ASSESSMENT COVER SHEET

Assignment Details

Course: Applied Natural Language Processing

Semester/Academic Year: Sem 1 2023

Assignment title: Question Answering System Using OpenAi

Assessment Criteria

Assessment Criteria are included in the Assignment Descriptions that are published on each course's website.

Plagiarism and Collusion

Plagiarism: using another person's ideas, designs, words, or works without appropriate acknowledgment.

Collusion: another person assisting in the production of an assessment submission without the express requirement, consent, or knowledge of the assessor.

Consequences of Plagiarism and Collusion

The penalties associated with plagiarism and collusion are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity. Penalties may include the requirement to revise and resubmit assessment work, receiving a result of zero for the assessment work, failing the course, expulsion, and/or receiving a financial penalty.

declaration

I declare that all material in this assessment is my own work except where there is clear acknowledgment and reference to the work of others. I have read the University Policy Statement on Plagiarism, Collusion, and Related Forms of Cheating:

mttp://www.adelaide.edu.au/policies/?230

I give permission for my assessment work to be reproduced and submitted to academic staff for the purposes of assessment and to be copied, submitted, and retained in a form suitable for electronic checking of plagiarism.

Yash Kasundra (A1838670)
28/06/2023

SIGNATURE AND DATE

Introduction

Our goal in this project is to create a ChatPDF-like system for answering questions. The system aims to quickly respond to three different types of questions: direct query questions, indirect query questions, and identification of essential references that serve as the foundation for a paper's methodology.

Our algorithm will examine the text of a paragraph for direct query questions and provide precise responses. These questions frequently require searching for specific details contained inside the paragraph. To do this, text preprocessing techniques will be used, and embeddings or other types of representation for sentences or paragraphs will be generated. The amount of material needed for analysis can be greatly reduced by finding important sentences or paragraphs that are connected to the query. The system will therefore be developed to fit into an 800-token window, and prompts will be made to ask ChatGPT for the solution.

On the other hand, indirect query questions lack specific keywords that are found in the text. Therefore, to answer these queries truthfully, our system needs to improve its matching capabilities. By employing appropriate techniques, we can enhance the system's capacity to locate pertinent data even in the absence of explicit keyword matches.

Finally, the system should be able to localize important references that serve as the basis for the suggested technique in a particular paper. Without using ChatGPT, this localization stage can be completed using straightforward methods. We can effectively locate and extract these references, giving consumers important information about the methodology's underlying assumptions.

Throughout the development process, we have the freedom to utilize third-party libraries for tasks such as text preprocessing, generating embeddings, and localizing references, in addition to the GPT API. However, it is essential to implement the overall workflow ourselves. Furthermore, it is required to report the number of tokens used by ChatGPT when answering questions during testing.

By creating an efficient and accurate question-answering system, we can facilitate the retrieval of specific information from pdf, research papers without explicit keyword matches, and provide insights into the influential references behind the proposed methodology.

1. Pre-processing and EDA:

Preprocessing is essential to every natural language processing (NLP) project if we want to increase the efficacy of our model and the quality of our data. In this project, we put much time and effort into meticulously cleaning and transforming our textual data. These initial steps have been diligently developed to increase our system's precision and effectiveness. Let's now examine the particular preparation methods applied to our dataset:

1. **Removal of HTML tags and brackets**: The text likely contained HTML tags or brackets, which were not relevant to the analysis. These tags and any content within them were removed to eliminate any potential noise in the data.
2. **Removal of special characters**: Special characters, apart from full stops, were removed from the text. This step aimed to eliminate unnecessary symbols that might hinder the subsequent analysis.
3. **Removal of non-alphanumeric characters**: All non-alphanumeric characters, such as punctuation marks (except full stops), were removed from the text. This step helps in simplifying the text and making it more consistent for further processing.
4. **Removal of numbers**: Since the focus of the project does not involve analyzing numerical data within the text, numbers were removed from the text. However, it was noted that the original text still contains numbers, which can be extracted if needed for obtaining the best reference.
5. **Removal of empty characters**: Any empty or whitespace characters were removed from the text to ensure consistency and avoid potential issues during analysis.

After performing the initial preprocessing steps, exploratory data analysis (EDA) techniques were applied to gain insights into the data. The following approaches were employed:

1. **Wordcloud**: A word cloud is a visual representation of the most frequently occurring words in a text corpus. By creating a word cloud, it becomes possible to identify the most common terms within the data. This visualization helps in understanding the prominent themes or concepts present in the text.

