Bag of words (Countvectorizer - TF-IDF- BM25 et Text similarity)

Réalisé par:
Equipe des stagiaires (NLP)

Encadré par :
Mr.EL-FENAOUY Reda
Mr.Thierry Bertin

2022-2023

Plan

- Introduction
- Feature extraction
- Bag of words
- Count vectorizer
- TF-IDF
- Okapi BM25
- Text similarity
- Application pratique
- Conclusion

Introduction

L'un des aspects les plus fondamentaux de l'analyse de texte et du traitement de langage naturel est la représentation de texte en tant que données exploitables pour les algorithmes informatiques. Le sac de mots (Bag of Words) est l'une des approches les plus simples et les plus courantes pour représenter du texte sous forme de vecteurs numériques. Dans cette méthode, chaque mot dans le texte est considéré comme une caractéristique, et une matrice de comptage des occurrences de chaque mot est créée pour représenter le texte. Plusieurs techniques peuvent être utilisées pour améliorer la qualité de la représentation, notamment TF-IDF, et BM25. Ces techniques permettent également de mesurer la similarité entre deux textes en utilisant des mesures de distance et de similarité appropriées.

Feature Extraction

L'extraction de caractéristiques en traitement du langage naturel (NLP) est le processus de conversion de données textuelles brutes en caractéristiques numériques ou catégorielles pouvant être utilisées pour des modèles d'apprentissage automatique.
En NLP, les modèles de machine learning ont besoin de données textuelles sous forme de chiffres ou de catégories pour être entraînés et effectuer des prédictions précises.
L'extraction de caractéristiques permet de transformer le texte en vecteurs de caractéristiques exploitables par les algorithmes de machine learning.

Techniques de feature extraction :

- Bag of words
- Count vectorizer
- TF-IDF
- Word2vec

Bag of words

La technique de sac de mots (BoW) est une méthode simple et couramment utilisée pour extraire des caractéristiques à partir de données textuelles. Elle consiste à représenter un texte sous forme d'un vecteur de fréquences de mots, sans tenir compte de l'ordre ou du contexte dans lequel les mots apparaissent.

Fonctionnement :

- Tokenisation: La division du texte en mots
- Création du vocabulaire : Un vocabulaire est créé à partir des mots uniques trouvés dans le texte.
- Comptage des occurrences : Pour chaque mot dans le vocabulaire, on compte le nombre d'occurrences dans le texte.
- Création du vecteur BoW : Le vecteur BoW est créé en plaçant le nombre d'occurrences de chaque mot dans le vocabulaire dans un vecteur de dimension égale à la taille du vocabulaire.

Bag of words

Exemple:

- "the cat sat"
- "the cat sat in the hat"
- "the cat with the hat"

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

Vecteur 1 : **[1 1 1 0 0 0]**

Vecteur 2: **[2 1 0 0 1 1]**

Vecteur 3: **[2 1 1 1 1 0]**

Source::

- https://victorzhou.com/blog/bag-of-words/

Bag of words use cases

Classification de texte

Clustering de texte

Analyse des sentiments

Reconnaissance des entités nommées

Détection de langue

Extraction de mots-clés

Modélisation de sujet

Count vectorizer

La technique de CountVectorizer est une implémentation spécifique de la technique de BoW qui compte le nombre d'occurrences de chaque mot dans un document. La technique de BoW est plus générale et peut être appliquée à différents niveaux de granularité quelque soit document , phrase. Le CountVectorizer offre également des fonctionnalités supplémentaires telles que :

- Normalisation : Qui compare les documents de longueurs différents de maniére equitables

$$TF(t,d) = f(t,d) / \Sigma f(w,d)$$

- Taille maximale(maximale length) : Qui limite le nombre de mots prise en consideration et évite l'overfitting

TF-IDF

TF-IDF (term frequency-inverse document frequency) est un système de représentation des termes couramment utilisé pour représenter les documents textuels sous forme de vecteurs (à des fins de classification, de regroupement, de visualisation, de recherche, etc.) , on utilisant les règles suivant :

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

La fréquence de terme (TF) : Il s'agit du rapport entre le nombre de fois où le mot apparaît dans un document et le nombre total de mots dans ce document.

$$idf(w) = log(\frac{N}{df_t})$$

La fréquence de document inverse (IDF) : utilisé pour calculer le poids des mots rares dans tous les documents du corpus. Les mots qui apparaissent rarement dans le corpus ont un score IDF élevé.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

TF-IDF (score): le produit de **TF** et **IDF**

Source::

- (2017). TF–IDF. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7687-1_832
- https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/

TF-IDF

Sentence 1 : The car is driven on the road.

