Projet Python
My first chatBot

Consignes & Informations générales :

⇒ Ce projet est à réaliser exclusivement en langage Python

⇒ Pièces jointes au sujet du projet :
- Fichiers des discours de présidents

⇒ Organisation des équipes :
- Ce projet est à réaliser en binôme (**un seul trinôme est autorisé** si un **nombre impair** d'élèves)
- La liste des équipes est à remettre aux enseignants **au plus tard** à la fin de la première séance de suivi de projet

⇒ Dates clé :
- Date de publication : **04/11/2023**
- Date de suivi 1 : Semaine du **13/11/2023**
- Date du premier dépôt : **26/11/2023** à **23h59**
- Date de suivi 2 : Semaine du **27/11/2023**
- Date du dépôt final : **17/12/2023** à **23h59**
- Date de soutenance : Semaine du **18/12/2023**

⇒ Rendu final :
- Le code du projet contenant les fichiers **.py** et **.txt**
- Le rapport en **.pdf**
- Un fichier **README.txt** donnant la liste des programmes et comment les utiliser en pratique. Ce fichier doit contenir toutes les instructions nécessaires à l'exécution ; il est très important pour expliquer à l'utilisateur comment se servir de l'outil.

⇒ Dépôt du projet :
- Le dépôt du projet se fera sous Moodle

⇒ Évaluation
- Barème détaillé : sera fourni plus tard
- Note finale du projet = Note code + Note rapport + Note soutenance
- Rappel : note projet = 20% de la note du cours « Programmation Python »
- Les membres d'une même équipe peuvent avoir des notes différentes en fonction des efforts fournis dans la réalisation de ce projet.

⇒ **Travail collaboratif**
- Le suivi du travail au sein d'une équipe projet se fera via Git
- Le tutoriel d'utilisation de Git sera mis en ligne la semaine du **06/11/2023**

Organisation du code :

⇒ La notation du code tiendra compte principalement de :
- L'implémentation des fonctionnalités demandées : avancer au mieux mais PAS en hors sujet
- La qualité du code fourni : organisation en fonctions, **commentaires**, noms de variables significatifs, respect des noms de fichiers.
- Ergonomie : Facilité d'utilisation de l'interface utilisateur

Notions pédagogiques traitées :

⇒ La réalisation de ce projet vous aidera à appliquer les notions pédagogiques suivantes :
- Notions des bases, listes 1D, listes 2D, les fonctions, les fichiers et les collections

⇒ Pour vous aider à la réalisation de ce projet, vous pouvez vous appuyer sur :
- Les supports du cours TI101 (I, B, P)
- Livres, cours divers sur le web. **ALERTE PLAGIAT !!** il ne s'agit pas de copier des programmes entiers.
- Enseignants **Efrei** lors des séances de suivi du projet

Description

Le projet que nous vous proposons porte sur l'analyse de texte. Ce projet vous permettra de comprendre quelques concepts de base utilisés dans le traitement de texte et de vous aider à comprendre l'une des méthodes utilisées pour développer des chatbot et/ou des intelligences artificielles génératives telles que : chatGpt.

Il est évident, qu'il ne s'agit pas ici de manipuler des réseaux de neurones, mais nous allons dans ce projet nous focaliser sur la méthode basée sur le nombre d'occurrences des mots permettant de générer des réponses intelligentes à partir d'un corpus de textes. L'objectif est de concevoir un système qui peut répondre à des questions en se basant sur la fréquence des mots dans le corpus.

Cette application repose sur l'algorithme de base suivant :

Prétraitement des données : votre programme commence par collecter et prétraiter un ensemble de documents afin de comprendre la nature de leurs contenus avant de s'en servir lors de l'élaboration des réponses. Cette phase permet de nettoyer le texte en supprimant la ponctuation, en convertissant les lettres en minuscules, et en divisant le texte en mots (ou "tokens").

Création d'une matrice TF-IDF : Pour chaque mot unique dans les documents, vous serez amenés à calculer un vecteur TF-IDF en utilisant la méthode TF-IDF. Chaque mot est associé à un vecteur de dimension égale au nombre de documents dans le corpus. Cela crée une matrice TF-IDF où chaque ligne représente un mot et chaque colonne représente un document.

