MY FIRST CHATBOT

```python
import fitz

read = False

while not read:
        with fitz.open("MyFirstChatbot.pdf") as pdf_document:
            num_pages = pdf_document.page_count

            for page_num in range(num_pages):
                page = pdf_document[page_num]
                text = page.get_text("text")
            read = True
            print ("Vous avez fini de lire le PDF, bien joué !")
```

Yéléna SAINTE-ROSE et Justine GARNUNG
EFREI BORDEAUX L1

2023-2024

SOMMAIRE

Dans le cadre de notre projet My First ChatBot qui nous a été proposé dans le module « Programmation en Python » nous avons pour but de concevoir un programme capable de répondre à des questions en se basant sur la fréquence des mots dans un corpus, en l'occurrence des discours d'investitures présidentielles.

L'objectif de ce travail était de nous faire mettre en pratique tout ce que l'on a appris depuis le début de l'année en programmation. C'est-à-dire :

- Le prétraitement des données, le fait de « nettoyer » tous les textes de toutes les virgules, points, apostrophes, traits d'union…

- La création de la matrice TF-IDF, c'est elle qui permet de calculer la fréquence des mots dans les textes (s'ils sont très présents, ils auront un score TF-IDDF élevé, et inversement s'ils sont peu présents). Cette matrice fait de même dans les questions posées par les utilisateurs. Le but étant de calculer la similarité entre les scores TF-IDF des mots présents dans la question et dans les textes afin de préparer la réponse la plus adéquat à cette demande.

- Une sélection de la meilleure réponse, qui revient à donner une réponse inspirée des discours présidentiels, qui contient le plus de mots communs entre la question et les textes, toujours grâce à leur score TF-IDF.

- Et enfin de renvoyer une réponse à la question posée.

Après plusieurs semaines de travail sur ce projet, nous avons obtenu ce fameux programme, qui nous a permis d'automatiser une tâche redondante.

I. La présentation fonctionnelle du projet

a) A quoi sert ce chatbot ?

Ce chatbot est destiné à des utilisateurs qui se posent des questions en lien avec les discours d'investitures des derniers présidents français, de Valéry Giscard d'Estaing à Emmanuel Macron. Il permet tout simplement de faciliter l'obtention de réponses, en allant directement piocher les éléments de la réponse dans ces discours. Le but étant de formuler des réponses claires et précises afin d'aider les utilisateurs dans leurs recherches.

Il permet concrètement d'afficher :

- Les mots les moins importants des textes
- Les mots ayant le score TF-IDF le plus élevé
- Les mots les plus importants répétés par Chirac
- Les présidents qui ont le plus parlé de la Nation et celui qui l'a le plus répété
- Le premier président qui a parlé du climat
- Les mots que tous les présidents ont évoqués

b) Comment fonctionne le programme ?

Ce programme est basé sur un ensemble de fonctions mises bout à bout (telle que la fonction qui permet de nettoyer les textes, c'est-à-dire le prétraitement des données : def clean_speeches(files_names):).

```
def clean_speeches(files_names):
    for file in files_names:
        f = open("./speeches/" + file, "r", encoding="utf-8")
        lines = f.readlines()
        f.close()
        lines = [line.lower() for line in lines]
        if not os.path.exists("./cleaned"):
            os.makedirs("./cleaned")
        f = open("./cleaned/" + file, "w", encoding="utf-8")
        for line in lines:
            if line != "\n":
                for char in line:
                    if char == '.' or char == ',' or char == '!' or char == '?' or char == ';' or char == ':':
                        line = line.replace(char, "")
                    elif char == '-' or char == "'":
                        line = line.replace(char, " ")
                f.write(line)
        f.close()
```

Fonction qui permet de nettoyer les textes, pour que le programme lise mieux les textes

Le programme est surtout basé sur la création de la matrice TF-IDF. Cette matrice permet de calculer la fréquence à laquelle les mots d'un discours apparaissent et permet de leur attribuer un score. Si ce score est élevé, le mot est souvent présent, et si le score est faible, le mot est peu présent dans le texte. Le but de cette matrice est d'analyser de la même sorte les mots dans les questions et de chercher leurs score TF-IDF dans les textes. Une fois ce lien effectuer, le programme est capable de venir récupérer des éléments de réponse et d'en formuler une. La matrice TF-IDF est réellement le cœur de ce programme. Elle est ensuite entourée de beaucoup d'autres fonctions et boucles, qui permettent d'élaborer la meilleure réponse possible pour les utilisateurs.

```
def tf_idf(dir):
    idf_dict = idf(dir)
    tf_idf_dict = {}
    for file in list_of_files(dir, "txt"):
        tf_idf_dict[file] = {}
        f = open(dir + "/" + file, "r", encoding="utf-8")
        lines = f.readlines()
        for line in lines:
            for word in line.split():
                if word in tf_idf_dict[file]:
                    tf_idf_dict[file][word] += 1
                else:
                    tf_idf_dict[file][word] = 1
        for word in tf_idf_dict[file]:
            tf_idf_dict[file][word] = tf_idf_dict[file][word] * idf_dict[word]
    return tf_idf_dict
```

Matrice TF-IDF : calcul de la distribution de chaque mot dans les textes et les questions.

II. La présentation technique du projet

a) Quelles sont les principales fonctions du programme ?