Sentence 2: The truck is driven on the highway.

1-création de data frame(corpus)

Word
The
Car
Truck
Is
Driven
On
The
Road
Highway

Source::

- https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/

TF-IDF

Sentence 1 : The car is driven on the road.

Sentence 2: The truck is driven on the highway.

2-calculons le TF

Word	TF	
	A	B
The	1/7	1/7
Car	1/7	0
Truck	0	1/7
Is	1/7	1/7
Driven	1/7	1/7
On	1/7	1/7
The	1/7	1/7
Road	1/7	0
Highway	0	1/7

Source::

- https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/

TF-IDF

Sentence 1 : The car is driven on the road.

Sentence 2: The truck is driven on the highway.

2-calculons le IDF

Word	TF		IDF
	A	B	
The	1/7	1/7	log(2/2) = 0
Car	1/7	0	log(2/1) = 0.3
Truck	0	1/7	log(2/1) = 0.3
Is	1/7	1/7	log(2/2) = 0
Driven	1/7	1/7	log(2/2) = 0
On	1/7	1/7	log(2/2) = 0
The	1/7	1/7	log(2/2) = 0
Road	1/7	0	log(2/1) = 0.3
Highway	0	1/7	log(2/1) = 0.3

Source::

- https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/

TF-IDF

Sentence 1 : The car is driven on the road.

Sentence 2: The truck is driven on the highway.

2-TF-IDF

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Source::

- https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/

TfidfVectorizer

Un module de la bibliotheque scikit-learn permet de Convertir une collection de documents bruts en une matrice de caractéristiques TF-IDF.
Équivalent de CountVectorizer suivi de TfidfTransformer.

Le fonctionnement de TfidfVectorizer

1. Prétraitement des données textuelles
2. Calcul de la fréquence des termes (tf)
3. Calcul de l'inverse de la fréquence des documents (idf)
4. Normalisation des vecteurs

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

BM25

BM25 (Okapi Best Matching 25) est un algorithme de pondération de termes utilisé dans la recherche d'informations et le traitement automatique du langage naturel. Il est largement utilisé pour le classement des résultats de recherche et la récupération de documents pertinents dans un corpus.

BM25 utilise une fonction de score qui attribue un poids à chaque terme dans un document, en fonction de la fréquence des termes dans le corpus de documents et de la fréquence des termes dans chaque document. Le score est calculé à l'aide d'une formule mathématique qui prend en compte plusieurs facteurs, tels que la longueur du document et la rareté des termes.

Source:
https://www.analyticsvidhya.com/blog/2021/06/part-11-step-by-step-guide-to-master-nlp-syntactic-analysis/

BM25

Il s'agit d'une mise à niveau de la TF IDF :

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \longrightarrow \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

$$idf(w) = log(\frac{N}{df_t}) \longrightarrow \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \longrightarrow \text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

f(qi,D) est la fréquence du terme qi dans le document **D** , **|D|** est la longueur du document **D** en nombre de mots, et **avgdl** est la longueur moyenne des documents dans la collection considérée. **k1** et **b** sont des paramètres libres pouvant être optimisés selon les cas d'usage mais qui, en l'absence de toute optimisation sont usuellement fixés à **k1∈[1.2,2.0]** et **b=0.75**. **IDF(qi)** est la fréquence inverse de document pondérant le terme **qi** de la requête.

