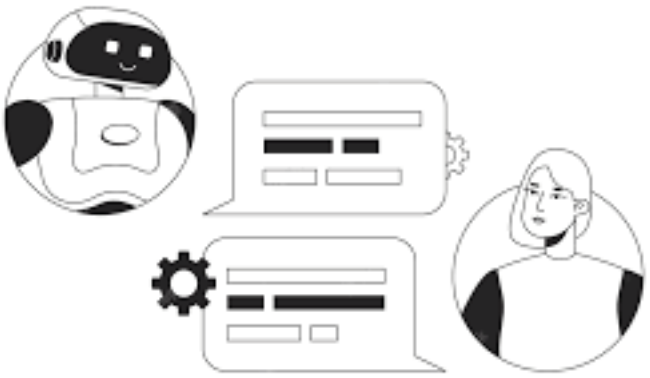
PROJET PYTHON MY FIRST CHATBOT



Années : 2023

Groupe F: Groupe de TP JIN et ROUSSELY

JIN Johnny et ROUSSELY Tom

Sommaire :

INTRODUCTION

Le projet "My First ChatBot" à l'Efrei propose aux étudiants une expérience d'apprentissage pratique axée sur la programmation en Python. Il vise à développer les compétences des étudiants en matière de manipulation de fichiers, de traitement de chaînes de caractères, et de gestion de données textuelles. L'objectif principal est de fournir une compréhension approfondie des concepts fondamentaux de la programmation tout en engageant les étudiants dans la création d'un ChatBot, une application permettant des interactions simples avec les utilisateurs.

À travers ce projet, les étudiants explorent des concepts tels que la tokenisation, qui consiste à diviser une chaîne de caractères en unités individuelles, ainsi que le traitement du langage naturel (NLP) appliqué à des tâches de base de compréhension textuelle. En construisant ce ChatBot, les étudiants acquièrent une expérience pratique dans la résolution de problèmes concrets liés à l'interaction homme-machine, consolidant ainsi leurs compétences en programmation et en manipulation de données. Ce projet constitue une étape cruciale dans l'apprentissage des fondamentaux de la programmation à l'Efrei.

PRESENTATION DU PROJET

Le projet réalisé est divisé en plusieurs partie.

Pour commencer, la Partie 1 permet de faire les fonctionnalités de base de notre CHATBOT, les fonctions créer dans la première Partie serviront plus tard dans la

Partie 2 de ce projet.

La Partie 2 consiste a construire le CHATBOT, c'est-à-dire de construire le système de questions réponse, tout cela grâce au fonctions de la Partie 1.

PRESENTATION TECHNIQUE

Partie 1 :

Les fonctions de bases :

- **Fonction doublon :**

```
def doublon():
    d = extraction()
    s=[]
    for mot in d :
        if mot not in s :
            s.append(mot)
    return s
```

- **Fonctionnalité : Élimine les doublons de la liste des noms des présidents obtenue à partir de extraction, fonction ci-dessus.**
- **Retour : Une liste de noms de présidents uniques.**

- **Fonction conversion en minuscule et suppression de la ponctuation :**

- **Fonctionnalité : Crée de nouveaux fichiers à partir des fichiers existants en convertissant les majuscules en minuscules, en supprimant les accents et en éliminant la ponctuation.**
- **Opération effectuée : Lecture depuis le répertoire source (speeches) et écriture dans un répertoire de destination (Cleaned).**
- **Détail : La suppression de la ponctuation se fais à l'aide d'un dictionnaire ou y est référencé tout type de ponctuation à supprimer. La conversion des textes en minuscules ce fait avec le code ASCII en ligne 10 et 11 du programme, de plus la conversion des lettres avec accent c'est fait a l'aide de la conditions IF. Ce choix a été fais, car c'était le plus simple.**

La méthode TF-IDF et Fonctionnalités a développer:

- **Fonction TF :**

```python
def tf(c):
    z = c.read()
    resultats = {}
    mot = z.split()
    for m in mot :
        if m in resultats :
            resultats[m]+= 1
        else :
            resultats[m] = 1
    return resultats
```

- **Fonctionnalité : Compte les occurrences de chaque mot dans un texte donné.**
- **Paramètres : c - objet fichier.**
- **Retour : Un dictionnaire avec les occurrences de chaque mot.**

