

***mohammed fakil***

Word Embedding

## introducion :

Vous êtes-vous déjà demandé comment des assistants personnels IA tels que Siri ou Cortana fonctionnent ? Comment votre correcteur d’orthographe a été capable de détecter des erreurs de syntaxe que vous-même n’auriez pas repérées ? Comment votre moteur de recherche réussite à deviner les mots que vous étiez sur le point d’écrire dès les premiers lettres ?

Ces outils reposent tous sur des méthodes communes celles de ***Natural Language Processing (NLP).***

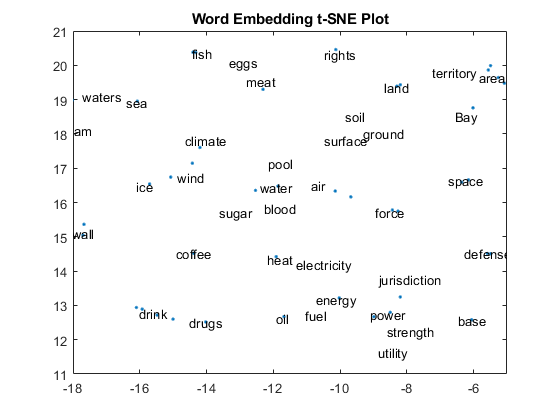
* ***NLP :*** C’estune technologie permettant aux machines de comprendre le langage humain grâce à l’intelligence artificielle.

Pour améliorer les performances de traitement automatique des langues (ou ***Natural Language Processing),*** il existe des nouvelles représentations de données textuelles telle que le ***Word embedding.***

## word embedding :

Le ***word embedding*** («plongement lexical» en français) est une méthode d'apprentissage d'une représentation de mots utilisée notamment en ***traitement automatique des langues(NLP)***. Le terme devrait plutôt être rendu par ***vectorisation de mots*** pour correspondre plus proprement à cette méthode.

Cette technique permet de représenter chaque mot d'un dictionnaire par un vecteur de nombres réels. Cette nouvelle représentation a ceci de particulier que les mots apparaissant dans des contextes similaires possèdent des vecteurs correspondants qui sont relativement proches. Par exemple, on pourrait s'attendre à ce que les mots « energy » et « fuel » soient représentés par des vecteurs relativement peu distants dans l'espace vectoriel où sont définis ces vecteurs. Cette technique est basée sur l'hypothèse (dite « de Harris » ou *distributional hypothesis*) qui veut que les mots apparaissant dans des contextes similaires ont des significations apparentées.



OBJECTIF :

L’objectif de ce mini-projet est developpé une application web afin de visualiser les incorporations des mots arabes en utilisant le word embedding .Vous trouverez ci-après les étapes importantes pour la création de cette application web :

# Etape 1 : Le choix de corpus

La première étape de la réalisation de modèle est la récupération du texte. Il existe plusieurs manières de récupérer du texte : soit depuis une base de données que vous possédez, soit depuis des fichiers XML ou autres que vous possédez, soit en sacrpant des pages comme le font les moteurs de recherches, en utilisant une API.

Dans notre cas, on va récupérer le texte depuis une base de données (arabic-twitter) en utilisant la librairie pandas

Chargeons donc les données dans un fichier python, ce qu’on a fait avec la fonction pd de librairie pandas.

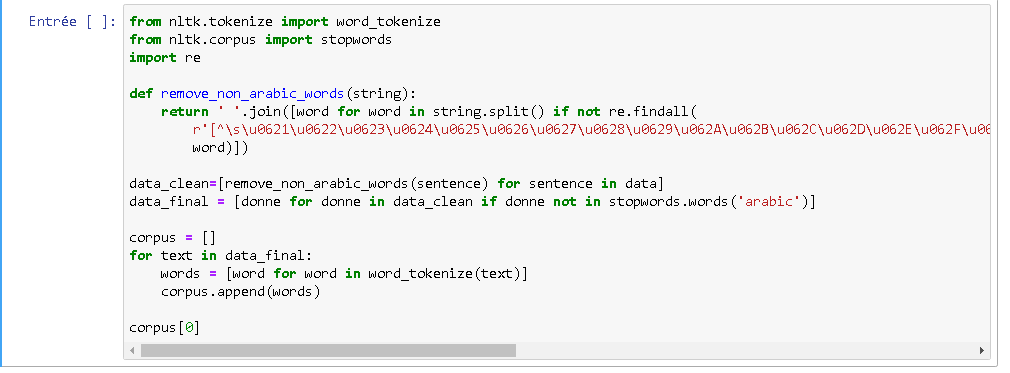


# Etape 2 : pretraitement et segmentation des donnees

Après avoir chargé la base de données il est indispensable de transformer les données brutes (arabic-twitter) en des données exploitables. Cette transformation consiste sur :