BM25Okapi

Rank-BM25:une bibliothèque qui offre une Collection d'algorithmes permettant d'interroger un ensemble de documents et de renvoyer ceux qui sont les plus pertinents par rapport à la requête

parmi ces algorithmes :

- Okapi BM25
- BM25L
- BM25+
- BM25-Adpt
- BM25T

Sournce : https://pypi.org/project/rank-bm25/

BOW vs TF-IDF

Le modèle Bag of Words crée simplement un ensemble de vecteurs contenant le nombre d'occurrences de mots dans le document, tandis que le modèle TF-IDF contient des informations sur les mots les plus importants et les moins importants.

Les vecteurs du sac de mots sont faciles à interpréter. Cependant, le modèle TF-IDF est généralement plus performant dans les modèles d'apprentissage automatique.

Source:

https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

TF-IDF vs BM25

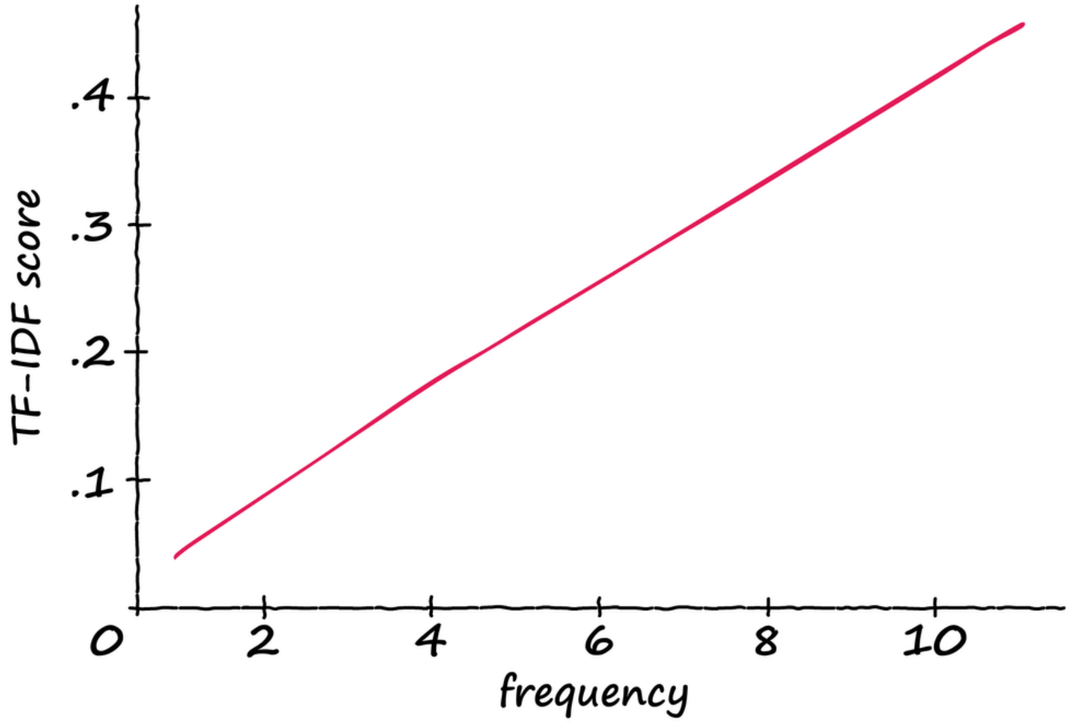
Le successeur de TF-IDF, Okapi BM25 est le résultat de l'optimisation de TF-IDF principalement pour normaliser les résultats en fonction de la longueur du document.