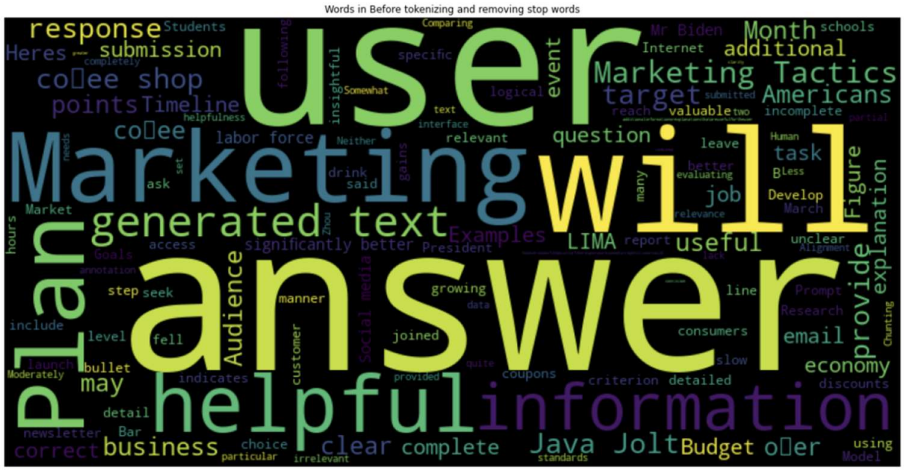

Fig-1: Word cloud for semi-cleaned data

2. **N-gram plots**: N-grams are contiguous sequences of n items (words, characters, etc.) within the text. By generating n-gram plots, which depict the frequency of specific n-gram sequences, it becomes possible to observe patterns or relationships between words. This analysis aids in uncovering significant collocations or phrases that may carry relevant information for the question-answering system.