- **Fonction IDF :**

```python
def Idf(d):
    idf_s={}
    for i in d.keys():
      for j in d[i]:
        if j in idf_s:
          idf_s[j]+=1
        else:
          idf_s[j]=1
    for a in idf_s.keys():
          idf_s[a]=math.log10((len(d)/idf_s[a]))
    return idf_s
```

- **Fonctionnalité : Calcule la fréquence inverse du document (IDF) pour chaque mot dans une collection de textes.**
- **Paramètres : d - un dictionnaire avec les occurrences de mots dans différents textes.**
- **Retour : Un dictionnaire avec les scores IDF pour chaque mot.**

- **Fonction TF_IDF :**

```python
def TF_IDF(A):
    l=Idf(A)
    Matrice={}
    cleaned = r'C:\pychatbot\Cleaned'
    for mot in l:
        Matrice[mot]=[]
        for file  in list_of_files("./cleaned",".txt" ) :
            with open(cleaned + "/" + file , "r")as f :
                d=tf(f)
            if mot in d:
                Matrice[mot].append(d[mot]*l[mot])
            else:
                Matrice[mot].append(0.0)
    return Matrice
```

- **Fonctionnalité : Crée une matrice TF-IDF pour les mots dans des fichiers spécifiques.**
- **Paramètres : A - un dictionnaire avec les occurrences de mots dans différents textes.**
- **Retour : Une matrice TF-IDF.**

- **Fonction des non important :**

```python
def nn_important(B):
    Liste_de_mot=TF_IDF(B)
    mot_pas_important=[]
    for g,i in Liste_de_mot.items():
            d=0
            for h in i:
                    if h==0.0:
                        d+=1
            if d == 8 :
                mot_pas_important.append(g)
    return mot_pas_important
```

- **Fonctionnalités : Retourne une liste de mots considérés comme peu importants en fonction de leur score IDF.**
- **Paramètres : B - un dictionnaire avec les occurrences de mots dans différents textes.**
- **Retour : Une liste de mots peu importants**
-

- **Fonction score plus élevé**

```python
def m_score_plus_elevé(H):
        Liste_de_mot = TF_IDF(H)
        mot = None
        scoremax = 0.0
        for mots, score in Liste_de_mot.items():
            for d in score :
                if d > scoremax:
                    scoremax = d
                    mot = mots
                elif d == scoremax:
                    mot = mots
        return mot
```

- **Fonctionnalités : Retourne le mot ayant le score IDF le plus élevé à partir d'une matrice TF-IDF.**
- **Paramètres : H - un dictionnaire avec les occurrences de mots dans différents textes.**
- **Retour : Le mot avec le score IDF le plus élevé.**

- **Fonction mot plus répétés par M. Chirac :**

```python
def Mot_plus_répétés_par_chirac():
    a = r'C:\pychatbot\Cleaned\Nomination_Chirac1.txt'
    b=r'C:\pychatbot\Cleaned\Nomination_Chirac2.txt'
    with open(a, 'r') as f , open(b,'r') as f1 :
        d=tf(f)
        s=tf(f1)
        q={}
        for g,i in d.items():
            for c,v in s.items():
                if g == c :
                    q[g]= i+v
        mot = None
        scoremax = 0
        for mots, score in q.items():
                if score > scoremax:
                    scoremax = score
                    mot = mots
        return mot
```

- **Fonctionnalités : Retourne le mot le plus fréquemment utilisé par Chirac dans deux fichiers spécifiques.**
- **Retour : Le mot le plus fréquents**

- **Détails : Utilisation de la fonction tf (occurrence des mots). Nous avons utilisé un dictionnaire pour renvoyer les keys = nombre de fois ou le mots apparais et les items qui sont les mots**

- **Fonction « écolo »**

```
def ecolo():
    a = r'C:\pychatbot\Cleaned'
    d = []
    for nom_f in os.listdir(a):
        f_entree = os.path.join(a, nom_f)
        with open(f_entree, 'r') as f:
            z = f.read()
            h = z.split()
            for c in h:
                if c == 'climat' or c == 'ecologie' or c == 'ecologique' or c == 'climatique':
                    d.append(nom_f)
    g = ['1', '2', '3', '4', '5', '6', '7', '8', '9']

    for i in range(len(d)):
        d[i] = d[i][11:-4]
        for j in range(len(d[i])):
            if d[i][j] in g:
                d[i] = d[i][:-1]

    doublon = []
    for double in d:
        if double not in doublon:
            doublon.append(double)

    return doublon
```

