QUESTIONNAIRE AND NOTES GENERATOR FOR VIDEO LECTURES

GROUP NO- 50

PIYUSH VERMA - 2020097
MOHAMMAD AMAAN KHAN - 2020388
NIKITA KAPOOR - 2020531
SHUBHAM SAHNI - 2020132
MEGHNA - 2020080

PROBLEM STATEMENT

YouTube has emerged as a widely used platform for educational content, but students often encounter challenges in comprehending, recalling, and retaining information presented in video lectures. This problem is due to the excessive amount of information provided and the absence of interactive features. There is no way for the student to know whether he has absorbed the video's content or not. Hence the student's learning is not optimised.

SOLUTION STATEMENT

We have developed a tool that assists students in creating text summaries from YouTube videos they watch. This text summarizer condenses the content of the video into a shorter form that can be easily understood by students. Furthermore, it provides a transcript and questions based on the summarized text. To complement this, we have also developed a questionnaire generator that further helps the student in self assessment.

SCOPE OF THE PROBLEM

The tool has been designed to assist them in generating notes and quiz questions for recorded video lectures, which can be accessed on any internet-connected device. With the use of natural language processing and machine learning algorithms, the tool can identify crucial concepts from the lectures and create questions based on them. We anticipate that this tool will provide students with a hassle-free and effective way to study and prepare for their exams. Additionally, this tool will benefit academic YouTube channels by boosting their viewership

TEXT SUMMARIZATION

The code defines a function called **extract_info**, which takes in four parameters and aims to extract information from the input text using natural language processing techniques. The function breaks down the input text into smaller chunks and processes each chunk using SpaCy's pre-trained language model to extract named entities and keypoints. The function uses multi-threading to process text chunks in parallel to expedite processing time. The function extracts additional speaker and topic information from the entire text and returns the extracted information as a dictionary with keys for "dates," "times," "locations," "speaker," "topic," and "keypoints."

QUESTION GENERATION

The QuestionExtractor class is a program that uses natural language processing (NLP) techniques and machine learning models to extract questions from a given document. The program works in four main steps: First, it identifies candidate entities by using spaCy's NER tagger, which tags named entities such as people, organizations, and locations. Second, it assigns tf-idf scores to each word in the document using scikit-learn's TfidfVectorizer module. The tf-idf score is a measure of how important a word is in a document relative to the entire corpus of documents. Third, the program ranks the keywords based on their tf-idf scores, with the highest scoring keywords considered the most important. Finally, the program constructs questions by replacing the highest scoring keywords with a blank space in a sentence containing the keyword. The resulting questions are returned as a dictionary with associated numbers and a placeholder for the answer. We have also considered the use of API calls to to enhance the function of our project.

NOVELTY

there are various summarizers and quiz generators available, but none of them are integrated with video lectures from YouTube. We are providing a comprehensive solution by combining video lectures from popular channels such as Khan Academy and ByJu's, along with summary and questionnaire generator. The main objective of this project is to make it simple and efficient for students to evaluate their knowledge and determine where they stand.

OVERALL METHODOLOGY

In addition to what was completed in the Baseline results, we further complete the Questionnaire generator. For the same, firstly a constructor method is called to initialize the QuestionExtractor class with a parameter to indicate the number of questions that need to be generated or extracted. The method generating the questions takes a document string as its input and returns a dictionary of questions as the result. For the above, a method has been created that returns a list of entities according to the Named Entity Recognition tagger provided by the Spacy Library. The NER tagger analyzes the document and tags the words that represent named entities such as people, organizations, locations, etc. The method returns a list of entities as candidates for the questions which are used to rank the keywords and form the questions. NLP techniques and machine learning models have been used for the generation of questions in this project,

COMPARISION WITH BASELINE

- Firstly, we have added the quiz generation capability.
- The changes made between the baseline, mid-review and final submissions follow an iterative development process based on feedback from our TA and regular testing.
- In our baseline submission, we used a regex-based model that used cosine similarity for summarising YouTube video transcripts.
- But for the final submission, we wanted a much more efficient model to run this function. So, in our updated final submission we have now implemented a much more efficient Hugging face transformer model for summarising the YouTube transcript after getting feedback from our TA.