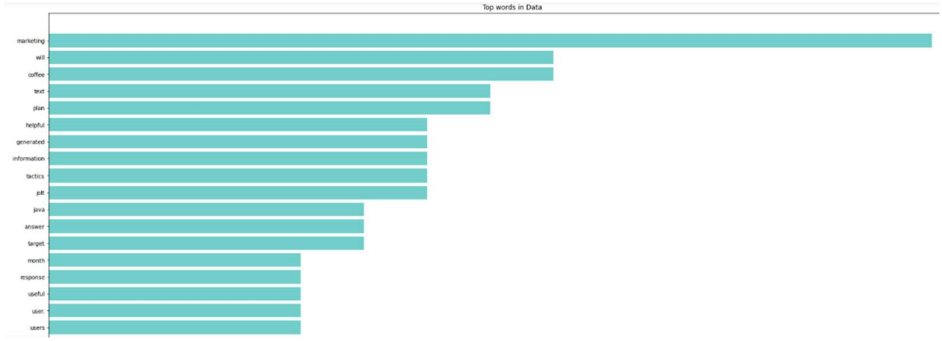

Fig-2: Unigram plot for semi-cleaned data

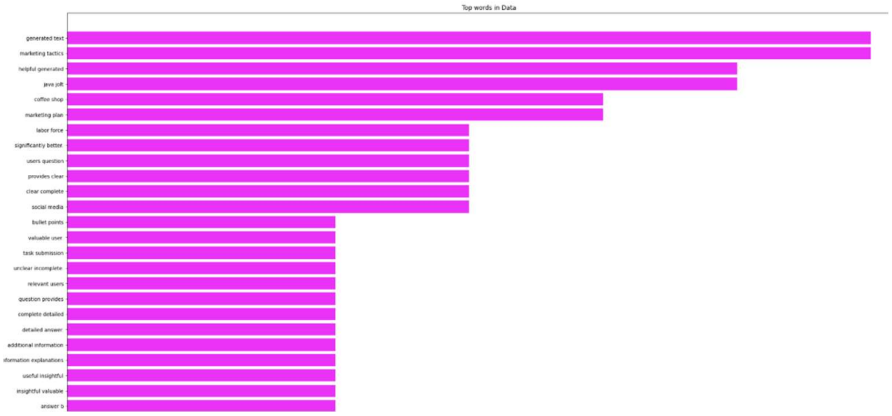

Fig-3: Bigram plot for semi-cleaned data

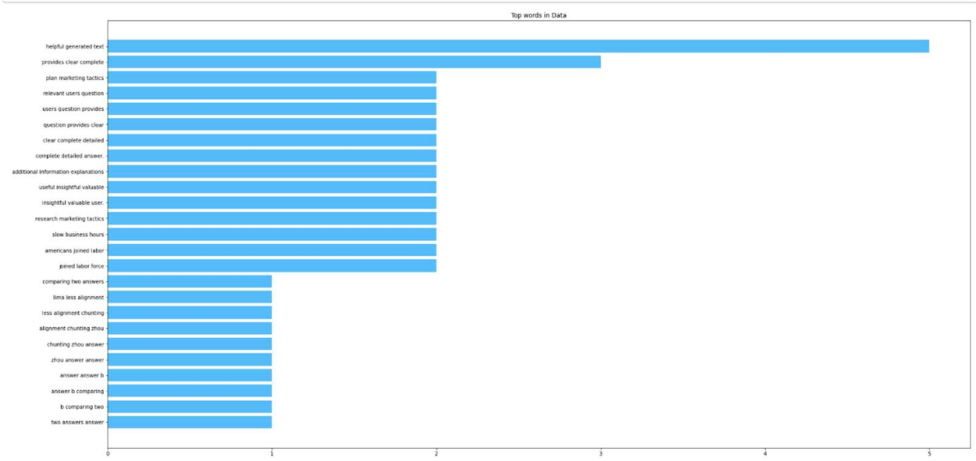

Fig-4: Trigram plot for semi-cleaned data

In addition to the preprocessing and EDA steps described above, the gensim library's preprocessing module was utilized. This module offers functionality to remove stopwords, which are commonly occurring words (e.g., "the," "and," "is") that do not carry much information for analysis. Stopword removal helps to focus on the more meaningful terms in the text. Additionally, the text was tokenized using the library, which involves breaking the text into individual tokens (words or phrases) for further analysis.

By performing these preprocessing operations and exploring the cleaned data through techniques like word clouds and n-gram plots, we can gain valuable insights into the text corpus and prepare it for subsequent stages of analysis and modeling.

2. Model

In this project, we used SBERT first to provide us with the top 10 most likely answers to the question asked by the user. So let's look at how SBERT works

i. **S-BERT (Sentence Bidirectional Encoder Representations from Transformers):**

Using a cross-encoder network based on BERT, SBERT (Sentence-BERT) addresses the semantic search problem. By contrasting pairs of sentences, it determines similarity ratings. However, comparing several sentences necessitates lengthy computations and training periods. By entering individual sentences and averaging word-level embeddings, researchers attempted to use BERT to generate sentence embeddings as a workaround. This method frequently resulted in poor sentence embeddings [1].

SBERT uses siamese neural networks to address this. It is made up of two BERT architectures that are exact duplicates and share weights. Using mean-pooling, sentence pairings are processed to create pooled sentence embeddings. The embeddings are combined and then run through a softmax classifier with a softmax-loss function during training. At inference, a similarity score is calculated by comparing the embeddings using cosine similarity [1].

Siamese networks are effective because they can compare inputs and are adaptable for a variety of uses. Due to the removal of the final classification head and the use of two BERT architectures with shared weights for processing sentence pairs, SBERT's Siamese architecture performs better than the original cross-encoder architecture of BERT [1].

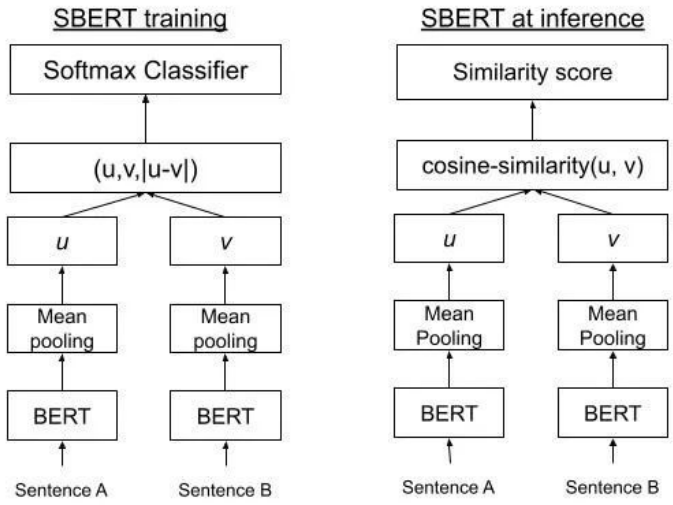

Fig-5: Architecture of S-BERT model [1]

ii. **CHAT-GPT (OpenAi):**

The GPT (Generative Pre-trained Transformer) architecture, specifically the GPT-3.5 variation, is the foundation of ChatGPT. It is a cutting-edge language model created to produce text replies that resemble those of a human in response to provided instructions or messages [2].

