un prétraitement simple du corpus qui consiste a minuscules et supprimer les caractères spéciaux\l'espace blanc → tokeniser → enlever les stopwords : Utiliser nltk.WordPunctTokenizer()

4. Déclarer le Modèle de sac de mots en utilisant CountVectorizer : from sklearn.feature_extraction.text import CountVectorizer

- Visualiser les positions non nulles de l'élément dans la matrice éparse
- Visualiser sous forme de suite de vecteurs
- Visualiser sous forme de Matrice sparse dont les colonnes sont le vocabulaire et les lignes les documents
- Que remarquez vous ?

Gestion du vocabulaire

- À mesure que la taille du vocabulaire augmente, la représentation vectorielle des documents croît également.
- Dans un exemple précédent, la longueur du vecteur document équivaut au nombre de mots connus.
- Pour un corpus important comme des milliers de livres, le vecteur peut comporter des milliers voire des millions de positions, avec peu de mots présents dans chaque document.
- Cela génère des vecteurs épars, avec de **nombreuses valeurs nulles**, nécessitant plus de ressources pour la modélisation, rendant le processus complexe pour les algorithmes traditionnels.

→ **Besoin de réduire la taille du vocabulaire dans le modèle de sac de mots**.

- Des techniques de nettoyage simples, telles que l'ignorance de la casse, de la ponctuation, des mots fréquents ou vides et la réduction des mots à leur racine (stem), sont utilisées pour cette réduction.

Modèle Bag of N-Grams

- Une approche plus sophistiquée consiste à créer un vocabulaire de mots groupés. Cela élargit le champ du vocabulaire et permet au sac de mots de capturer un peu plus de sens du document.

- Dans cette approche, chaque mot ou jeton est appelé un "gramme". La création d'un vocabulaire de paires de mots est appelée modèle de bigramme. Seuls les bigrammes présents dans le corpus sont modélisés, pas tous les bigrammes possibles.

- Un N-gramme est fondamentalement une collection de tokens de mots provenant d'un document textuel, tels que ces tokens sont contigus et se produisent dans une séquence. Les bi-grammes indiquent des n-grammes d'ordre 2 (deux mots), les tri-grammes indiquent des n-grammes d'ordre 3 (trois mots), et ainsi de suite.
- Le modèle de sac de N-grammes est une extension du modèle de sac de mots qui exploite les caractéristiques basées sur les N-grammes.

Modèle Bag of N-Grams : Exemple

Phrase 1: “This is a good job. I will not miss it for anything”
Phrase 2: ”This is not good at all”

Pour cet exemple, prenons un vocabulaire de 5 mots seulement : {good, job, miss, not, all}

Les vecteurs respectifs de ces phrases sont donc les suivants :
“This is a good job. I will not miss it for anything”=[1,1,1,1,0]
”This is not good at all”=[1,0,0,1,1]

Quel est le problème ici ? La phrase 2 est négative et la phrase 1 est positive. Mais, pas reflété au niveau des vecteurs ? → Solution : N-grammes

Par exemple, les bigrammes de la première ligne du texte de la section précédente : " This is not good at all " sont les suivants : “This is”, “is not”, “not good”, “good at”, “at all”

→ Modèle peut différentier entre Phrase 1 et Phrase 2

Modèle Bag of N-Grams : Lab

Définissez la plage de n-grammes à 1,2 :
pour obtenir des unigrammes ainsi que des bigrammes en utilisant ngram_range=(*min_n, max_n*)

Pour obtenir uniquement des bigrammes

Modèle TF-IDF

Le modèle Bag of Words peut avoir des **problèmes** avec de **larges corpus** en **favorisant les termes fréquents**, occultant ainsi d'autres mots moins courants mais potentiellement pertinents.

→ **Solution : TF-IDF (Term frequency-Inverse document frequency).** Essentiellement, il mesure l'importance d'un mot en comparant sa fréquence dans un document spécifique à sa fréquence dans l'ensemble du corpus. L'hypothèse sous-jacente est qu'un mot qui **apparaît plus fréquemment dans un document mais rarement dans le corpus est particulièrement important dans ce document.**

→ Mathématiquement, le TF-IDF est le produit de deux mesures et peut être représenté comme suit :

$$tfidf = tf \times idf$$

- Le TF-IDF est noté entre 0 et 1. Plus la valeur numérique est élevée, plus le terme est rare. Plus le poids est faible, plus le terme est courant.

- TF = (Nombre de fois où le mot apparaît dans le document) / (Nombre total de mots dans le document)

Modèle TF-IDF

- IDF : mesure l'importance d'un mot dans l'ensemble du corpus
- Dans notre implémentation, nous ajouterons 1 à la fréquence des documents pour chaque terme afin d'indiquer que nous disposons d'un document supplémentaire dans notre corpus, qui contient essentiellement tous les termes du vocabulaire. Cela permet d'éviter les erreurs potentielles de division par zéro. Nous ajoutons également 1 au résultat de notre calcul de l'idf pour éviter d'ignorer les termes dont l'idf est nulle

$$idf(w,D) = 1 + \log \frac{N}{1 + df(w)}$$

- idf (w,D) représente l'idf pour le terme/mot w dans le document D, N représente le nombre total de documents dans notre corpus, et df(w) représente le nombre de documents dans lesquels le terme w est présent.

- Nous appliquerons une métrique finale TF-IDF normalisée à partir de la matrice TF-IDF en la divisant par sa norme L2, qui représente la racine carrée de la somme des carrés des poids TF-IDF de chaque terme.

$$tfidf = \frac{tfidf}{\| tfidf \|}$$

Modèle TF-IDF : Implémentation avec Python (Devoir noté)

1. Implémentation :
    1. Configuration et prétraitement
    2. Calcul de l'IDF des tokens uniques dans le corpus
    3. Calcul du TF de chaque token pour chaque document
    4. Enfin, Calcul TF-IDF

2. Bonus : Construire un moteur de recherche avec TF-IDF

Modèle TF-IDF : Lab

Utilisation de TfidfTransformer

TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True)

smooth_idfbool, default=True → **L**isse les poids idf en ajoutant un à la fréquence des documents, comme si un document supplémentaire contenant exactement une fois chaque terme de la collection était vu. Empêche les divisions par zéro.

use_idf *bool, default=True* → **A**ctive la repondération de la fréquence inverse du document. Si False, idf(t) = 1.

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

Modèle TF-IDF : Lab

Utilisation de TfidfVectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer(min_df=0., max_df=1., norm='l2',  use_idf=True, smooth_idf=True)
```

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer