MY FIRST CHAT BOT :PYTHON PROJECT

MERMONT--DOMINGUEZ SOPHIE
YOUSSAME MBILI JOY
L1INT4-2023

efrei
PARIS PANTHÉON-ASSAS UNIVERSITÉ

--Table of contents—

--Introduction to the projet--

This project explores text analysis concepts for creating a question-answering system. It employs a method based on word frequency to generate intelligent responses from a text corpus. The algorithm involves data pre-processing, where documents are cleaned and tokenised. A TF-IDF matrix is constructed, associating each word with a vector reflecting its occurrence across documents. When a question is asked, the chatbot calculates a TF-IDF vector and assesses its similarity to word vectors in the corpus. The system then selects the best answer based on TF-IDF similarity scores, considering words most similar to the question. Ultimately, the chatbot provides the selected answer as the response. The focus is on designing a system adept at answering questions by leveraging word frequency in the corpus.

--Functional Presentation--

1.extract_president_names(file_names):Returns a list of the names of each president from the speech file names.

2.clean_text(path, speech): Converts the speeches to lowercase and removes punctuation.

3.count_word_occurrences(text):Counts the number of times a word is present in a text.

4.calculate_tf(path):Returns the term frequency of each word in a text.

5.in_doc(directory): Returns a dictionary with the number of speeches each word is present in, used in calculate_idf().

6.calculate_idf(path): Returns a dictionary with the IDF score of each word for a given speech.

7.calculate_tf_idf_matrix_all(directory): Returns a TF-IDF matrix for all speeches in the directory.

8.display_least_important_words(tf_idf_matrix): Displays the list of least important words.

9.display_highest_tfidf_words(tf_idf_matrix): Displays word(s) with the highest TF-IDF score.

10.calculate_tf_idf_matrix_with_presidents(directory, president): Returns a TF-IDF matrix of each speech for a given president in a given directory.

11.most_repeated_words_by_president(directory, president): Returns the most repeated word said by a given president.

12.word_frequence_comparison(directory, target_word): Returns a list of who says the target word more in its speeches.

13.calculate_tf_idf_matrix_with_target_word(directory, president, target_word): Returns a TF-IDF matrix of each speech for a given president with respect to a target word.

14.president_speaking_about_nation_the_most(tf_idf_matrix, president_name): Indicates the most repeated word(s) by a given president.

15.word_frequence_comparison(directory, target_word): Returns a list of who says the target word more in its speeches.

16.calculate_tf_idf_matrix_with_first_mention(directory, president, target_words): Returns a TF-IDF matrix with the first mention of target words.

17.first_president_to_mention_climate_or_ecology(tf_idf_matrix, target_words): Identifies the first president to talk about 'Climate' and/or 'Ecology'.

18.words_mentioned_by_all_presidents(tf_idf_matrix, unimportant_words): Returns words mentioned by all presidents, excluding unimportant words.

19. main program:that provides a menu to interact with different functionalities.

20.question_token(question): Tokenizes a question, returning a cleaned version as a list of words.

21.allcorpus(directory): Returns all the speeches as a single string.

22.tokens_in_corpus(token_list, directory): Returns a list of tokens that are in the corpus.

23.tf_idf_matrix_fix_separated_by_document(directory): Returns a TF-IDF matrix that depends on each independent document.

--Technical presentation—

1. Data Pre-processing:

--Text cleaning involves removing punctuation and converting letters to lowercase.

--The text is split into words or tokens.

This step ensures a standardized format for subsequent analysis.

2. Creating a TF-IDF Matrix:

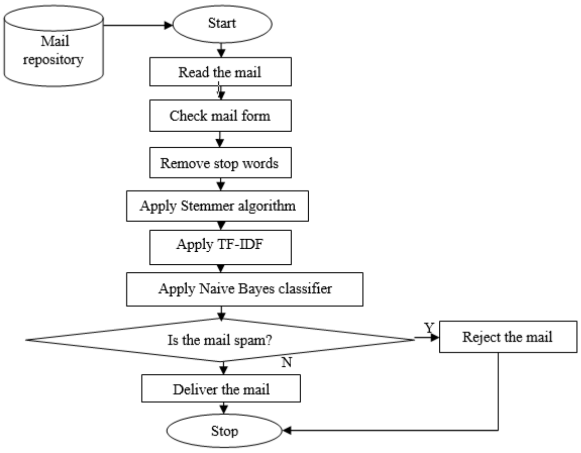
--For each unique word, calculate Term Frequency (TF) and Inverse Document Frequency (IDF).

--Create a TF-IDF vector for each word, resulting in a matrix where rows represent words, and columns represent documents.

3. Question Representation:

--Pre-process the user's question similarly to the data, ensuring consistency.

--Calculate a TF-IDF vector for the question using the same vocabulary as the documents.