Architecture:

1. A transformer-based design serves as the foundation for ChatGPT. Transformers are deep learning models that depend on processes for self-attention to identify contextual relationships between words or tokens in a sequence. Due to its

architecture, ChatGPT can effectively model and produce responses that are both coherent and appropriate for the situation [2].

2. Encoder-Decoder Configuration: ChatGPT uses an encoder-decoder configuration. The model's encoder, which captures the contextual representation of the text, encodes the input messages or prompts. Based on the learned representation, the decoder then produces a response [2].
3. Large-Scale Pre-Training: ChatGPT has already received extensive pre-training on a wide variety of text data gathered from the internet. Predicting masked words or tokens in the input text during this pre-training phase enables the model to learn linguistic patterns, syntax, and semantic correlations. The extensive pre-training enables ChatGPT to produce pertinent and logical responses in a variety of scenarios and conversational subjects [2].
4. After pre-training, ChatGPT is fine-tuned using special datasets chosen for a given purpose. To improve the model's performance and customize it for particular applications, fine-tuning entails training the model using domain- or task-specific data. ChatGPT may be fine-tuned to be more conversational and context-aware when used with chat-based applications.

Working: ChatGPT tokenizes input into smaller units (words or subwords) and feeds these smaller units into the encoder when a message or prompt is supplied to it.
To provide a contextual representation of the input message, the encoder processes the tokens. The decoder receives this representation and produces a response based on the ingested context and linguistic patterns [2].

ChatGPT employs a sampling strategy during inference to produce a variety of original responses. To change the randomness and diversity of the generated responses, it can additionally incorporate strategies like top-k sampling or temperature control. The model can generate text outputs that mimic human-like dialogue and are instructive, interesting, and contextually relevant [2].

Although ChatGPT can provide some outstanding results, it occasionally produces outputs that are inaccurate, illogical, or biased. When implementing and utilizing the concept in actual applications, much thought and evaluation are required [2].

Overall, ChatGPT is an effective tool for chat-based applications due to its transformer-based architecture, extensive pre-training, and fine-tuning. As a result, it can produce conversational and context-aware responses [2].

iii. Text Summarization using Sumy Library:

Text summarization is a crucial task in natural language processing, aiming to condense a given text into a concise and informative summary. In this report, we will explore the Sumy library and focus on the Latent Semantic Analysis (LSA) algorithm, which is one of the methods implemented in Sumy for extractive text summarization [5].

Text summarization methods can be broadly categorized into two main types: extractive and abstractive. Extractive text summarization involves extracting important sentences from the original text, while preserving the original language. Abstractive text summarization, on the other hand, generates summaries by producing new sentences that may or may not exist in the original text [5].

Extractive Text Summarization with Sumy: Sumy is a Python library that simplifies the process of text summarization, offering various algorithms to achieve this task. One of the algorithms available in Sumy is the Latent Semantic Analysis (LSA). LSA is a statistical

method that identifies the latent semantic structure of the text to extract key information [5].

Working Principle of LSA: The LSA algorithm used in Sumy relies on singular value decomposition (SVD) to capture the underlying semantic relationships within the text. It constructs a matrix representing the relationships between words and sentences, allowing it to identify important concepts and patterns. By analyzing this matrix, LSA assigns weights to sentences based on their importance to the overall meaning of the text [5].

Implementation and Usage: Using Sumy with the LSA algorithm is straightforward. After installing the library, developers can integrate it into their Python projects and utilize the LSA algorithm for text summarization. By providing a text document as input, Sumy applies LSA to extract the most relevant sentences, creating a summary that captures the essential information [5].

Benefits and Limitations: The LSA algorithm offers several advantages for text summarization. It considers the semantic relationships between words and sentences, allowing it to capture contextual information effectively. LSA can handle large volumes of text and generate summaries that maintain the cohesiveness of the original content.

However, it is important to note that LSA has limitations. Since it relies on statistical patterns, it may not fully comprehend the nuances of language or grasp complex context. Additionally, LSA's performance can be influenced by the quality and diversity of the training data, as well as the preprocessing techniques employed.

Conclusion: The Sumy library provides a convenient and user-friendly solution for text summarization. Its implementation of the LSA algorithm offers an effective approach for extractive text summarization. By leveraging semantic relationships, LSA enables the generation of concise summaries that capture the main ideas of the original text.

