DocuPresenter

Overview

DocuPresenter is a tool that allows you to create presentations using text extracted from PDF documents. This README provides a step-by-step guide on how to set up and use DocuPresenter.

Setup

1. First, install the required Python packages by running the following commands:

```
!pip install -U -q google-generativeai

!pip install --upgrade google-api-python-client

!pip install PyPDF2

!apt-get update

!apt-get install -y wkhtmltopdf
```

2. Import the necessary libraries and set up your API Key:

```
import google.generativeai as palm

import textwrap

import numpy as np

import pandas as pd



# Set your API Key

palm.configure(api_key='YOUR_API_KEY_HERE')
```

3. Choose a model for text embedding. The following code lists available models and selects one for text embedding.

```
models = [m for m in palm.list_models() if 'embedText' in m.supported_generation_methods]

model = models[0]  # Choose the desired model
```

Retrieve PDF Files

1. Mount Google Drive to Google Colab to access your PDF files. Run the following code:

```
from google.colab import drive

drive.mount('/content/gdrive', force_remount=True)
```

2. Extract text from PDF files in a specific folder and create a Pandas DataFrame to store it. Ensure that your PDF files are stored in the folder you specify (folder_path). The code reads and compiles its text content, removing newline characters for better formatting.

“shortened_text = cleaned_text.encode('utf-8')[:9900].decode('utf-8', errors='ignore')”
At this moment, a characters limit of 9900 bytes is set since there is a maximum limit of 10000 bytes when embedding text. This processing step helps avoiding any potential errors. However, under ideal conditions, this API will be able to process a larger amount of text.

```
import os

import PyPDF2

import pandas as pd



folder_path = "/content/gdrive/MyDrive/Test"

pdf_dict = {}



# Check if the folder exists

if os.path.exists(folder_path):

    # Iterate over files in the folder

    for filename in os.listdir(folder_path):

        if filename.endswith(".pdf"):  # Check if it's a PDF file

            pdf_path = os.path.join(folder_path, filename)
```

```
            with open(pdf_path, 'rb') as pdf_file:
                pdf_reader = PyPDF2.PdfReader(pdf_file)

                # Extract text from each page
                text = ""
                for page_num in range(len(pdf_reader.pages)):
                    page = pdf_reader.pages[page_num]
                    text += page.extract_text()

                # Remove newline characters from the text
                cleaned_text = text.replace("\n", "")

                # Encode to bytes, extract the first 9000 bytes, and then decode
                shortened_text = cleaned_text.encode('utf-8')[:9900].decode('utf-8', errors='ignore')

                # Add the shortened text to the dictionary
                pdf_dict[filename] = shortened_text

else:
    print(f"Folder {folder_path} does not exist!")
# Convert the dictionary to a dataframe
df = pd.DataFrame(list(pdf_dict.items()), columns=['Filename', 'Text'])
df
```

3. Generate embeddings for the text and add them to the DataFrame.

```
def embed_fn(text):

  return palm.generate_embeddings(model=model, text=text)['embedding']

df['Embeddings'] = df['Text'].apply(embed_fn)
```

Query the Documents

1. Specify the topic for your query.

```
topic = "Physical Geography of Africa"
```

2. Create a function to find the most relevant passage related to the topic in your documents.

```
def find_best_passage(topic, dataframe):

  # Compute distances and return the most relevant passage

  # ...

  return relevant_passage
```

3. Query the documents to find the best passage.

```
passage = find_best_passage(topic, df)
```

4. Create a prompt using the found passage and topic.

```
prompt = make_prompt(topic, passage)
```

Generate a Presentation

1. Choose a text generation model and set parameters like temperature.

```
text_models = [m for m in palm.list_models() if 'generateText' in
m.supported_generation_methods]

text_model = text_models[0]  # Choose a text generation model

temperature = 0.5
```

2. Generate a presentation using the prompt and the selected text generation model.

```
        answer = palm.generate_text(prompt=prompt, model=text_model, temperature=temperature, max_output_tokens=1000)
```

3. Convert the generated Markdown content to a PDF and save it to Google Drive.

```
        # Convert the Markdown to PDF

        markdown_html = answer.result

        pdf_file_path = 'output.pdf'

        html = weasyprint.HTML(string=markdown_html)

        html.write_pdf(pdf_file_path)

        # Specify the Google Drive folder path and authenticate

        folder_path = "/content/gdrive/MyDrive/Test"

        # Authenticate with Google Drive and upload the PDF

        # ...



        # Print a confirmation message

        print(f'PDF file "{pdf_filename}" uploaded to Google Drive in folder: {folder_path}')
```