Représentation des questions : Lorsqu'une question est posée, le chatbot effectue le même prétraitement sur la question. Ensuite, il calcule un vecteur TF-IDF pour cette question en utilisant le même vocabulaire que les documents. Le vecteur de la question est de la même dimension que les vecteurs associés aux mots du corpus.

Calcul de la similarité : Le chatbot calcule la similarité entre le vecteur de la question et les vecteurs des mots du corpus à l'aide de la similarité cosinus ou d'une autre mesure de similarité. Cela lui permet de déterminer quels mots du corpus sont les plus similaires à la question.

Sélection de la meilleure réponse : Le chatbot identifie les mots du corpus les plus similaires à la question en fonction de leur score de similarité TF-IDF. Ensuite, il sélectionne la réponse qui contient le plus grand nombre de ces mots similaires.

Fournir la réponse : Le chatbot renvoie la réponse sélectionnée comme réponse à la question posée.

Travail demandé

Pour vous guider dans la réalisation de cette application, nous allons la travailler en 3 parties :

1. **Partie I** : Développement des fonctions de base qui permettent de bien comprendre le contenu des fichiers fournis.
2. **Partie II** : Calcule de la matrice de similarité et la génération de réponses automatiques.

3. **Partie III** : Généralisation de l'application pour couvrir divers thèmes.

Partie I

Base de fichiers

Dans ce projet, 8 fichiers texte sont fournis dans un répertoire nommé « speeches ». Ces fichiers représentent des discours de 6 présidents français lors de leurs investitures.

Le nom de chaque fichier est sous le format suivant : **Nomination_[nom d'un president][numéro].txt** où :

- **[nom d'un president]** : Est l'un des noms suivants : Chirac, Giscrad d'Estaing, Mitterrand, Macron, Sarkozy
- **[numéro]** : Est un champs optionnel représentant un numéro séquentiel. Il figure dans les noms des fichiers où un même président a prononcé plusieurs discours.

Les fonctions de base

Afin de bien analyser le contenu de ces fichier, il est d'abord demandé de développer les fonctions suivantes :

- Extraire les noms des présidents à partir des noms des fichiers texte fournis ;
- Associer à chaque président un prénom ;
- Afficher la liste des noms des présidents (attention aux doublons) ;
- Convertir les textes des 8 fichiers en minuscules et stocker les contenus dans de nouveaux fichiers. Les nouveaux fichiers doivent être stockés dans un nouveau dossier appelé « cleaned ». Ce dossier doit se situer dans le répertoire principal où se trouve le programme ***main.py*** et au même niveau que le répertoire « speeches »
- Pour chaque fichier stocké dans le répertoire « cleaned », parcourir son texte et supprimer tout caractère de ponctuation. Le résultat final doit donner un fichier avec des mots séparés par des espaces. Attention, certains caractères comme l'apostrophe (') ou le tiret (-) nécessitent un traitement spécial pour ne pas causer une concaténation de deux mots (exemple : « elle-même » devrait devenir « elle même » et non pas « ellemême »). Les modifications réalisées à cette étape devraient être stockées dans les mêmes fichiers du répertoire « cleaned ».

Notes :

1. Pour réaliser ces fonctions, il est possible d'utiliser les modules prédéfinis abordés en cours mais aussi le module « os »
2. Le code Python utilisé pour parcourir la liste des fichiers d'une extension donnée et dans un répertoire donné est le suivant :

```python
import os

def list_of_files(directory, extension):

    files_names = []
    for filename in os.listdir(directory):
        if filename.endswith(extension):
            files_names.append(filename)
    return files_names
```

```python
# Call of the function
directory = "./speeches"
files_names = list_of_files(directory, "txt")
print_list(files_names)
```

La méthode TF-IDF

Dans le traitement de texte, il est souvent utile de représenter les mots sous forme de vecteurs numériques afin de faciliter la recherche d'information, la classification de documents et/ou l'analyse de texte.