←(example)

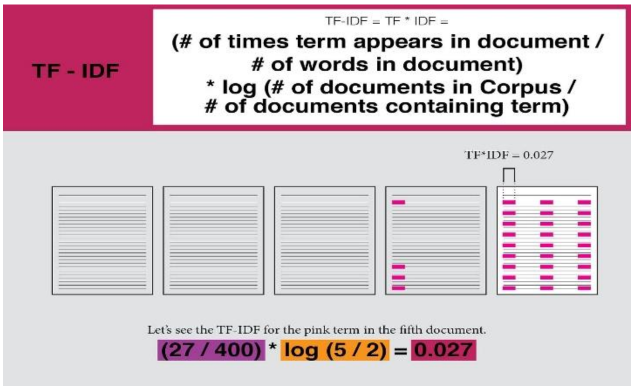
4. Similarity Calculation:

--Use cosine similarity or another measure to quantify the similarity between the question vector and word vectors in the corpus.(This step identifies relevant words in the corpus.)

5. Selecting the Best Answer:

--Based on TF-IDF similarity scores, determine words most similar to the question.

--Select the answer with the highest frequency of these similar words.



TF - IDF

TF-IDF = TF * IDF =
(# of times term appears in document /
# of words in document)
* log (# of documents in Corpus /
# of documents containing term)

Let's see the TF-IDF for the pink term in the fifth document.

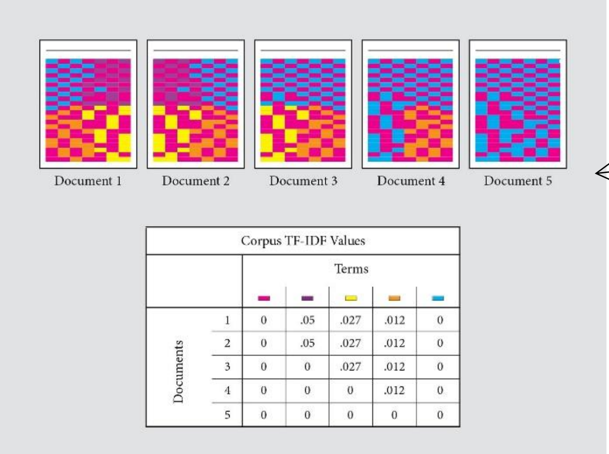
(27 / 400) * log (5 / 2) = 0.027

6. Text Analysis and TF-IDF Calculation:

--Calculate TF for each word's frequency in a document.

--Determine IDF, measuring a word's importance across multiple documents.

--Combine TF and IDF to obtain the TF-IDF score for each word in each document.



Corpus TF-IDF Values

		Terms				
		🟥	🟪	🟨	🟧	🟦
Documents	1	0	.05	.027	.012	0
	2	0	.05	.027	.012	0
	3	0	0	.027	.012	0
	4	0	0	0	.012	0
	5	0	0	0	0	0

TF-IDF : A visual explainer and Python

Implementation on Presidential Inauguration

Speeches

Difficulties Faced and solutions created:

=We had a dictionary system for the tf-idf.

- We remade the tf-idf matrix as a proper matrix.

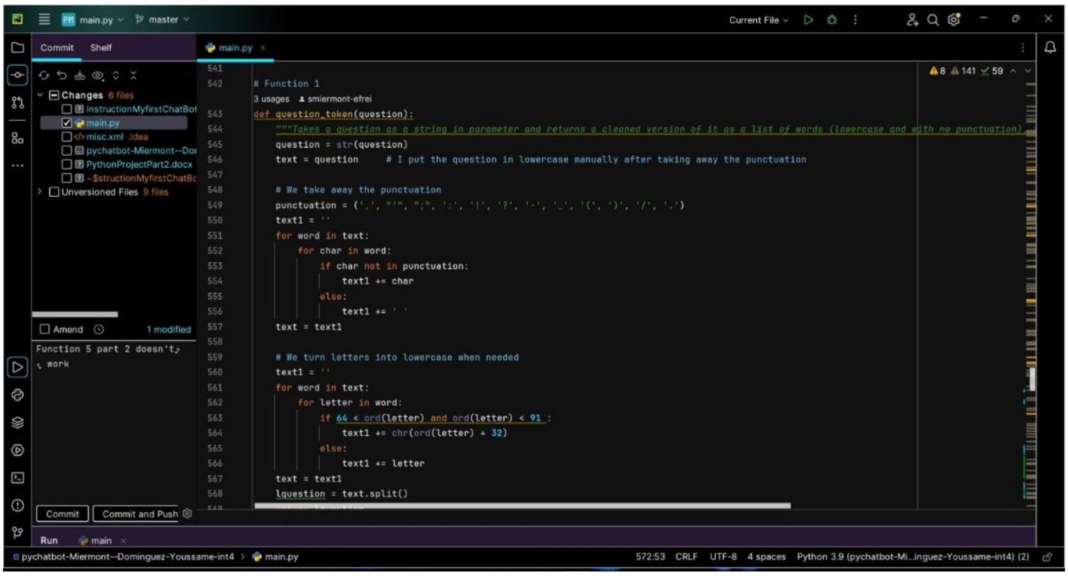
=We had functions that didn't work.