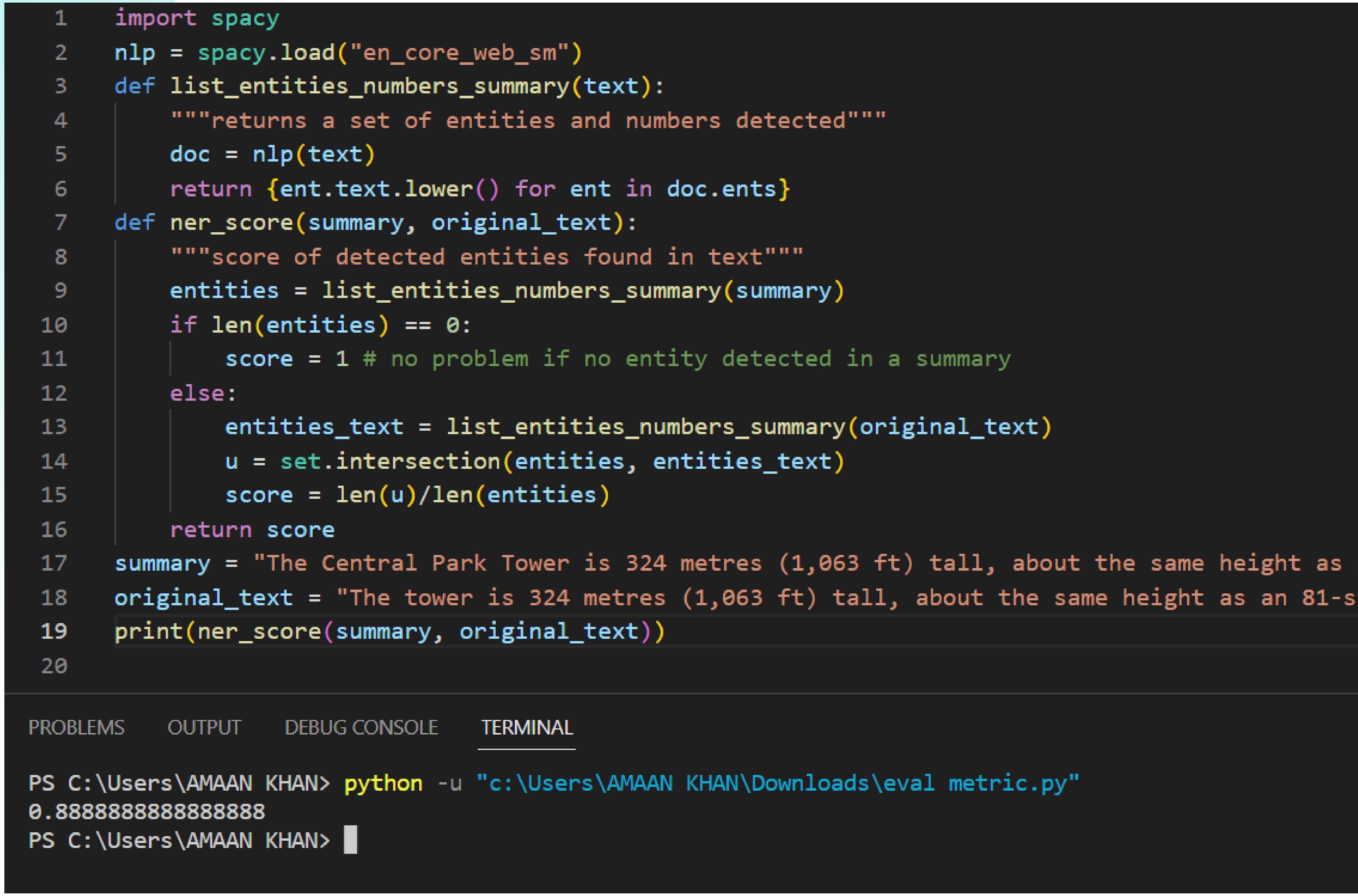
EVALUATION METRIC

For the evaluation, we used a commonly used code for which the link can be found at the bottom.

This code determines the similarity of the text generated by us with the original text that is obtained by the YouTube API transcript.

The resulting score as shown in the image gives us a score of 0.88... which indicates that our summary is of good accuracy and similar to the original transcript.

bit.ly/3N7sTt9



```python
import spacy
nlp = spacy.load("en_core_web_sm")
def list_entities_numbers_summary(text):
    """returns a set of entities and numbers detected"""
    doc = nlp(text)
    return {ent.text.lower() for ent in doc.ents}
def ner_score(summary, original_text):
    """score of detected entities found in text"""
    entities = list_entities_numbers_summary(summary)
    if len(entities) == 0:
        score = 1 # no problem if no entity detected in a summary
    else:
        entities_text = list_entities_numbers_summary(original_text)
        u = set.intersection(entities, entities_text)
        score = len(u)/len(entities)
    return score
summary = "The Central Park Tower is 324 metres (1,063 ft) tall, about the same height as
original_text = "The tower is 324 metres (1,063 ft) tall, about the same height as an 81-s
print(ner_score(summary, original_text))
```

```
PS C:\Users\AMAAN KHAN> python -u "c:\Users\AMAAN KHAN\Downloads\eval metric.py"
0.8888888888888888
PS C:\Users\AMAAN KHAN> 
```

CONTRIBUTIONS

The efforts put into the project were equally balanced amongst all the group members and was done collaboratively.

FRONTEND - PIYUSH
TEXT SUMMARIZATION - PIYUSH
EVALUATION - AMAAN, SHUBHAM, MEGHNA, PRAJEET
QUIZ GENERATION- AMAAN, SHUBHAM, MEGHNA
RESEARCH, PPT - AMAAN
LITERATURE REVIEW - NIKITA, PRAJEET
REPORT - NIKITA, PRAJEET