Option 3 : To create a summarization model

Explanation of the project:

This project aims to create a text summarization model on Google Colab. The model was chosen due to its state-of-the-art performance in natural language processing tasks, including text generation and summarization. Its large size and extensive pre-training make it suitable for complex summarization tasks.

I was failed making it with GPT-J model so build it with the help of ntlk library in python.

Method of approach:

1) Reading the content of the "fb.txt" file and splitting it into sentences.
2) Preprocessing each sentence by removing non-alphabetic characters and converting words to lowercase.
3) Calculating the cosine distance-based similarity between each pair of sentences.
4) Building a similarity matrix to store the pairwise similarities.
5) Using the PageRank algorithm to rank the sentences based on their similarity scores.
6) Selecting the top-ranked sentences to form the summary.

Documentation of methods used:

read_article(file_name): Reads the content of the file and preprocesses the sentences.

sentence_similarity(sent1, sent2, stopwords=None): Calculates the cosine similarity between two sentences after preprocessing.

build_similarity_matrix(sentences, stop_words): Constructs a similarity matrix for all sentences in the text.

generate_summary(file_name, top_n=5): The main function that generates the summary by ranking and selecting the top-n sentences.

AI tools:

NLTK (Natural Language Toolkit): Used for text preprocessing tasks like removing stopwords and stemming words.

Numpy: Utilized to handle numerical computations and create the similarity matrix.

NetworkX: Employed for implementing the PageRank algorithm to rank sentences.

CODE:

```python
import nltk
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
import networkx as nx

def read_article(file_name):
    file = open(file_name, "r")
    filedata = file.readlines()
    article = filedata[0].split(". ")
    sentences = []

    for sentence in article:
        print(sentence)
        sentences.append(sentence.replace("[^a-zA-Z]", " ").split(" "))
    sentences.pop()

    return sentences

def sentence_similarity(sent1, sent2, stopwords=None):
    if stopwords is None:
        stopwords = []

    sent1 = [w.lower() for w in sent1]
    sent2 = [w.lower() for w in sent2]

    all_words = list(set(sent1 + sent2))

    vector1 = [0] * len(all_words)
    vector2 = [0] * len(all_words)

    # build the vector for the first sentence
    for w in sent1:
        if w in stopwords:
            continue
        vector1[all_words.index(w)] += 1

    # build the vector for the second sentence
    for w in sent2:
        if w in stopwords:
            continue
        vector2[all_words.index(w)] += 1

    return 1 - cosine_distance(vector1, vector2)

def build_similarity_matrix(sentences, stop_words):
    # Create an empty similarity matrix
    similarity_matrix = np.zeros((len(sentences), len(sentences)))

    for idx1 in range(len(sentences)):
        for idx2 in range(len(sentences)):
            if idx1 == idx2: #ignore if both are same sentences
                continue
            similarity_matrix[idx1][idx2] =
sentence_similarity(sentences[idx1], sentences[idx2], stop_words)

    return similarity_matrix


def generate_summary(file_name, top_n=5):
    nltk.download("stopwords")
    stop_words = stopwords.words('english')
    summarize_text = []

    # Step 1 - Read text anc split it
    sentences =  read_article(file_name)

    # Step 2 - Generate Similary Martix across sentences
    sentence_similarity_martix = build_similarity_matrix(sentences,
stop_words)

    # Step 3 - Rank sentences in similarity martix
    sentence_similarity_graph =
nx.from_numpy_array(sentence_similarity_martix)
    scores = nx.pagerank(sentence_similarity_graph)

    # Step 4 - Sort the rank and pick top sentences
    ranked_sentence = sorted(((scores[i],s) for i,s in
enumerate(sentences)), reverse=True)
    # print("Indexes of top ranked_sentence order are ",
ranked_sentence)

    for i in range(top_n):
      summarize_text.append(" ".join(ranked_sentence[i][1]))

    # Step 5 - Offcourse, output the summarize text
    print("\n Summarize Text: \n", ". ".join(summarize_text))

generate_summary( "/fb.txt", 2)
```

Output Of Code :

fb.txt file content

```
For years, Facebook gave some of the world's largest technology companies more intrusive
access to users' personal data than it has disclosed, effectively exempting those business
partners from its usual privacy rules, according to internal records and interviews
The special arrangements are detailed in hundreds of pages of Facebook documents obtained
by The New York Times
The records, generated in 2017 by the company's internal system for tracking partnerships,
provide the most complete picture yet of the social network's data-sharing practices
They also underscore how personal data has become the most prized commodity of the digital
age, traded on a vast scale by some of the most powerful companies in Silicon Valley and
beyond
The exchange was intended to benefit everyone
Pushing for explosive growth, Facebook got more users, lifting its advertising revenue
Partner companies acquired features to make their products more attractive
Facebook users connected with friends across different devices and websites
But Facebook also assumed extraordinary power over the personal information of its 2 billion
users - control it has wielded with little transparency or outside oversight.Facebook allowed
Microsoft's Bing search engine to see the names of virtually all Facebook user's friends
without consent, the records show, and gave Netflix and Spotify the ability to read Facebook
users' private messages.

 Summarize Text:
 For years, Facebook gave some of the world's largest technology companies more intrusive
access to users' personal data than it has disclosed, effectively exempting those business
partners from its usual privacy rules, according to internal records and interviews. Facebook
users connected with friends across different devices and websites
```

So as you can see the output summarize text is what the task asked for , I tried the GPT-J model , but in colab downloading of the gpt model transformers library was crashing again and agin so made the summarizer with NLTK