Le programme est en partie composé de fonctions. Les principales fonctions du programme sont : bien sûr TF-IDF (présentée ci-dessus), Matrix_TF, Intersection et Menu. Voici leur construction :

```
def matrix_tf(question, directory):
    matrix = []
    corpus = []
    if directory[-1] != "/":
        directory += "/"
    for filename in os.listdir(directory):
        matrix.append([])
        corpus.append(open(directory + filename, "r", encoding="utf-8").read())
    question = clean_question(question)
    question = question.split(" ")
    for elem in question:
        if elem == '':
            question.remove(elem)
    for i in range(len(corpus)):
        corpus[i] = corpus[i].split(" ")
    cleaned_corpus = []
    for i in range(len(os.listdir(directory))):
        cleaned_corpus.append([])
        for j in range(len(corpus[i])):
            if cleaned_corpus[i].count(corpus[i][j]) == 0:
                cleaned_corpus[i].append(corpus[i][j])
    for i in range(len(os.listdir(directory))):
        matrix.append([])
        for j in range(len(corpus[i])):
            count = 0
            for k in range(len(question)):
                if corpus[i][j] == question[k]:
                    count += 1
            matrix[i].append(count)
    return matrix
```

La fonction Matrix_TF : permet de calculer le vecteur TF-IDF pour les termes de la question

```python
def intersection(question, directory):
    common_words = []
    corpus = ""
    if directory[-1] != "/":
        directory += "/"
    for filename in os.listdir(directory):
        corpus += (open(directory + filename, "r", encoding="utf-8").read())
    question = clean_question(question)
    question = question.split(" ")
    corpus = corpus.split(" ")
    for i in range(len(corpus)):
        for j in range(len(question)):
            if corpus[i] == question[j] and corpus[i] != '':
                common_words.append(corpus[i])
    return common_words
```

La fonction Intersection : identifie les termes de la question qui sont aussi présents dans le corpus

b) Justification des choix de structures

Pour la méthode TF-IDF et toutes les fonctions liées, nous avons utilisé des dictionnaires. Concernant, la création des matrices pour les mots de la question, les mots du corpus et leurs fréquences, nous avons notamment utilisé des listes 2D.

```python
# Méthode TF-IDF
def count_occurrence(files_names):
    dict = {}
    for file in files_names:
        f = open("./cleaned/" + file, "r", encoding="utf-8")
        lines = f.readlines()
        for line in lines:
            for word in line.split():
                if word in dict:
                    dict[word] += 1
                else:
                    dict[word] = 1
    return (dict)
```

Une fonction liée à la matrice TF-IDF, réalisée avec des dictionnaires

c) Quelles sont les difficultés que nous avons pu rencontrer ?

Au niveau de la partie 1, nous avons rencontré aucune difficulté spécifique. Cependant, pour la réalisation de la partie 2, les questions étaient plus complexes à comprendre et à exécuter. Par exemple, pour la question 3 (Calcul du vecteur TF-IDF pour les termes de la question), la question semblait plus compliquée que prévu mais finalement pour obtenir le résultat souhaité, il a juste fallu simplifier la question et cela a fonctionné. De plus, pour la question 6, les répertoires «speeches» et «cleaned» semblaient introuvables lorsque nous changions leur chemin, cependant ils étaient bien présents dans les fichiers, alors pour modifier la fonction, cela ne fonctionnait pas. Donc, le programme ne génère pas le bon discours de président pour répondre à la question.

III. Les résultats obtenus

```
Que voulez-vous faire ? Choisissez 1, 2 ou 3.
2
1. Afficher la liste des mots les moins importants
2. Afficher les mots ayant le score TD-IDF le plus élevé
3. Afficher les mots les plus répétés par le président Chirac
4. Afficher les noms des présidents qui ont parlé de la « Nation » et celui qui l'a répété le plus de fois
5. Afficher le premier président à parler du climat et/ou de l'écologie
6. Afficher les mots que tous les présidents ont évoqués (hormis les mots dits non importants)
Que voulez vous faire parmi ces 6 fonctionnalités ?
```

```
1.Utiliser les fonctions de base.
2.Utiliser les fonctionnalités avec TF-IDF.
3.Poser une question.
Que voulez-vous faire ? Choisissez 1, 2 ou 3.
3
Vous pouvez poser votre question.
Comment la France répond aux besoins de la nation tout en respectant la contrainte de l'écologie ?
```

```
Que voulez-vous faire ? Choisissez 1, 2 ou 3.
1
1.Afficher la liste des présidents
2.Afficher les prénoms des présidents associés à leurs noms


Que voulez vous faire ?
```

Pour conclure, nous pouvons dire que ce projet nous a permis de collaborer, d'en apprendre plus sur la programmation et surtout de partager nos connaissances. Tant sur le plan technique (python) que sur le plan logistique.

Sur le plan technique, nous avons appris à assembler des idées, des petits bouts de codes réalisés un à un. Il peut être difficile d'écrire un code en entier en une seule fois. Notre façon de procéder a donc été de d'abord résumer ce que l'on attendait de nous, quelle fonction était nécessaire, de quels outils on allait avoir besoin et enfin transcrire toutes ces idées en code au fur et à mesure. Le projet nous a également permis de plus manipuler les dictionnaires et les matrices

Sur le plan logistique nous avons beaucoup appris l'une de l'autre. Savoir s'organiser, se répartir les tâches, trouver des moments pour travailler et partager nos idées ensemble. Tomber d'accord sur une idée, lorsque nos avis divergeaient. Avoir deux profils différents dans une même équipe est très enrichissant, ça nous a permis de beaucoup progresser personnellement, notamment dans des secteurs où l'on pouvait avoir du mal auparavant.

De plus ce projet nous aura également permis de découvrir ou d'apprendre à mieux se servir d'un outil informatique professionnel : GitHub. Se servir d'outils comme celui-ci permet un gain de temps énorme en termes de partage et de collaboration, surtout quand nous avons une deadline de remise de projet.

Ce projet nous aura donc appris de nombreuses choses et nous sommes fières du travail réalisé et de notre progression.