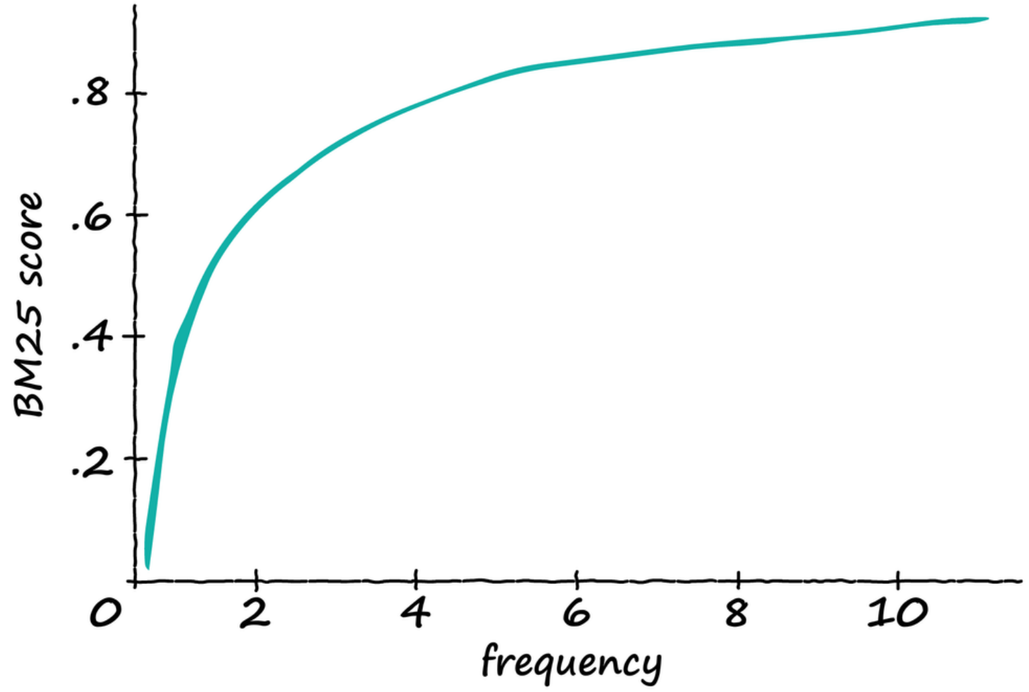
TF-IDF est génial mais peut renvoyer des résultats douteux lorsque nous commençons à comparer plusieurs mentions, Le score TF-IDF augmente linéairement avec le nombre de tokens pertinents

Exemple:
Si nous prenions un articles de 1000 mots et constations que l'article mentionne « chien » dix fois, si on double ce nombre en vingt fois, **tf-idf** score pour ce document va doubler aussi,
Donc logiquement doit-on considérer l'article premier comme à moitié pertinent ?

TF-IDF vs BM25

Le TF-IDF simple favorise la fréquence des termes et pénalise la fréquence des documents BM25 dépasse Le TF-IDF pour tenir compte de la longueur des documents et de la saturation de la fréquence des termes.





Sournce :
https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/

Text Similarity

La similarité textuelle est le processus qui consiste à comparer un texte à un autre et à trouver les similitudes entre eux. Il s'agit essentiellement de déterminer le degré de proximité du texte.
le calcul de similarité nécessite tout d'abord la conversion de texte vers une représentation numérique (des vecteurs)
La similarité de texte est largement utilisée dans le traitement automatique du langage naturel, notamment pour des tâches telles que la recherche d'informations, la classification de textes, l'analyse de sentiments, ou la traduction automatique.

Sournce : https://medium.com/msackiit/what-is-text-similarity-and-how-to-implement-it-c74c8b641883

Text Similarity

Il existe plusieurs techniques pour calculer la similarité entre textes :

- La distance euclidienne
- La similarité Jaccard
- La similarité cosinus
- La distance de Levenshtein
- La similarité de Jaro-Winkler
- etc..

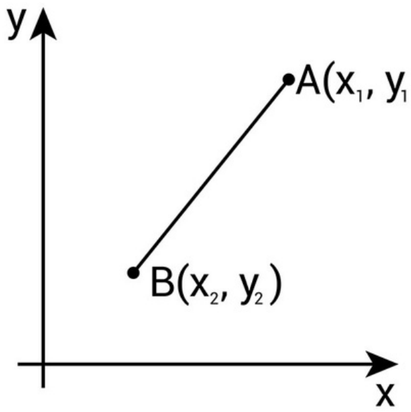
Sournce :https://medium.com/msackiit/what-is-text-similarity-and-how-to-implement-it-c74c8b641883

Text Similarity

La distance euclidienne

est une mesure de la distance entre deux vecteurs qui est définie comme la racine carrée de la somme des carrés des différences entre les valeurs des dimensions correspondantes dans les vecteurs. Elle peut être utilisée pour mesurer la distance entre les vecteurs de termes de deux textes.

