pychatbot-LaunaySchapman-Mouries-int3

2023-2024

Python ChatBot

Mouriès Pierre

Launay Schapman Alexis

Table des matières

Introduction

In this project our purpose is to do a “ChatBot”, like a cheap version of ChatGPT.

To lead us to our goal we have some text documents containing speeches of French presidents, forming a corpus. Moreover, with our program we are allowed to calculate which word is important or not (we see it more in details after).

Furthermore, you can use some pre-define functions in our menu to know some precise information.

Then a question can be asked by the user and by some calculations and process a relation between the question and our little data base can be rise. Thus, the ChatBot answer the question with appropriate words.

Technical presentation

Our project is made with python 3.12.

In our program we are capable to clean texts, calculate the occurrence of the words, in addition we can also compute the IDF of a word, with the formula below:

$$IDF\,(word) = log_{10}\left(\frac{Total\ number\ of\ documents\ in\ the\ corpus}{Number\ of\ documents\ in\ the\ corpus\ containing\ the\ word}\right)$$

This brings us to the calculation of the TF-IDF vectors which represent the importance of a word. Then those scores are putted in a matrix, where rows and column are respectively represented by the words and the documents. For instance, if we have 3 documents and 476 words, we have 3 columns and 476 rows.

Furthermore, we can tokenize a question ask by the user the question also got a TF-IDF score, next all the terms in the question are compare with the terms in the corpus it’s looks like an intersection between the words in the question and the words of the corpus. To know which

word comes next, we use some similarity based on a quotient of sums.

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\cdot\sqrt{\sum_{i=1}^{n} B_i^2}}$$

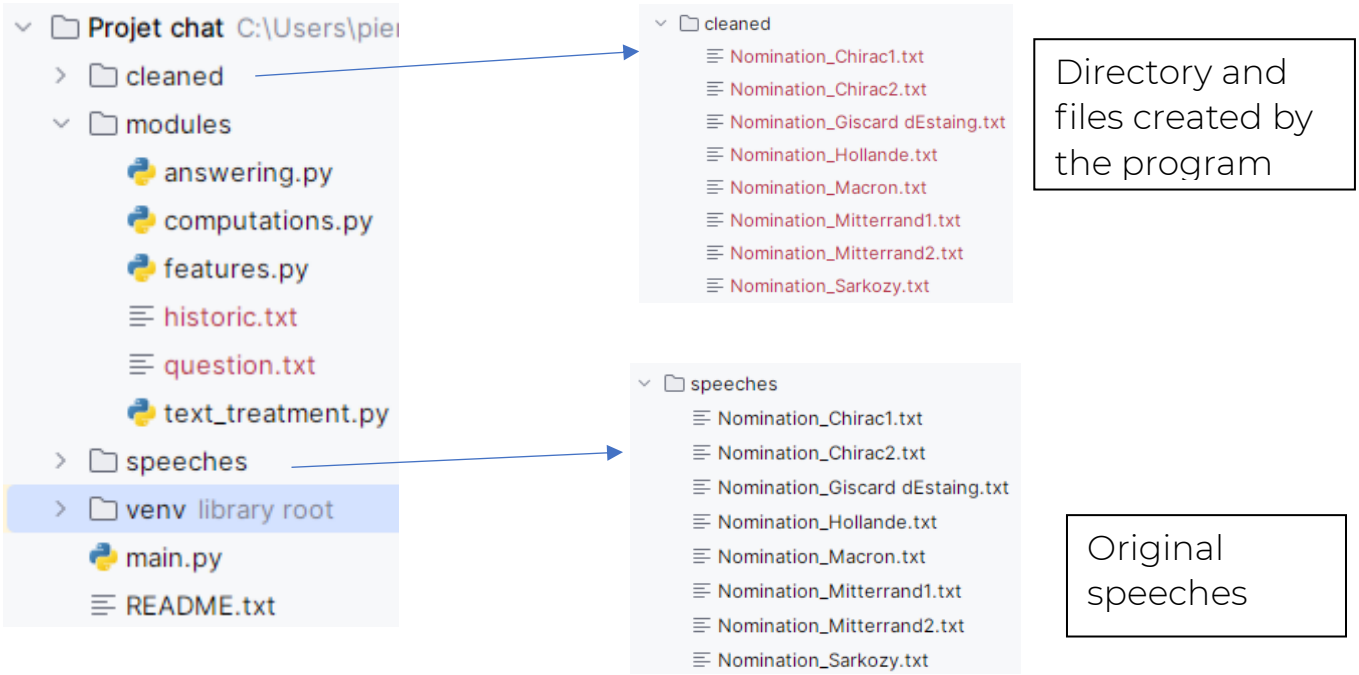
For that we use the product scalar following by the norm of the vector and the calculation of the similarity.

Finally, our ChatBot can answer our question by a generation for that it takes the higher TF-IDF of the question and generate a question from this.

To realize all of that we use algorithms shared in different files: main.py, text_treatment.py, features.py, computations.py, answering.py and some other text files.