To further enhance the text summarization process, developers can experiment with different algorithms offered by Sumy and explore the integration of abstractive methods. Regular evaluation and refinement based on user feedback are crucial for improving the accuracy and effectiveness of text summarization systems.

3. Analyzing solutions for each Question Type and Addressing Limitations:

i. **Direct Questions:** At the start, you can specify the path of the pdf file and the question you want to ask, through the CMD line. Once the data was loaded and I observed that it contains noise such as numbers, whitespaces, punctuations, and non-alphanumeric characters. To address this, the data was cleaned using the regex library. Further exploratory data analysis (EDA) techniques like word clouds and n-gram plots were employed to identify the most frequently used terms. It was discovered that stopwords were present in the most used terms, which were removed using the gensim library. The data was then tokenized and stored in a new column named "preprocessed_data". We pass the question we received from the user at the start and the preprocessed data into S-Bert, the S-Bert model from the sentence transformer library was selected based on extensive research and experimentation.

   The preprocessing and S-Bert-based solution yielded promising results. The token count of the data needed to be estimated for passing it to the OpenAI API, thus I used the tiktoken library recommended on the Openai website. After that, I created a function that will take the API key of the user and pass the data to chatgpt which will give 1 answer to the question asked based on the data we got from Sbert.

   Limitations: Reliance on a pre-trained model like S-Bert for question-answering poses limitations, such as its performance being influenced by the quality and representativeness of the pretraining data. Out-of-domain or specialized questions may also pose challenges. Furthermore, using this data the response generated by Chatgpt may produce incorrect or biased answers.

ii. **Indirect Questions:** Indirect Questions: Indirect questions present a different challenge, as they require the model to generate responses based on patterns and information present in the training data. The model relies on statistical patterns learned from the training data rather than a deep understanding of the specific question's content or context. First we will pass the data into S-Bert and then we found out that S-Bert doesn’t work well with indirect data since it semantically matches the data. Next, we used sumy library which is used for text summarization. We then pass the output generated from sumy to tiktoken to get the number of token we will use before passing it into Chatgpt. After passing it into chatgpt we can get the finial output.

   Limitations: It's important to note that this solution also relies on a pre-trained model, which inherits the limitations previously mentioned. Additionally, the approach of passing frequently occurring word phrases from the article or PDF to generalize the data and assist the model in answering indirect questions has its limitations. If the PDF itself lacks phrases that provide crucial information or instead contain irrelevant references, the model may struggle to answer indirect questions in such cases. Furthermore, if the preprocessing step is not conducted accurately, static elements such as stopwords, punctuations, or non-alphanumeric characters may end up being identified as the most frequently used phrases instead of the actual informative data.

iii. **Identification of Key Reference:** The simplest way is the citation count, which estimates a scientific publication's influence based on the number of citations it has [2]. We can also create a normalized citation count to provide a more equitable comparison of publications from various disciplines by multiplying a paper's citation count by the average number of citations received by papers in the same field of study. But this has its drawback which can be dealt with by just focusing on papers cited in the abstract or

Introduction, as the writers create an abstract to briefly and accurately summarize the work presented [4].

The second method(Local diffusion): The basic principle of the local diffusion approach is that papers that cite another publication should take note of that paper's significance. The frequency of co-citation of a paper and a paper by other papers serves as a good approximation of such recognition [4].

Step 1: The diffusion process starts with a value of 1 on paper α.

- The initial value on α is evenly distributed to the papers γ that cite α.
- Each paper γ receives a value of 1/kinα from α.
- The value Vαγ that a paper γ receives from α in the first step is calculated as Vαγ = Aγα / kinα.

Step 2: Redistributing the value from γ to β.

- The value on a paper γ is evenly redistributed back to the paper β cited by both γ and α.
- The value Uαβ that paper α's reference β receives is the summation of the values diffused from these papers.
- The final value Uαβ is calculated as Uαβ = Σ(Vαγ * Aγβ / koutγ).

Step 3: Adjust the diffusion score with a penalty factor.

- To suppress the tendency toward high-degree nodes, a parameter λ is introduced.
- Wαβ, the modified diffusion score, is calculated as Wλαβ = [kinβ]^-λ * Wαβ, where [kinβ]^-λ is the penalty factor based on the citation count.
- λ ($0 \leq \lambda \leq 1$) is a tunable parameter. A higher λ corresponds to a stronger penalty for highly cited references.