- We made updated versions of them but now we have multiple function that do the same thing but some of them don't work.

=We had a problem with our commit and the one team member couldn't push all the functions that they made.

-We created a word document "CommitsThatJoyYoussameHasDone" allowing the other member to make their own function and all we need to do is Crtl+C on PyCharm.

Results presentation:

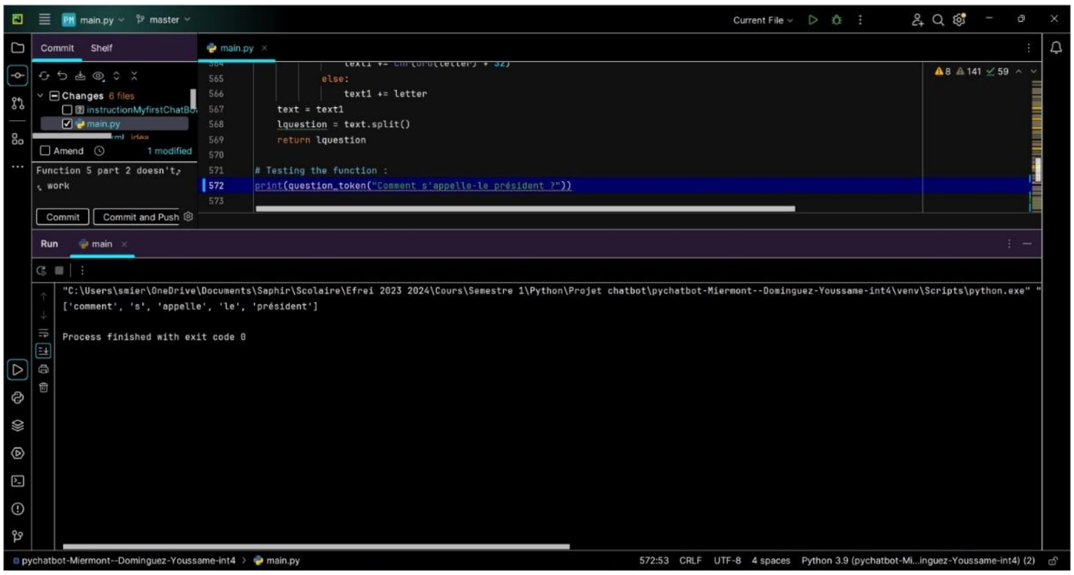


```python
# Function 1
def question_token(question):
    """Takes a question as a string in parameter and returns a cleaned version of it as a list of words (lowercase and with no punctuation)
    question = str(question)
    text = question     # I put the question in lowercase manually after taking away the punctuation

    # We take away the punctuation
    punctuation = (',', "'", ";", ':', '!', '?', '-', '_', '(', ')', '/', '.')
    text1 = ''
    for word in text:
        for char in word:
            if char not in punctuation:
                text1 += char
            else:
                text1 += ' '
    text = text1

    # We turn letters into lowercase when needed
    text1 = ''
    for word in text:
        for letter in word:
            if 64 < ord(letter) and ord(letter) < 91 :
                text1 += chr(ord(letter) + 32)
            else:
                text1 += letter
    text = text1
    lquestion = text.split()
```

Function 5 part 2 doesn't work



```python
            else:
                text1 += letter
    text = text1
    lquestion = text.split()
    return lquestion

# Testing the function :
print(question_token("Comment s'appelle-le président ?"))
```

```
"C:\Users\smier\OneDrive\Documents\Saphir\Scolaire\Efrei 2023 2024\Cours\Semestre 1\Python\Projet chatbot\pychatbot-Miermont--Dominguez-Youssame-int4\venv\Scripts\python.exe" "
['comment', 's', 'appelle', 'le', 'président']

Process finished with exit code 0
```

Function 5 part 2 doesn't work

main.py
master
Current File
Commit
Shelf
Changes 6 files
instructionMyfirstChatBot
main.py
Amend
1 modified
Commit and Push
Run
main
8 141 59
pychatbot-Miermont--Dominguez-Youssame-int4 > main.py
572:53 CRLF UTF-8 4 spaces Python 3.9 (pychatbot-Mi...inguez-Youssame-int4) (2)

--Conclusion--

The project has been a valuable learning experience on multiple fronts. From a technical perspective, we have deepened our understanding of text processing, TF-IDF algorithms, and practical applications of natural language processing. Collaborating as a duo highlighted the importance of effective communication and task coordination. Time management skills were very important while, problem-solving became a daily practice as we encountered and overcame technical challenges. Feedback integration played a crucial role in refining the project and improving team performance. Overall, the project provided a comprehensive platform for technical development.