Functional presentation

The file organization of our project



Computations.py

- *TF_creating* is to count the occurrence of the words in the corpus. And return a dictionary of the word in key and its occurrence in value.
- *IDF_creating* is to calculate the IDF of each word. And return a dictionary of the word in key and its IDF in value.
- *TF-IDF_creating* we use the previous function to finally calculate the TF-IDF. And its return a matrix of all TF-IDF, the list of all words and the files names.

We must notice that we use some global variables:

```
# computing useful globals variables
IDF = IDF_creating('./cleaned/')
TF_IDF_MATRIX, WORDS, FILES = TF_IDF_creating("./cleaned/")
NAMES = tx.associating_names(FILES)
```

features.py

Global variables

```
### Global Variables importation
# importation of IDF matrix from computations.py and associated row index, column index
TF_IDF_MATRIX, WORDS, FILES = cmp.TF_IDF_MATRIX, cmp.WORDS, cmp.FILES
# importation of the associated names (list with duplication)
names = cmp.NAMES
```

Those functions are all the functions used in the console.

- *Least* calculates the less important word.
- *finding_highest_tfidf* is to calculate the more important word.
- *finding_highest_tf_txtname* calculates the word that Chirac said thes most time.
- *nation* gives the name(s) of the president(s) who spoke about nation.
- *climat* said which president spoke about climate the first.
- *all_said* return the words that all presidents say.

Text_treatment.py

- *list_of_files* to do the list of the files we have and return it.
- *finding_names* take the list of file's names and return the president's names without duplication.
- *associating_names* same as finding _names but with duplication and in the same order than the list entered.
- *print_list* print a list properly, with a comma between each element and a point at the end.
- *new_folder* create a new folder, do nothing if the folder already exists.
- *file_in_lowercase* this function put all the letter which are in uppercase into the version in lowercase. Return the string in lowercase.
- *write_in_file*: write in a file. No return.
- *read_ str_in_file* reads the file and return what is reading.
- *file_cleaning* remove all the types of punctuations and return the string.
- *corpus_cleaning* uses all the previous functions to apply it on several texts directly. No return.

Answering.py

Global Variables

```
### Global Variables importation
# importation of IDF matrix from computations.py and associated row index, column index
TF_IDF_MATRIX, WORDS, FILES = cmp.TF_IDF_MATRIX, cmp.WORDS, cmp.FILES
# importation of the associated names (list with duplication)
names = cmp.NAMES
# importation of the IDF dictionnary of the corpus
IDF_corpus = cmp.IDF
```

- *tokenization* is to return the cleaned list of the words of the question.
- *intersection_question_corpus* is for return the words in common between the question and the corpus.
- *TF_IDF_vector_question* returns the TF-IDF of the question.
- *scalar_product* do the scalar product of 2 lists and return it.
- *vector_norm* calculate the norm of a list and return it.

- *similarity_calculating* does a similarity calculation by doing the product scalar over the norm of a list and the norm of another.
- *index_of_maxi* return the index of the highest value.
- *most_revelant* calculate and return the best text corresponding to the question.
- *I_CAN_ANSWER_WHAT_YOU_WANT* is to add word per word to form an answer to the question and return the answer to the question.
- *refine_answer* answers the question politely.
- *print_historic* write in a file each question and its answer. No return.
- *clean_historic* clear the file of the historic. Return “Done”

Presentation of our menu

Here is the first page when we run the program.

First let’s see what is going on if we enter “*f*”?

```
f
--------------------------------------------------------------------
Features:
Enter a number of the list:
Option :
     1) The list of leasts important words.
     2) The list of more important words (highest TD-IDF)
     3) The most repeated word(s) by President Chirac
     4) The name(s) of the president(s) who spoke of the 'Nation' and the one who repeated it the most times
     5) The first president to talk about climate ("climat") and/or ecology ("écologie")
     6) Word(s) which all the president mention (excepti the so-called 'unimportant' words)
     7) Main menu
```

Here you can choose a number, I let you see what it does by yourself!

Now, we can see what's happening when we use "*c*"?

```
c
-----------------------------------
Features:
You can ask a question or:
     - h for your historic,
     - m for the main menu
     - c for clean your historic
```

So, you can ask your question to our ChatBot and see the result.

There is an example at the end 😉.

Feedback

Alexis: I feel that the project was a good idea I learned a lot of things like how to use a match/case, how to deal with several files and a big number of functions. I learned how to collaborate and work on the same project. I acquired to use Git. But unfortunately, there were some incorrect things sometimes and for the part 2 we have only 1 week to do it. However, I'm happy because I feel not very strong in Python, but I learned skills and tips with my very good teammates who know how to share knowledge and be patient.

Pierre: In this project I have learned how to use GIT in order to collaborate with my teammate. This was useful and not too complicated. We finale arrive to deal with branch! The team organization of the project in our team was good despite the difference of level between the two of us. I'm just unsatisfied because we hadn't a lot of time and I would have liked to improve the project.

Our chatbot can give better answers than ChatGPT!! This is thanked to the telepathy module implemented in our bot.