These steps aim to identify the most important references to a paper by considering both the citation network's importance and relevance to the target paper. The local diffusion method offers an alternative approach to PageRank, focusing on local relevance rather than global importance [4].

Limitations: The first method of identification of key references has a major drawback in citation count and its variants have been noted in the literature as being subpar indicators of a paper's quality. Also, a work may be referenced in comment papers that are intended to highlight its flaws rather than being pivotal and relevant [4].

4. Results & Observation:

Due to time and resource constraints, I could not implement the part for Identifying Key references. But I have added detail about how it can be done using 2 different methods. Now let's look at the few sample questions I tested my model on, but before that let me re-iterate that during the start of the code, you'll be asked to put the file path. So enter the pdf file's path you want to get answers from (note you will have to enter .pdf as well), so, for example, if your file is named "Neural networks", then you should enter "Neural networks.pdf" in the prompt shown in fig-6.

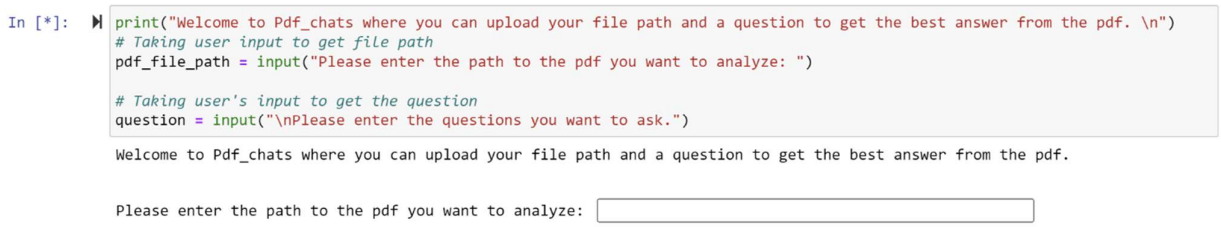

Fig-6: Taking user's input for file path

The next Prompt will ask for the question you want to know the answer from the pdf path you provided. And the fig-7 shows the prompt.

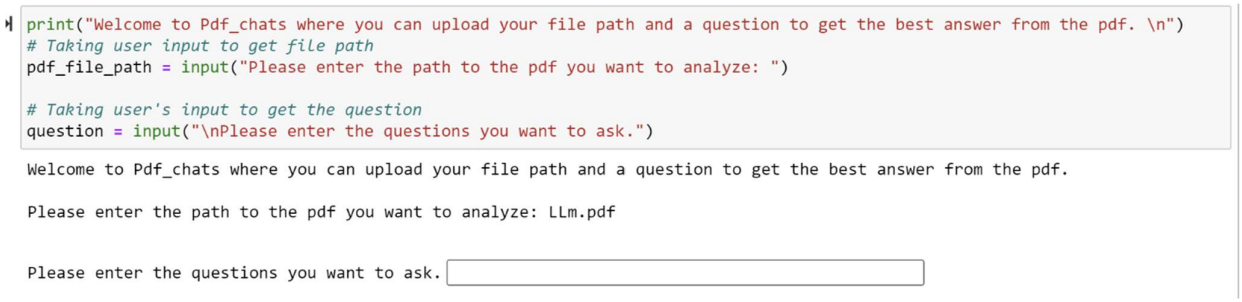

Fig-7: Taking User's input for the question they want to ask about the pdf

Now let's take a look at both different type of question types:

- **Direct Question:-** Question asked was "What is the hypothesis about alignment in this paper?" after preprocessing we passed the question and preprocessed data into our S-Bert model to give us top-relevant information. Fig-8 shows the Sbert model's output for this question.

	data	preprocessed_data	similarity_score
46	Figure Model outputs from test prompts that as...	figure model outputs test prompts ask model ge...	tensor(0.7148)
4	You are evaluating a response that has been su...	evaluating response submitted particular task ...	tensor(0.6735)
21	Then print the choice only from without quotes...	print choice quotes punctuation line correspon...	tensor(0.6157)
24	The placeholders task and submission will be r...	placeholders task submission replaced specic d...	tensor(0.6069)
23	Figure Prompt for ChatGPT evaluation with a sc...	figure prompt chatgpt evaluation scale likert ...	tensor(0.5824)
0	LIMA Less Is More for Alignment Chunting Zhou ...	lima alignment chunting zhou answer answer b c...	tensor(0.5694)
3	Figure Human annotation interface.	figure human annotation interface	tensor(0.5519)
26	Createamarketingplanwiththefollowingelements M...	createamarketingplanwiththefollowingelements m...	tensor(0.5352)
20	Avoid simply stating the correct answers at th...	avoid simply stating correct answers outset	tensor(0.5352)
22	At the end repeat just the selected choice aga...	end repeat selected choice new line	tensor(0.5160)