Formule :



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Sournce :https://science.howstuffworks.com/math-concepts/distance-formula.htm

Text Similarity

La similarité de Jaccard

est une mesure de la similarité entre deux ensembles qui est définie comme le nombre d'éléments communs divisé par le nombre total d'éléments dans les deux ensembles. Elle est souvent utilisée pour comparer des ensembles de mots dans des textes.

Formule :

$$J(doc_1, doc_2) = \frac{doc_1 \cap doc_2}{doc_1 \cup doc_2}$$

Text Similarity

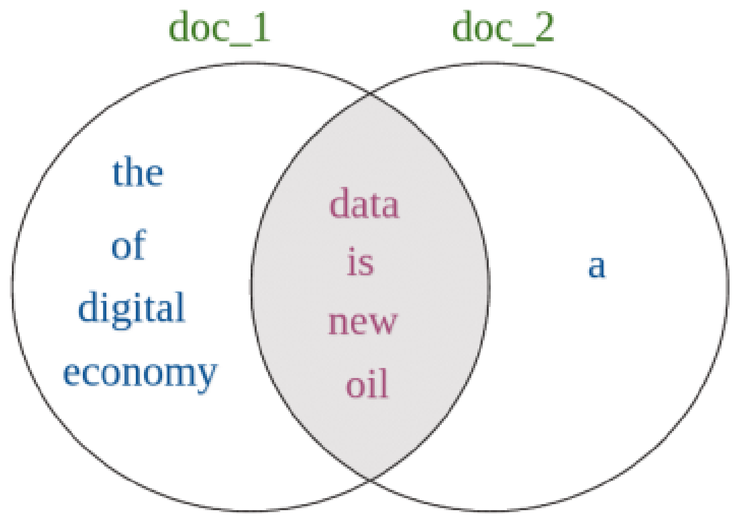
La similarité de Jaccard

doc_1 = "Data is the new oil of the digital economy"
doc_2 = "Data is a new oil"

$$J(doc_1, doc_2) = \frac{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cap \{'data', 'is', 'a', 'new', 'oil'\}}{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cup \{'data', 'is', 'a', 'new', 'c}$$

$$= \frac{\{'data', 'is', 'new', 'oil'\}}{\{'data', 'a', 'of', 'is', 'economy', 'the', 'new', 'digital', 'oil'\}}$$

$$= \frac{4}{9} = 0.444$$

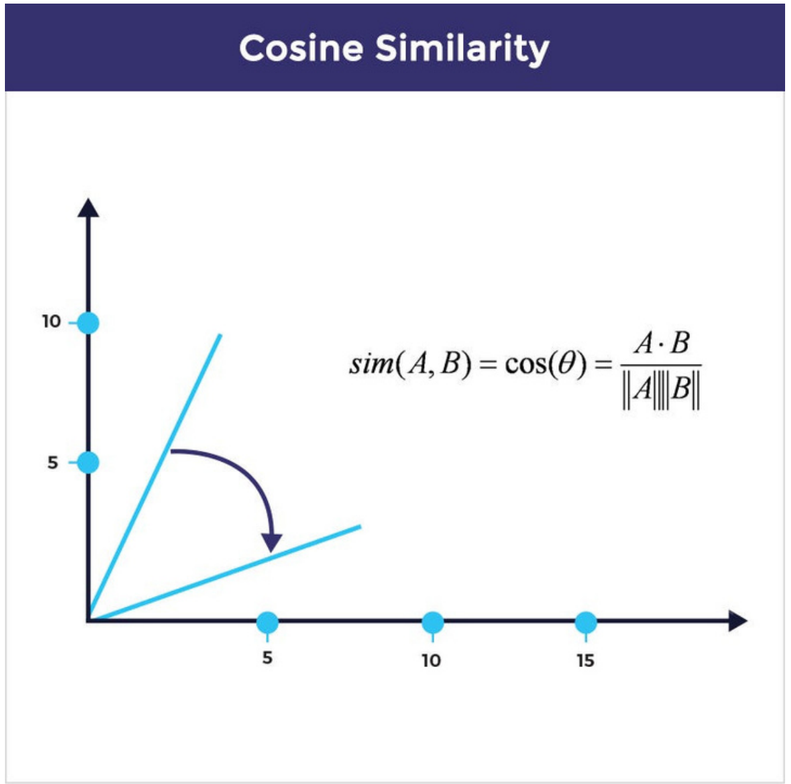


Sournce https://studymachinelearning.com/jaccard-similarity-text-similarity-metric-in-nlp/

Text Similarity

La similarité cosinus

est une mesure de la similarité entre deux vecteurs qui prend en compte leur angle dans un espace multidimensionnel. Elle est souvent utilisée pour mesurer la similarité entre deux documents représentés sous forme de vecteurs de termes.

