SCTR’s Pune Institute of Computer Technology (PICT), Pune
Maharashtra 411043

B.E Artificial Intelligence for Big Data Mining (410503) (Honors)

SUBMITTED BY

Name: Gaurav Boob

ABC Id: 979-134-252-303

PRN No: 72138249F

Class: BE 5

Roll no: 42209

Under the guidance of

Ms.S.M.Hosamani

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

ACADEMIC YEAR 2023-24 SEM II

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

SCTR's Pune Institute of Computer Technology (PICT), Pune
Maharashtra 411043

CERTIFICATE

This is certified that DS-AIBDA laboratory experiment/project submitted by **Gaurav Boob** has satisfactorily completed the curriculum-based B.E. Artificial Intelligence for Big Data Mining Honors project under the guidance of Ms.S.M.Hosamani towards the partial fulfillment of final year Electronics and Telecommunication Engineering Semester VIII, Academic Year 2023-24 of Savitribai Phule Pune University.

Ms.S.M.Hosamani Principal

Place:

Date:

Problem Statement: LLMs for interaction with scientific documentation.

In the realm of scientific research and documentation, the sheer volume and complexity of information often pose significant challenges for researchers, academics, and professionals alike. Leveraging Language Model Models (LLMs) for interaction with scientific documentation presents an opportunity to streamline and enhance various aspects of this process. LLMs, powered by advanced natural language processing techniques, have shown promise in understanding, summarizing, generating, and contextualizing textual data.

However, despite their potential, effectively utilizing LLMs for interaction with scientific documentation remains a multifaceted challenge. Firstly, scientific texts often contain highly technical jargon, complex terminology, and nuanced concepts, which may present obstacles for LLMs in accurately interpreting and generating meaningful responses. Secondly, the diverse range of disciplines within the scientific community necessitates adaptability and specialization in LLMs to cater to specific domains and subfields.

Moreover, ensuring the reliability, accuracy, and credibility of information extracted or generated by LLMs from scientific documents is paramount. This involves addressing issues such as bias, misinformation, and the ability to discern between validated research findings and speculative conjecture.

Furthermore, the usability and accessibility of LLMs for individuals with varying levels of expertise in scientific research must be considered. Designing intuitive interfaces and developing user-friendly functionalities that facilitate efficient navigation, information retrieval, and collaboration within scientific documents are essential for maximizing the utility of LLMs in this context.

Therefore, the challenge lies in harnessing the capabilities of LLMs to empower researchers, educators, students, and professionals in their quest for knowledge discovery, dissemination, and innovation within the vast landscape of scientific documentation. By addressing these challenges, we can unlock the full potential of LLMs as indispensable tools for advancing scientific inquiry, communication, and progress.