Fig-8: S-Bert output for question-1 (Direct Question)

Now before we pass our data into Chatgpt we need to calculate the number of tokens used by this data, and to achieve that I have used the tiktoken library recommended by

Openai. Fig-9 shows the number of tokens for the above data.

```
In [50]: # For getting the total amount of tokens passed into Chatgpt we are using Tiktoken library recommended by openai
         # I'm first saving the top10 outputs from the S-Bert model into a new dataframe and then here I'm changing that dataframe int
         # Since this function only accepts strings and the second argument is the encoding model.

         print( "Number of token we are passing into chatgpt are: ",num_tokens_from_string(' '.join(df_top10['preprocessed_data'].toli

         Number of token we are passing into chatgpt are:  113
```

Fig-9: Number of tokens generated by S-Bert's output

Next, we pass the data into chatgpt and receive our final output as shown below.

```
print("Question asked by the user was: ", question)
# Print the answer
print("Answer Generated by ChatGPT is: ",answer)

Question asked by the user was:  what is the hypothesis about alignment in this paper?
Answer Generated by ChatGPT is:  Answer: The hypothesis about alignment in this paper is that a more concise answer is gener
ally better than a more detailed answer.
```

Fig-10: ChatGPT's answer to the question asked

Now let's look at the output for another direct question in fig-11.

```
print("Question asked by the user was: ", question)
# Print the answer
print("Answer Generated by ChatGPT is: ",answer)

Question asked by the user was:  what is the experiment setup of this paper?
Answer Generated by ChatGPT is:  Answer:
The experiment setup of this paper involves using Figure Model outputs from test prompts to generate according to specified
structures, evaluating responses submitted for a task using a set of standards, creating a marketing plan with the specified
elements, prompting ChatGPT evaluation with a Likert score, printing the choice only from without quotes or punctuation on i
ts own line corresponding to the correct answer, using a human annotation interface with placeholders for task and submissio
n details, providing partial information or information that may not be useful for the user's needs, and using LIMA wo Forma
t Constraint Examples and Somewhat helpful examples to judge the relevance of the generated text.
```

Fig-11: ChatGPT's answer for the second question.

Based on these 2 outputs you can tell our models are working great for Direct questions, as S-Bert can get semantic similarity between the text quite easily and give the best matching text from the pdf, which can then be tokenized and passed into chatgpt through Openai API.

- **Indirect Questions:-** Similar to the direct question we first pass the question and preprocessed data into S-bert. Here the question was "What is the main discovery of this paper?" Fig-12 shows S-bert's output.

	data	preprocessed_data	similarity_score
4	You are evaluating a response that has been su...	evaluating response submitted particular task ...	tensor(0.6307)
21	Then print the choice only from without quotes...	print choice quotes punctuation line correspon...	tensor(0.5646)
12	It offers additionalinformationorexplanationsth...	oers additionalinformationorexplanationsthatar...	tensor(0.5629)
46	Figure Model outputs from test prompts that as...	figure model outputs test prompts ask model ge...	tensor(0.5474)
19	Does the submission meet the criterion First w...	submission meet criterion write step step mann...	tensor(0.5449)
17	It offers additional information or explanation...	oers additional information explanations usefu...	tensor(0.5446)
6	It does not provide any useful information to ...	provide useful information user	tensor(0.5408)
14	Itoffers additional information explanations or...	itoers additional information explanations ana...	tensor(0.5313)
15	However the structured of the response is not ...	structured response wellorganized clear progre...	tensor(0.5280)
7	Somewhat helpful The generated text has some r...	somewhat helpful generated text relevance user...	tensor(0.5117)

Fig-12: S-Bert output for question-1 (Indirect Questions)

Now I add the trigram most used phrases to the data and then pass it into tiktoken for getting the number of tokens.

```
In [69]:  # For getting the total amount of tokens passed into Chatgpt we are using Tiktoken library recommended by openai
          # I'm first saving the top10 outputs from the S-Bert model into a new dataframe and then here I'm changing that dataframe int
          # Since this function only accepts strings and the second argument is the encoding model.

          print( "Number of token we are passing into chatgpt are: ",num_tokens_from_string(' '.join(df_top10['preprocessed_data'].toli

          Number of token we are passing into chatgpt are:  103
```