* ***Nettoyage* :**Variable selon la source des données, cette phase consiste à réaliser des tâches telles que la ***suppression d’url*, *d’emoji*** etc.
* ***Normalisation des données :***
* ***Tokenisation*** **:** au découpage de texte en plusieurs pièces appelées tokens.
* ***Stemming* :** désigne généralement le processus heuristique brut qui consiste à découper la fin des mots dans afin de ne conserver que la racine du mot.
* ***Autres opérations* :** suppression des chiffres, ponctuation, symboles et ***stopwords***.

### nous allons utiliser la librairie de traitement de langage nltk





# Etape 3 : Vectorisation et choix de l’architecture

Afin de pouvoir réaliser un modèle, il est indispensable de transformer les données récupérées dans la phase précédente en données numériques (vecteurs).

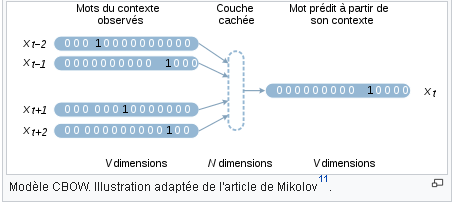
Il existe plusieurs modèles pour entraîner des word embeddings. Les plus connus sont Word2Vec et FastText de librairie ***gensim***.



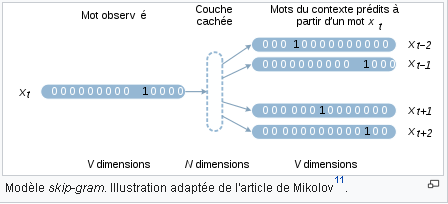
L'idée générale de ces deux modèles est de projeter un ensemble de mots d'un vocabulaire de taille {\displaystyle V}V dans un espace vectoriel continu où les vecteurs de ces mots ont une taille N{\displaystyle N}NNN relativement petite. De plus, on veut trouver une représentation vectorielle de chaque mot de {\displaystyle V}V de façon que les mots aux représentations voisines apparaissent dans des contextes similaires.

L'approche de Mikolov s'appuie sur ***des réseaux de neurones artificiels*** pour construire ces vecteurs. Ces modèles sont entraînés sur des corpus très volumineux. Le principe de base est qu'on tente de prédire un mot à partir de son contexte ou vice-versa. Un sous-produit de cet apprentissage constitue les plongements lexicaux. Il existe deux variantes de l'algorithme de vectorisation :

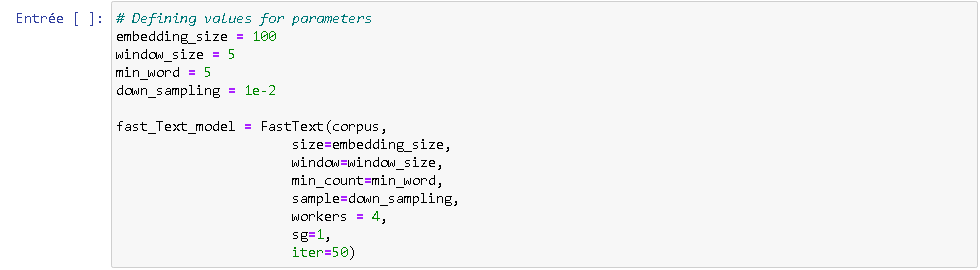
1. Le modèle ***continuous bag-of-words*** (sac de mots continu) cherche à prédire un mot à partir de ses mots voisins.



2. Le modèle ***skip-gram*** cherche à prédire les mots du contexte à partir d'un mot central, par exemple prédire les quatre mots *le*, *chat*, *la* et *souris* à partir de *attrape.*



* NB : Le *word embedding* utilisé dans notre application web est ***FastText*** avec la variante ***Skip-Gram***



Explorons les paramètres utilisés dans ce modèle.