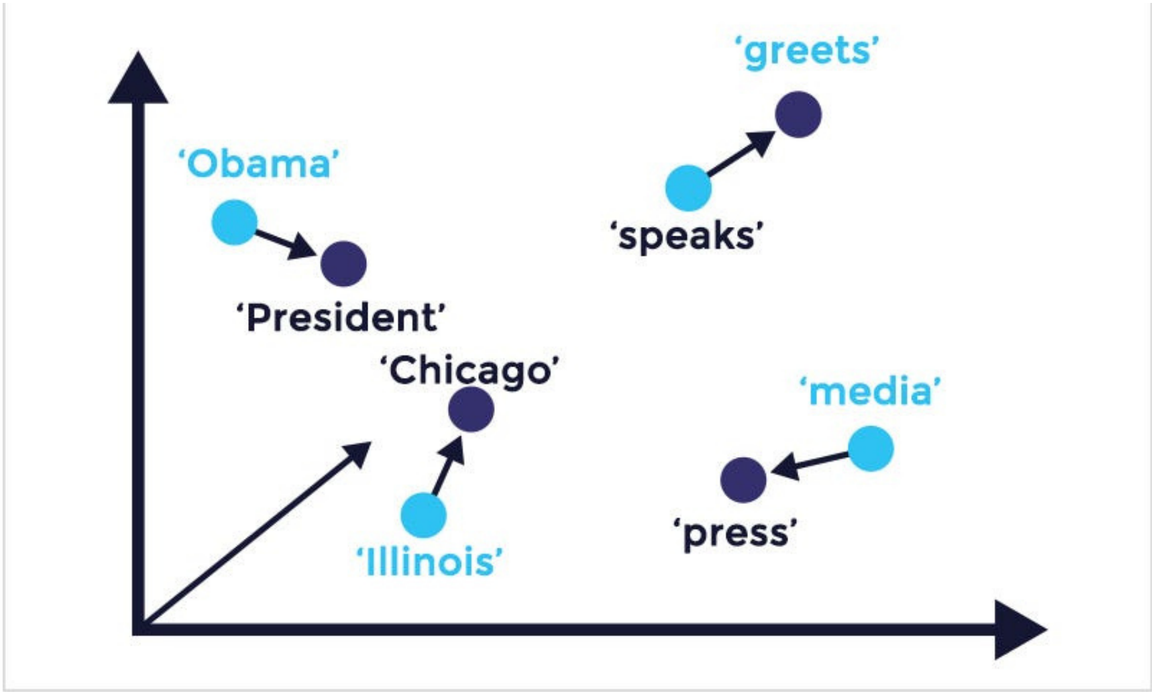


Sournce https://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/%5D

Text Similarity

La similarité cosinus

Document 1	Document 2
Obama	The
speaks	Presidents
to	greets
the	the
media	press
in	in
Illinois	Chicago



Sournce https://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/%5D

Application pratique

rank_bm25

Conclusion

En conclusion, nous avons présenté différentes méthodes pour représenter les textes sous forme de vecteurs, en utilisant les techniques Bag of Words. Nous avons vu que les approches de CountVectorizer, TF-IDF et Okapi BM25 offrent des représentations efficaces pour l'extraction d'informations et la recherche d'informations similaires. Nous avons également vu comment mesurer la similarité entre des textes en utilisant des mesures de distance et de similarité telles que l'Euclidean distance, la Jaccard similarity et la cosine similarity. En somme, ces techniques sont essentielles pour analyser des données textuelles à grande échelle et permettent d'extraire des informations précieuses à partir de textes.

Merci pour votre attention !