Fig-13: Number of tokens generated by S-Bert + trigram(top 5) output

Let's look at chatgpt's output for indirect questions in fig-14 & 15.

```
print("Question asked by the user was: ", question)
# Print the answer
print("Answer Generated by ChatGPT is: ",answer)

Question asked by the user was:  what is the main discovery of this paper?
Answer Generated by ChatGPT is:  Answer: The main discovery of this paper is that it offers additional information, explanati
ons, or analogies that are not only useful but also insightful and valuable to the user.
```

Fig-14: Chatgpt's answer to the indirect question based on our input

```
print("Question asked by the user was: ", question)
# Print the answer
print("Answer Generated by ChatGPT is: ",answer)

Question asked by the user was:  how to explain the phenomenon observed in this paper?
Answer Generated by ChatGPT is:  Answer: The phenomenon observed in this paper is the evaluation of a response submitted for
a particular task using a specific set of standards. This evaluation can take the form of a figure model output from a test
prompt that asks the model to generate according to specified structures, a human annotation interface, a prompt for ChatGPT
evaluation with a scale Likert score, or simply stating the correct answers at the outset. In addition, it offers additional
information or explanations that are useful for the user. The evaluation process involves determining if the generated text
has some relevance to the user's question but may be unclear or incomplete (somewhat helpful).
```

Fig-15: Chatgpt's answer to the second indirect question based on our input

You can observe that indirect questions' outputs are not as accurate as direct question answers. Reasons are already mentioned in the above section where we discussed the limitation of the indirect questioning system. To improve the output we used Sumy library which can used to summarize any text data. Based on the Sumy's LexRank model we will generate an output which will then be passed into tiktoken to get the size of token we are passing into chatgpt. At the end we will receive an answer from chatgpt based on the data we provide. Fig-16,17 & 18 shows the working of sumy till chatgpt's output.

```
for sentence in summary:
    print(sentence)

To enable this transfer, various methods for aligninglanguage models have thus been proposed, primarily focusing on instruct
ion tuning [Mishra et al., 2021, Wei et al., 2022a, Sanh et al., 2022] over large multi-million-example datasets [Chung et a
l., 2022, Beeching et al., 2023, Köpf et al., 2023], and more recently reinforcement learning from human feedback (RLHF) [Ba
i et al., 2022a, Ouyang et al., 2022], collected over millions of interactions with human annotators.
LIMA0%25%50%75%100% 50% Excellent38% Pass12% Fail Figure 3: Analysis of LIMA over 50 test prompts.Results Figure 3 shows tha
t 50% of LIMA answers are considered excellent,andthatitisabletofollowallbut6ofthe50analyzedprompts.
```

Fig-16: Output from sumy's LexRank model

```

print( "Number of token we are passing into chatgpt are: ",num_tokens_from_string(str(sentence), "cl100k_base"))

Number of token we are passing into chatgpt are:  68
```

Fig:17: Token size for sumy's output

```
print("Question asked by the user was: ", question)
# Print the answer
print("Answer Generated by ChatGPT is: ",answer)

Question asked by the user was:  what is the main discovery of this paper?
Answer Generated by ChatGPT is:  Answer: The main discovery of this paper is that LIMA is able to follow all but 6 of the 5
analyzed prompts with 50% of answers considered excellent.
```

Fig-18: Chatgpt's answer for the question based on sumy's output

References:

[1] Kotamraju, S. (2022) *An intuitive explanation of sentence-bert*, *Medium*. [Link: https://towardsdatascience.com/an-intuitive-explanation-of-sentence-bert-1984d144a868]

[2] *OpenAI platform* (no date) *OpenAI Platform*. [Link: https://platform.openai.com/docs/introduction]

[3] Garfield E, Merton RK (1979). Citation Indexing: Its Theory and Application in Science, Technology, and Humanities.vol8. Wiley, New York.

[4] Cui, Haochuan & Zeng, An & Fan, Ying & Di, Zengru. (2020). Identifying the Key Reference of a Scientific Publication. Journal of Systems Science and Systems Engineering. 29. 10.1007/s11518-020-5455-3.

[5] Foster, K. (2023) *Text summarization for NLP: 5 best apis, AI models, and AI summarizers in 2023*, *News, Tutorials, AI Research*. [Link: https://www.assemblyai.com/blog/text-summarization-nlp-5-best-apis/#:~:text=What%20is%20Text%20Summarization%20for,into%20their%20most%20important%20parts]