- **Fonctionnalités : Retourne une liste de noms de présidents ayant parlé d'écologie.**
- **Retour : Une liste de noms de présidents.**
- **Détails : Cette fonction va parcourir chaque fichier texte « cleaned » pour trouver des mots en rapport avec l'écologie et si un termes est trouver alors il ajoute le nom du fichier a la liste d. Les noms des fichier sont ensuite nettoyer et on élimine les doublons**

Partie 2 :

- **Fonction tokenisation de la question :**

```
def Tokenisation_de_la_Q(question):
    Question=[]
    Question_nettoye=''
    for ponctuation in question :
        if ponctuation in [',','.','?',';',':','!']:
            ponctuation=''
        elif ponctuation in ["'","-"]:
            ponctuation= ' '
        else :
            ponctuation=minuscule(ponctuation)
        Question_nettoye+=ponctuation
    h = Question_nettoye.split()
    for i in h :
            Question.append(i)
    return Question
```

- **Fonctionnalités : Tokenise une question en demandant à l'utilisateur de la saisir.**
- **Valeur de Retour : Une liste de mots dans la question.**
- **Details : Nettoyage de la questions de la même manière que la fonctions conversion en minuscule.**

- **Fonction « mot dans leur corpus » :**

Fonctionnalités: Trouver les mots présents dans un ensemble spécifié de mots dans les fichiers d'un répertoire donné.
Paramètre :
listedemot : Une liste de mots à rechercher dans les fichiers.
Retour : Une liste des mots trouvés dans les fichiers du répertoire spécifié.

- **Fonction calcul de vecteur :**

```
def calcul_de_vecteur(motdanslecorpus,A):
    resultats={}
    matrice={}
    d = mot_dans_le_corpus(motdanslecorpus)
    for m in motdanslecorpus:
        if m in resultats and m in d:
            resultats[m] += 1
        elif m not in resultats and m in d:
            resultats[m] = 1
    a=TF_IDF(A)
    for mot in resultats:
        matrice[mot]=[]
        if mot in matrice:
                matrice[mot].append(resultats[mot]*a[mot])
        else:
                matrice[mot].append(0.0)

    return matrice
```

Objectif de la fonction :

L'objectif de la fonction calcul_de_vecteur est de calculer un vecteur représentant la fréquence des mots spécifiés dans une liste (motdanslecorpus) et d'ajuster ces fréquences en utilisant le score TF-IDF calculé à partir d'un corpus de textes (A).
Paramètres de la fonction :

motdanslecorpus : Une liste de mots dont nous voulons calculer la fréquence et le score TF-IDF.
A : Le corpus de textes à partir duquel la matrice TF-IDF sera calculée.
Retour de la fonction :

La fonction retourne une matrice où chaque mot de la liste motdanslecorpus est associé à un vecteur représentant sa fréquence ajustée par le score TF-IDF.
Détails supplémentaires :

La fonction utilise la fonction auxiliaire mot_dans_le_corpus pour récupérer la liste des mots présents dans le corpus.
Pour chaque mot dans la liste motdanslecorpus, la fonction calcule la fréquence d'apparition dans le corpus et ajuste cette fréquence en utilisant le score TF-IDF correspondant à ce mot.
Le résultat final est une matrice où chaque ligne représente un mot et chaque colonne représente un fichier du corpus, avec les valeurs correspondant à la fréquence ajustée par le score TF-IDF.

CONCLUSION

En conclusion, le projet proposé se concentre sur l'analyse de texte pour développer un système de chatbot basé sur la fréquence des mots dans un corpus. L'algorithme implique le prétraitement des données, la création d'une matrice TF-IDF, la représentation des questions par des vecteurs TF-IDF, le calcul de similarité avec les mots du corpus, la sélection des mots similaires et enfin la détermination de la meilleure réponse en fonction de ces mots. Cette approche offre une méthode efficace pour générer des réponses intelligentes en se basant sur la compréhension du contenu des documents du corpus, sans nécessiter la manipulation de réseaux de neurones.