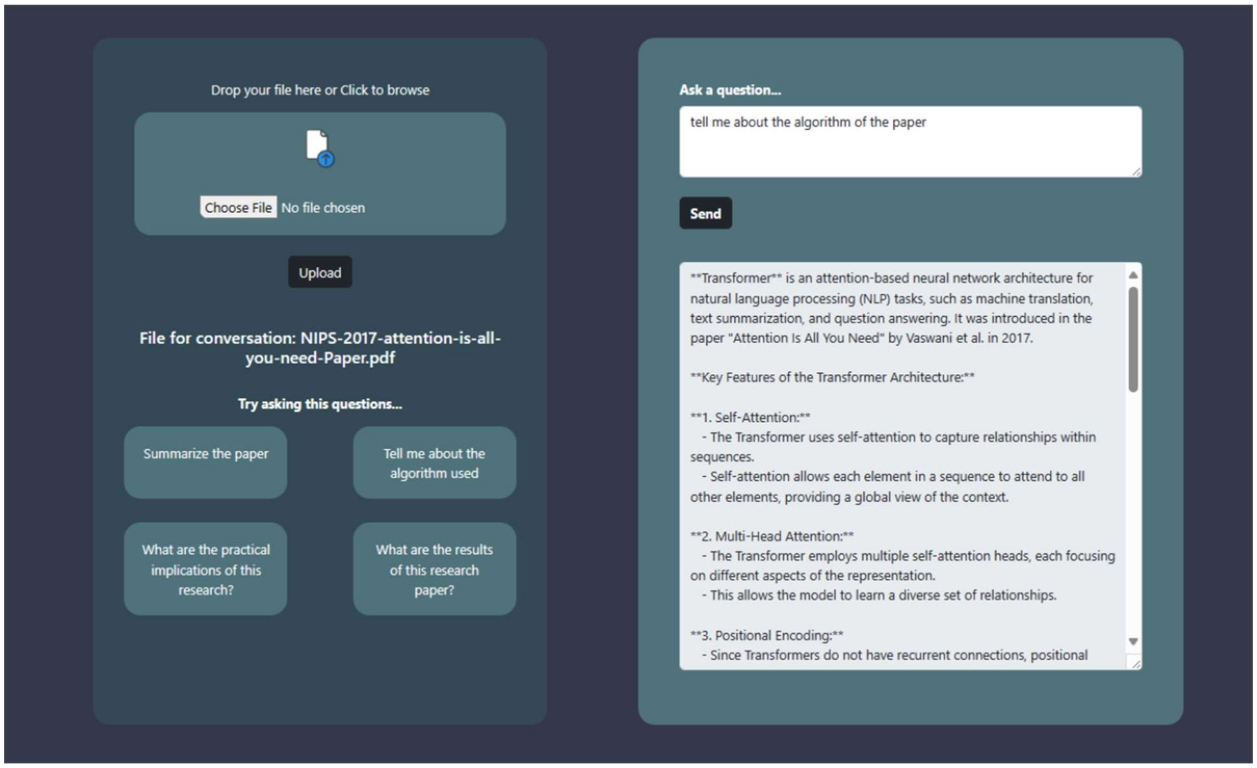
Code :

```
from django.shortcuts import render
import re
import os
import json
from django.core.files.storage import FileSystemStorage
from django.http import HttpResponse, JsonResponse, FileResponse
import google.generativeai as genai
import re
from tika import parser
import docx
import pathlib
from docx import Document
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from pathlib import Path
import matplotlib.pyplot as plt
import csv
import base64
from io import StringIO, BytesIO
from django.shortcuts import render
from io import StringIO
import random
import nltk

nltk.download('stopwords')
nltk.download('punkt')

import convertapi

convertapi.api_secret = ''
genai.configure(api_key="")

def remove_stopwords(text):
   stop_words = set(stopwords.words('english'))
   word_tokens = word_tokenize(text)
   filtered_text = [word for word in word_tokens if word.lower() not in stop_words]
   return ' '.join(filtered_text)

def genaiModel(file, query):
   generation_config = {
   "temperature": 0.9,
   "top_p": 1,
   "top_k": 1,
   "max_output_tokens": 2048,
   }

   safety_settings = [
   {
     "category": "HARM_CATEGORY_HARASSMENT",
     "threshold": "BLOCK_MEDIUM_AND_ABOVE"
   },
   {
     "category": "HARM_CATEGORY_HATE_SPEECH",
     "threshold": "BLOCK_MEDIUM_AND_ABOVE"
   },
   {
     "category": "HARM_CATEGORY_SEXUALLY_EXPLICIT",
     "threshold": "BLOCK_MEDIUM_AND_ABOVE"
   },
   {
     "category": "HARM_CATEGORY_DANGEROUS_CONTENT",
     "threshold": "BLOCK_MEDIUM_AND_ABOVE"
   },
   ]
   model = genai.GenerativeModel(model_name="gemini-1.0-pro",
                                 generation_config=generation_config,
                                 safety_settings=safety_settings)
   root, extension = os.path.splitext(file)
   file = root
   text = open('output/' + file + '.txt', "r+", encoding="utf8", errors="ignore")

   que= query
   prompt_parts = [
   text.read(),
   "Answer the questions only if they are related to given paper.",
   query,
   ]

   response = model.generate_content(prompt_parts)
   return response.text

def upload_files(request):
   print('upload file')
   output = ""
   file = None
   context = {
     'file' : "Not uploaded",
     'output' : output
   }
   if request.method == 'POST':
```

```
if 'upload' in request.POST:
    for file in request.FILES.getlist('document'):
        file_name = re.sub(r"(\s)|(\")|(')|(&)", "_", str(file.name))
        file_name = re.sub(r'%22','_',f'{file_name}')
        file.name = file_name
        fs = FileSystemStorage()

        path = "data/"+file.name
        extension = pathlib.Path(path).suffix

        print(extension)
        if extension == ".pdf":
            if fs.exists(file.name):
                fs.delete(file.name)
            fs.save(file.name, file)
            convertapi.convert('txt', {
                'File': path
            }, from_format = 'pdf').save_files('output')
            context = {
                'file' : file,
                'output' : ""
            }
        elif extension == ".pptx" or extension ==  ".ppt":
            if fs.exists(file.name):
                fs.delete(file.name)
            fs.save(file.name, file)
            convertapi.convert('pdf', {
                'File': path
            }, from_format = 'pptx').save_files('pdf')
            print("fileName:" , file.name)
            convertapi.convert('txt', {
                'File': 'pdf/'+ 'Host-afe.pdf'
            }, from_format = 'pdf').save_files('output')
            context = {
                'file' : file,
                'output' : ""
            }
        elif extension == ".docx" :
            if fs.exists(file.name):
                fs.delete(file.name)
            fs.save(file.name, file)

            filename = path
            document = docx.Document(filename)
