Projects

1.Chat with pdf

* Read pdf
* Convert to text chunks
* Convert to text embeddings
* Create knowledge base (vectore store)

USER SIDE

* Read question ,convert to query embeddings and similarity search on vector data base
* Convert the search results to natural language using llm

Code

* Read pdf

**pdf = st.file\_uploader("upload the pdf",type='pdf')**

**text=""**

**for page in pdf\_reader.pages:**

**text+=page.extract\_text()**

* Convert to text chunks

**text\_splitter = RecursiveCharacterTextSplitter(**

**chunk\_size=1000,**

**chunk\_overlap=200,**

**length\_function=len**

**)**

**chunks = text\_splitter.split\_text(text=text)**

* Convert to text embeddings

**embeddings = OpenAIEmbeddings()**

**Vectorstore = FAISS.from\_texts(chunks,embedding=embeddings)**

* Query

**docs = Vectorstore.similarity\_search(query=query)**

**llm = OpenAI(temperature=0)**

**chain = load\_qa\_chain(llm=llm,chain\_type='stuff')**

**response = chain.run(input\_documents=docs,question=query)**

**2.document summarization**

* Load the model lamini

**checkpoint ="LaMini-Flan-T5-248M"**

**tokenizer = T5Tokenizer.from\_pretrained(checkpoint)**

**base\_model = T5ForConditionalGeneration.from\_pretrained(checkpoint,device\_map='auto',torch\_dtype=torch.float32)**

* Load the file and split to chunks

**loader = PyPDFLoader(file)**

**pages = loader.load\_and\_split()**

**text\_splitter = RecursiveCharacterTextSplitter(chunk\_size=200,chunk\_overlap=50)**

**texts = text\_splitter.split\_documents(pages)**

**final\_text =""**

**for text in texts:**

**print(text)**

**final\_text = final\_text + text.page\_content**

* Load the llm pipeline

**def llm\_pipeline(filepath):**

**pipe\_sum = pipeline(**

**'summarization',**

**model=base\_model,**

**tokenizer = tokenizer,**

**max\_length = 500,**

**min\_length=50 )**

**input\_text = file\_loader(filepath)**

**result = pipe\_sum(input\_text)**

**result = result[0]['summary\_text']**

**return result**

* Display pdf

**@st.cache\_data**

**def display\_pdf(file):**

**# Opening file from file path**

**with open(file, "rb") as f:**

**base64\_pdf = base64.b64encode(f.read()).decode('utf-8')**

**# Embedding PDF in HTML**

**pdf\_display = F'<iframe src="data:application/pdf;base64,{base64\_pdf}" width="100%" height="600" type="application/pdf"></iframe>'**

**# Displaying File**

**st.markdown(pdf\_display, unsafe\_allow\_html=True)**

3.fake news detection

* Read the text
* Remove not alphabetical character
* Convert to lower case
* Split the text
* Stem the words
* Vectorize the word tfidf

Till stem

**def stemming(content):**

**stemmed\_content = re.sub('[^a-zA-Z]',' ',content)**

**stemmed\_content = stemmed\_content.lower()**

**stemmed\_content = stemmed\_content.split()**

**stemmed\_content = [ps.stem(word) for word in stemmed\_content if not word in stopwords.words('english')]**

**stemmed\_content = ' '.join(stemmed\_content)**

**return stemmed\_content**

* vectorize

**vector = TfidfVectorizer()**

**vector.fit(X)**

**X = vector.transform(X)**