**size:** Dimensionnalité des vecteurs de mots. window = window\_size,  
**min\_count:** Le modèle ignore tous les mots dont la fréquence totale est inférieure à cela.  
**sample:** Le seuil pour configurer les mots de fréquence plus élevée qui sont échantillonnés de manière aléatoire, la plage utile est (0, 1e-5).  
**workers:** utilisez ces nombreux threads de travail pour entraîner le modèle (= formation plus rapide avec des machines multicœurs).  
**sg:** algorithme d'[apprentissage](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=https://thinkinfi.com/2019/06/continuous-bag-of-words-cbow-single-word-model-how-it-works/&usg=ALkJrhjUH16tZM99Lg1jf164MR52sgg2Qg) :[skip-gram](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=https://thinkinfi.com/2019/08/word2vec-skip-gram-explained/&usg=ALkJrhg4J0rVoJDYxuaaaQYye02ScmvJ9A) si sg = 1, sinon[CBOW](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=https://thinkinfi.com/2019/06/continuous-bag-of-words-cbow-single-word-model-how-it-works/&usg=ALkJrhjUH16tZM99Lg1jf164MR52sgg2Qg) .  
**iter:** nombre d'itérations (époques) sur le corpus.

Après avoir créé le module on doit l’enregistrer, pour cela on va importer la librairie ***joblib***

******



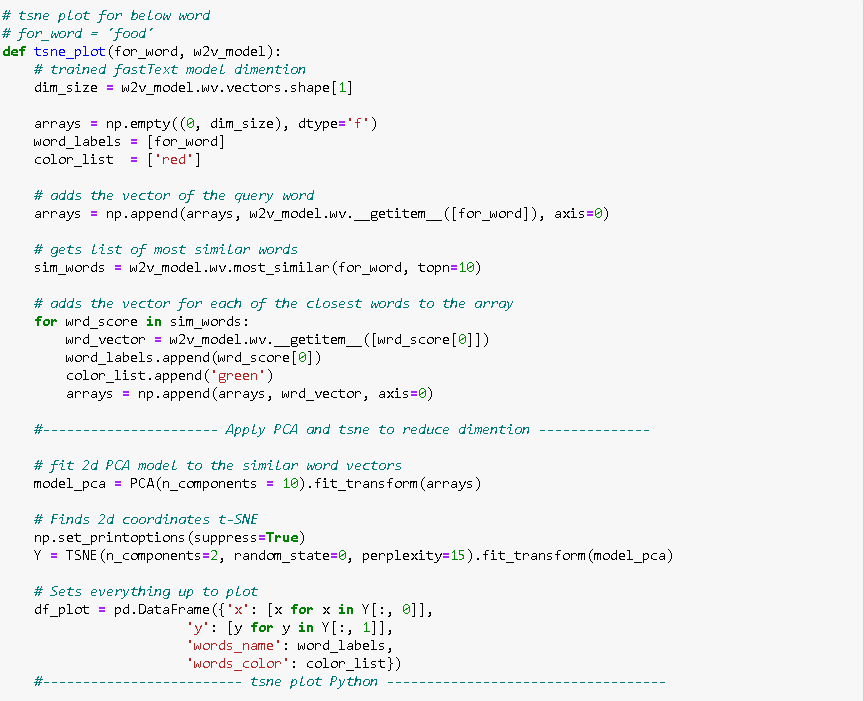
# Etape 4 : Reduction de dimensionnalité

Définition : La ***réduction de la dimensionnalité*** est un processus étudié en mathématiques et en informatique, qui consiste à prendre des données dans un espace de grande dimension, et à les remplacer par des données dans un espace de plus petite dimension. Pour que l'opération soit utile il faut que les données en sortie représentent bien les données d'entrée.

Il est difficile de visualiser directement les incorporations de mots FastText car l'incorporation de mots a généralement plus de 3 dimensions (dans notre cas 100).

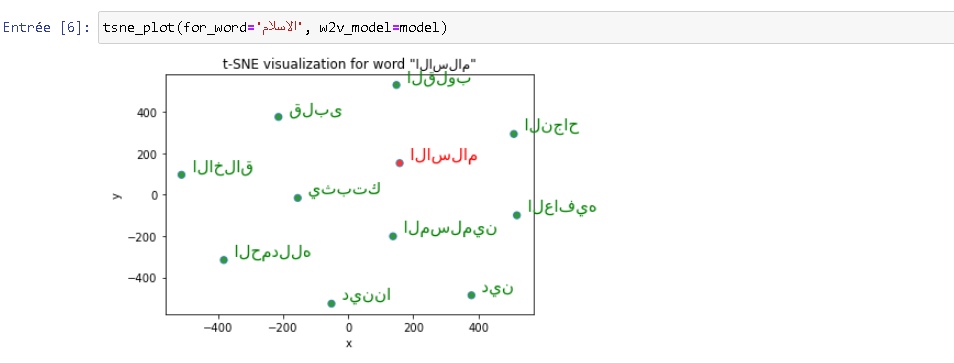
Maintenant, pour la visualisation de l'intégration de mots FastText, nous devons réduire la dimension en appliquant PCA (Analyse en Composantes Principales) et T-SNE.