            text_file_path = 'output\\' + filename[5:-5] + '.txt'
            print(text_file_path + "!!!")
```

```
                with open(text_file_path, "w", encoding="utf-8") as f:

                    f.write("**Text:**\n")
                    for paragraph in document.paragraphs:
                        f.write(paragraph.text.strip() + "\n")

                    f.write("\n**Tables:**\n")
                    for table in document.tables:
                        for row in table.rows:
                            for cell in row.cells:
                                f.write(cell.text.strip() + "\t")
                            f.write("\n")
                context = {
                    'file' : file,
                    'output' : ""
                }
            elif extension == ".tex":
                if fs.exists(file.name):
                    fs.delete(file.name)
                fs.save(file.name, file)

                context = {
                    'file' : file,
                }
            return render(request, 'upload.html', context)
        elif 'visualize' in request.POST:
            print("in")
            op =  request.POST.get('outputText1')
            modified_text = op.replace('|', ',')
            print(modified_text)
            csv_file_like_object = StringIO(modified_text)
            csv_reader = csv.reader(csv_file_like_object)
            data = list(zip(*csv_reader))
            x_values = data[0]
           y_values = [list(map(lambda x: int(x) if x.isdigit() else random.randint(1,100), col)) for col
in data[1:]]

            fig, ax = plt.subplots()
            for y_values_col in y_values:
                ax.plot(x_values, y_values_col, marker='o')

            ax.set_xlabel('X Values')
            ax.set_ylabel('Y Values')
            ax.set_title('Dynamic CSV Data Plot')
            buffer = BytesIO()
            plt.savefig(buffer, format='png')
            buffer.seek(0)
            plt.close()
            plot_base64 = base64.b64encode(buffer.getvalue()).decode('utf-8')
            context = {
                'file' : file,
                'output' : output,
                'plot' : plot_base64
            }
            return render(request, 'upload.html', context)
        else:
            message = request.POST.get('message')
            file = request.POST.get('file')
            output = genaiModel(file, message)
            context = {
                'file' : file,
                'output' : output,
                'que' : message,
            }

    return render(request, 'upload.html', context)
```

Interface:


Drop your file here or Click to browse
Choose File
No file chosen
Upload
Try asking this questions...
Summarize the paper
Tell me about the algorithm used
What are the practical implications of this research?
What are the results of this research paper?
Ask a question...
Send

File for conversation: NIPS-2017-attention-is-all-you-need-Paper.pdf

tell me about the algorithm of the paper

```
**Transformer** is an attention-based neural network architecture for natural language processing (NLP) tasks, such as machine translation, text summarization, and question answering. It was introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017.

**Key Features of the Transformer Architecture:**

**1. Self-Attention:**
  - The Transformer uses self-attention to capture relationships within sequences.
  - Self-attention allows each element in a sequence to attend to all other elements, providing a global view of the context.

**2. Multi-Head Attention:**
  - The Transformer employs multiple self-attention heads, each focusing on different aspects of the representation.
  - This allows the model to learn a diverse set of relationships.

**3. Positional Encoding:**
  - Since Transformers do not have recurrent connections, positional
```