Python project
My first chatBot

Instructions & general information :

⇒ This project is to be done exclusively in Python.

⇒ Project attachments :
- Presidents' speeches files

⇒ Team organization :
- This project is to be carried out in pairs (**only one trinomial is allowed** if an **odd number** of students).
- The list of teams is to be given to teachers at the end of the first project follow-up session at the **latest.**

⇒ Key dates :
- Publication date: **04/11/2023**
- Follow-up session 1: Week of **13/11/2023**
- First filing date: **11/26/2023** at **11:59 p.m.**
- Follow-up session 2: Week of **27/11/2023**
- Final deposit date: **12/17/2023** at **11:59 p.m.**
- Date of defense: Week of **18/12/2023**

⇒ Final deposit:
- The project will be submitted using Moodle
- Project code containing **.py** and **.txt** files
- The **.pdf** report
- A **README.txt** file listing the programs and how to use them in practice. This file must contain all the instructions necessary for execution; it is very important to explain to the user how to use the tool.

⇒ Evaluation
- Detailed scale: to be provided later
- Final project grade = Code grade + Report grade + Defense grade
- Reminder: project grade = 20% of "Python Programming" course grade
- Members of the same team can receive different marks depending on the effort they put into the project.

⇒ **Collaborative working**
- Project team work will be tracked via Git
- The Git tutorial will be online the week of **06/11/2023**

Code organization :

⇒ The rating of the code will mainly take into account :
- Implementing requested functionalities: moving forward as best you can, but NOT off-topic
- The quality of the code supplied: organization into functions, **comments**, meaningful variable names, respect for file names.
- Ergonomics: Ease of use of the user interface

Teaching concepts covered :

⇒ Completing this project will help you apply the following pedagogical concepts:
- Basic concepts, 1D lists, 2D lists, functions, files and collections

⇒ To help you realize this project, you can rely on :
- TI101 course materials (I, B, P)
- Books, various courses on the web. **ALERT PLAGIAT!!** it's not about copying entire programs.
- **Efrei** teachers during project follow-up sessions

Description

This project is about text analysis. This project will allow you to understand some basic concepts used in text processing and help you understand one of the methods used in developing chatbots and/or generative artificial intelligences such as ChatGPT.

Obviously, we're not talking here about manipulating neural networks, but in this project we're going to focus on a method based on words occurrences to generate intelligent answers from a corpus of texts. The aim is to design a system that can answer questions based on the frequency of words in the corpus.

This application is based on the following algorithm:

Data pre-processing: your program begins by collecting and pre-processing a set of documents in order to understand the nature of their contents, before using them to prepare answers. This phase cleans up the text by removing punctuation, converting letters to lower case, and dividing the text into words (or "tokens").

Creating a TF-IDF matrix: For each unique word in the documents, you'll need to calculate a **TF-IDF** vector using the **TF-IDF** method. Each word is associated with a vector whose dimension is equal to the number of documents in the corpus. This creates a **TF-IDF** matrix where each row represents a word and each column represents a document.

Question representation: When a question is asked, the chatbot performs the same pre-processing on the question. It then calculates a TF-IDF vector for this question, using the same vocabulary as the documents. The question vector has the same dimension as the vectors associated with the words in the corpus.

Similarity calculation: The chatbot calculates the similarity between the question vector and the word vectors in the corpus, using cosine similarity or another similarity measure. This enables it to determine which words in the corpus are most similar to the question.

Selecting the best answer: The chatbot identifies the words in the corpus most similar to the question, based on their **TF-IDF** similarity score. It then selects the answer that contains the greatest number of these similar words.

Provide answer: The chatbot returns the selected answer as the answer to the question asked.

Required work

To guide you in the realization of this application, we will divide the work in 3 parts:

1. **Part I**: Development of basic functions for understanding the content of the given files.
2. **Part II**: Calculation of the similarity matrix and generation of automatic responses.
3. **Part III**: Generalizing the application to cover various themes.

Part I

File database

In this project, 8 text files are provided in a directory called "speeches". These files represent speeches by 6 French presidents at their nominations.

The name of each file is in the following format: **Nomination_[president's name][number].txt** where :

- **[name of a president]**: Is one of the following names: Chirac, Giscrad d'Estaing, Mitterrand, Macron, Sarkozy
- **[number]**: Is an optional field representing a sequential number. It appears in the names of files where the same president has given several speeches.