Le code suivant permet de visualiser les incorporations de mots fastText à l'aide de tsne plot





Exemple :Visualisation de mots « الاسلام » après la réduction de dimensionnalité



# Etape 5 : realisation de l’application Web

Afin de visualiser les incorporations des mots arabes on a développé une application web à l’aide de Flask.

* ***Flask :*** est un micro framework-open source de développement web en python.

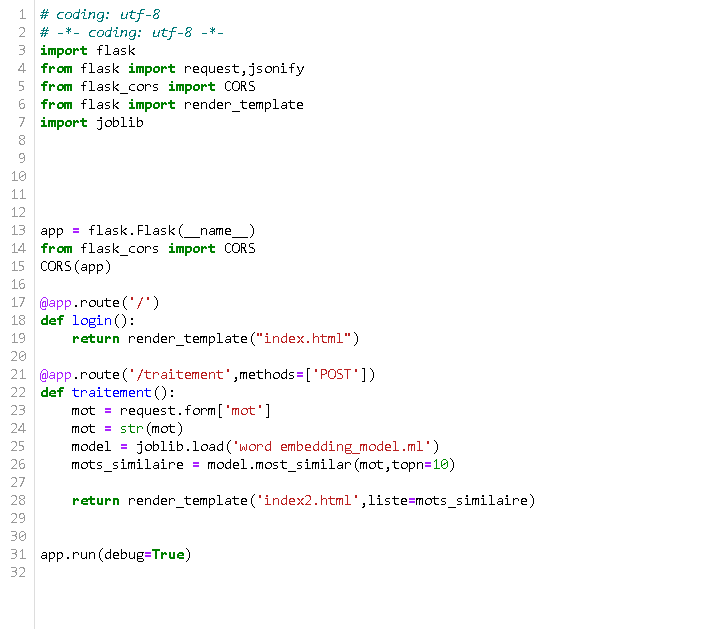
Les étapes de créations de l’application web :

***Etape 1 :***

***Créer une application de base*** :

* Créer un fichier « app.py » dans un répertoire « ***word embedding*** » .
* Importer les librairies ***flask*** et ***joblib.***
* Créer un chemin de routage sous forme de chaîne en utilisant ***@app.route*** (ici on va créer deux chemins un pour la page ***d’accueil*** et un autre pour la page des ***résultats***).
* Définir deux fonction ***login*** et ***traitement*** qui vont renvoyer respectivement les pages ***accueil*** et ***résultats*** par la fonction ***render\_template().***
* Utiliser la méthode ***load*** de la libraire ***joblib*** pour importer le modèle développé dans l’étape 3 afin d’afficher les 5 mots similaires de mot entrer par l’utilisateur .
* NB : Il faut utiliser la méthode ‘POST’ afin de récuperer la donnée entrée dans le champ de formulaire HTML par l’utilisateur

.



***Etape 2 :***

***Utilisation des modèles HTML :***

* Créer un répertoire « ***templates »*** dans le répertoire « ***word embedding*** ».
* Ouvrir deux fichiers html ***index.html*** et ***index2.html*** qui seront renvoyer par les deux fonctions ***login*** et ***traitement*** par la fonction ***render\_template()***.