Une des méthodes basée sur la fréquence pour générer des vecteurs numériques pour les mots s'appelle la méthode **TF-IDF** (**T**erm **F**requency-**I**nverse **D**ocument **F**requency). Voici comment elle fonctionne :

TF (Term Frequency - Fréquence du terme) : La première partie du **TF-IDF** mesure à quelle fréquence un terme (un mot dans notre cas) apparaît dans un document spécifique. Plus un terme apparaît fréquemment dans un document, plus son score **TF** est élevé pour ce document ➔ Pour chaque mot dans chacun des fichiers, il est demandé de compter le nombre de fois où ce mot apparaît. Pour ce faire, il est au minimum nécessaire d'avoir une fonction qui calcule le nombre d'occurrence de chaque mot dans un texte.

- Écrire une fonction qui prend en paramètre une chaine de caractères et qui retourne un dictionnaire associant à chaque mot le nombre de fois qu'il apparait dans la chaine de caractères.

IDF (Inverse Document Frequency - Fréquence Inverse du Document) : La deuxième partie du **TF-IDF** mesure l'importance d'un terme dans l'ensemble du corpus de documents. Les termes qui sont très fréquents dans de nombreux documents auront un score **IDF** plus faible, tandis que les termes qui sont rares auront un score **IDF** plus élevé. ➔ Pour chaque mot, calculer le **logarithme** de l'inverse de la proportion de documents dans le corpus qui contiennent ce mot. Cela permet de donner plus de poids aux mots rares et de réduire le poids des mots communs.

- Écrire une fonction qui prend en paramètre le répertoire où se trouve l'ensemble des fichiers du corpus et qui retourne un dictionnaire associant à chaque mot son score **IDF**.

Note :

Il est autorisé d'utiliser le module « math » afin de pouvoir appeler sa fonction prédéfinie calculant le logarithme.

Score final : Le score **TF-IDF** d'un mot dans un document est obtenu en multipliant le score **TF** par le score **IDF**.

- TF-IDF = TF * IDF.

Ainsi, le score **TF-IDF** d'un mot dans un document donné est un vecteur numérique qui reflète à la fois la fréquence du mot dans ce document et son importance relative par rapport à l'ensemble du corpus.

Pour générer des vecteurs numériques pour les mots, chaque mot dans le corpus de documents se voit attribuer un vecteur de scores **TF-IDF**. Ces vecteurs numériques permettent de représenter la distribution de chaque mot dans tout le corpus. L'ensemble de ces vecteurs forme une matrice appelée "**matrice TF-IDF**", où chaque ligne représente un mot et chaque colonne représente un document.

- Écrire une fonction qui prend en paramètre le répertoire où se trouvent les fichiers à analyser et qui retourne au minimum la matrice TF-IDF.
- Le nombre de lignes de la matrice résultante doit être égal au nombre de mots uniques dans tous les fichiers, et le nombre de colonnes doit être égal au nombre de fichiers se trouvant dans le répertoire
  ➔ Si besoin, écrire une fonction qui calcule la transposée d'une matrice (**Attention !!** ne pas utiliser de fonction prédéfinie)

Fonctionnalités à développer

Sur la base des fonctions précédentes, écrire des programmes qui permettent de :

1. Afficher la liste des mots les moins importants dans le corpus de documents. Un mot est dit non important, si son **TD-IDF** = 0 dans tous les fichiers.
2. Afficher le(s) mot(s) ayant le score **TD-IDF** le plus élevé
3. Indiquer le(s) mot(s) le(s) plus répété(s) par le président Chirac
4. Indiquer le(s) nom(s) du (des) président(s) qui a (ont) parlé de la « Nation » et celui qui l'a répété le plus de fois
5. Indiquer le premier président à parler du climat et/ou de l'écologie
6. Hormis les mots dits « non importants », quel(s) est(sont) le(s) mot(s) que tous les présidents ont évoqués.

Programme principal

Il s'agit ici d'écrire un programme principal temporaire dans lequel il est demandé de :

- Avoir un menu
- Accéder aux fonctionnalités précédentes selon la demande de l'utilisateur

À retenir :

⇒ Le premier dépôt est à effectuer le **26/11/2023** à **23h59**
⇒ Les consignes de ce dépôt seront fournies la semaine du 06/11/2023

Partie II

Si vous voyez ce message, c'est que vous avez bien tout lu, Bravo! Avancez bien sur la partie I pour avoir celle-ci ☺