Basic functions

In order to analyze the contents of these files, we first ask you to develop the following functions:

- Extract the names of the presidents from the names of the text files provided ;
- Associate a first name with each president;
- Display the list of president's names (watch out for duplicates) ;
- Convert the texts in the 8 files to lower case and store the contents in new files. The new files are to be stored in a new folder called "cleaned". This folder should be located in the main directory of the ***main.py*** program, and at the same level as the "speeches" directory.
- For each file stored in the "cleaned" directory, run through its text and remove any punctuation characters. The final result should be a file with words separated by spaces. Please note that some characters, such as the apostrophe (') or the dash (-), requires special treatment to avoid concatenating two words (e.g. "elle-même" should become "elle même" and not "ellemême"). Changes made at this phase should be stored in the same files in the "cleaned" directory.

Notes :

1. These functions can be implemented using the predefined modules covered in the course, as well as the "os" module.
2. The Python code used to run through the list of files with a given extension and in a given directory is as follows:

```
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
```

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

The TF-IDF method

In text processing, it is often useful to represent words as numerical vectors to facilitate information retrieval, document classification and/or text analysis.

One frequency-based method for generating numerical vectors for words is called **TF-IDF** (Term Frequency-Inverse Document Frequency). Here's how it works:

TF (Term Frequency): The first part of the **TF-IDF** measures how often a term (a word in our case) appears in a specific document. The more frequently a term appears in a document, the higher its **TF** score for that document ➔ For each word in each file, you are asked to count the number of times that word appears. To do this, we need at least one function that calculates the number of occurrences of each word in a text.

- Write a function that takes a string as a parameter and returns a dictionary associating with each word the number of times it appears in the string.

IDF (Inverse Document Frequency) : The second part of the **TF-IDF** measures the importance of a term in the entire corpus of documents. Terms that are very frequent in many documents will have a lower **IDF** score, while terms that are rare will have a higher **IDF** score. ➔ For each word, calculate the **logarithm of the** inverse of the proportion of documents in the corpus that contain that word. This gives more weight to rare words and reduces the weight of common words.

- Write a function that takes the directory (where all the files in the corpus are located) as a parameter and returns a dictionary associating the **IDF** score with each word.

Note:

It is allowed to use the "math" module to call its predefined function calculating the logarithm.

Final score: The **TF-IDF** score of a word in a document is obtained by multiplying the **TF** score by the **IDF** score.

- TF-IDF = TF * IDF.

Thus, the **TF-IDF** score of a word in a given document is a numerical vector that reflects both the frequency of the word in that document and its relative importance in relation to the corpus as a whole.

To generate numerical vectors for words, each word in the document corpus is assigned a vector of **TF-IDF** scores. These numerical vectors represent the distribution of each word throughout the corpus. Together,

these vectors form a matrix called a "**TF-IDF matrix**", where each row represents a word and each column represents a document.

- Write a function that takes the directory where the files to be analyzed are located as a parameter and returns at least the TF-IDF matrix.
- The number of rows in the resulting matrix must be equal to the number of unique words in all the files, and the number of columns must be equal to the number of files in the directory ➔ If necessary, write a function that calculates the transpose of a matrix (**Warning!!** do not use a predefined function).

Features to be developed

Based on the above functions, write programs that allow you to :

1. Display the list of least important words in the document corpus. A word is said to be unimportant if its **TD-IDF** = 0 in all files.
2. Display the word(s) with the highest **TD-IDF** score
3. Indicate the most repeated word(s) by President Chirac
4. Indicate the name(s) of the president(s) who spoke of the "Nation" and the one who repeated it the most times.
5. Identify the first president to talk about climate (“climat”) and/or ecology (“écologie”)
6. Excepti the so-called "unimportant" words, which word(s) did all the president mention?

Main program

The aim here is to write a temporary main program in which you are asked to :

- Propose a menu for the user
- Access previous functions according to user request

To remember:

⇒ The first filing must be made on **26/11/2023** at **11:59 p.m.**
⇒ Instructions for this filing will be provided the week of 06/11/2023

Part II

If you see this message, it means that you have read everything, Bravo ! Keep going on this part to get the next one ☺