***Index.html Index2.html***



***Etape 3 :***

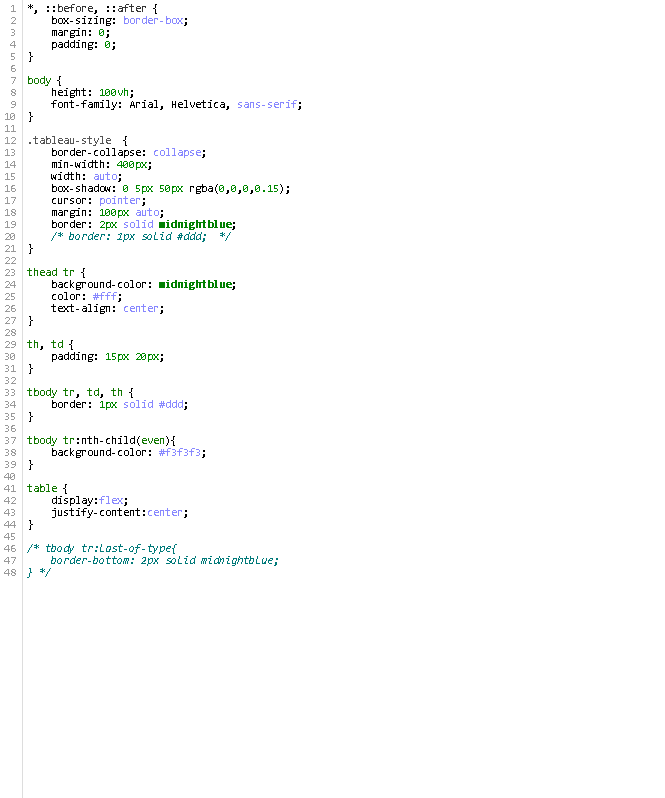
***Utilisation de modèles Css :***

* Créer un répertoire « ***static*** » dans le répertoire « ***word embedding*** ».
* Créer deux fichiers css ***style.css*** et ***style2.css.***
* NB :Il faut ajouter les liens entre les fichiers html et les fichiers css :
  1. ******
  2. ******

***Style.css***



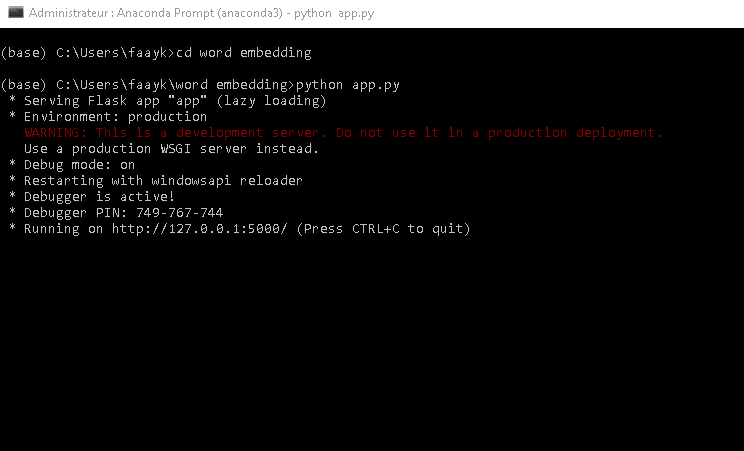
***Style2.css***



***Etape 4 :***

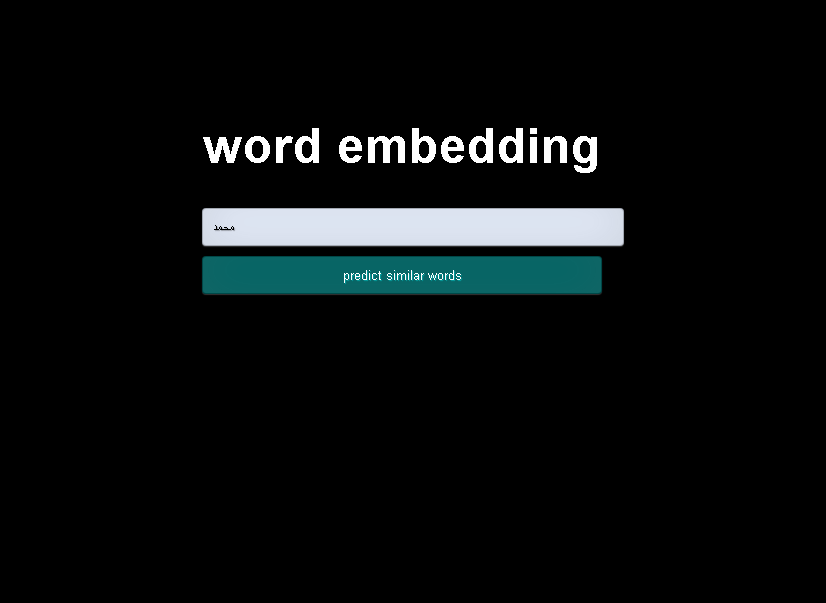
***Lancer le serveur :***

* Entrer dans Anaconda prompt (anaconda3)
* Accéder au répertoire « ***word embedding*** » par la commande ***cd***
* Entrer dans l’environnement ***python***
* lancer le fichier ***app.py***



***Etape 5 :***

* Entrer dans un navigateur et lancer l’application web (<http://127.0.0.1:5000/>)



* Resultat :



# Ressources :

<https://radimrehurek.com/gensim/models/fasttext.html>

<https://ichi.pro/fr/introduction-a-nlp-partie-1-pretraitement-du-texte-en-python-157232264432278>

<https://radimrehurek.com/gensim/models/word2vec.html>

<https://flask.palletsprojects.com/en/1.1.x/>