CHAT PDF

Melinda Dong | a1870910@adelaide.edu.au

1. OVERVIEW

The aim of this project is to develop a question-answering system similar to ChatPDF(https://www.chatpdf.com/). But our system is mainly focused on research papers in the field of AI.

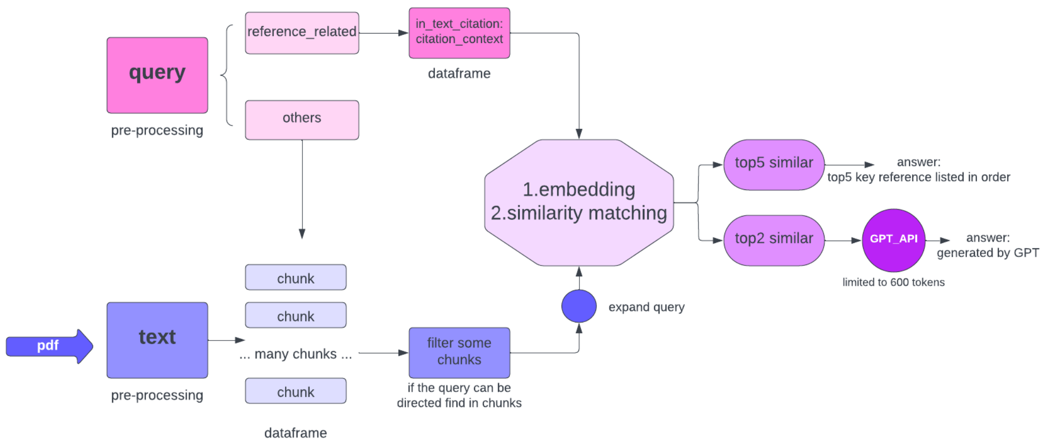
Our question-answering system is able to handle 3 kinds of questions related to PDF:

1. Direct query questions that can be matched to the text in the paragraph.

2. Indirect query questions without specific keywords in the text.

3. Identification of key references that inspire the proposed methodology in the paper.

This is the workflow of the whole system:



[There are 3 submitted files:

`main.py` and `functions.py` are the final running code, users can interact with the system in terminal.

`final_test.ipynb` is the final test version on Notebook, where shows more results of the intermediate steps.

Note: pre-trained glove file is needed to run the code; openAI API key is set as an environment variable and you need to replace it with your actual key to run.]

- BERT.pdf (https://arxiv.org/pdf/1810.04805.pdf) is used as a test sample for experiments in the system. 2 more new papers interaction result can be found in appendix.

2. DATA PREPROCESSING

2.1 extract text from DPF

`PdfReader` library is used for this part. Charts and graphs are not able to be read, tables are read as column by column.

There are some words looks wired because of the typesetting of the original paper, I replaced newline characters with spaces and remove the "-" inside the words that was originally caused by the newline problem.

2.2 main texts process

1. There are many in-text citations, which bring noises for machine to understand the meaning of the sentences, do I extract them by use regular expression(there might be various in-text citation styles, the regular expression can only match some mainstream style, like “(Radford et al., 2018)”, “(Peters et al., 2018a; Rad- ford et al., 2018)” and “[HLW+20]”, “[GSL+18, NK19]”).

2. Normally speaking, there’s a `References` section in research paper, this section contains a lot of texts that is less meaningful for the whole paper content understanding. Therefore, `References` section is removed, but sometimes the paper has `Appendix ` section behind the `References` section, which contains some useful information, so when removing `References` section, I’ll check if there’s `Appendix ` section, if yes, I’ll keep the `Appendix ` section.

3. In a reach paper, most of the structure information is captured by the section title, like “Abstract”, “Introduction” and “Conclusion”. But title and author structure cannot be located because there’s no label for title and author, so when the query ask about title or author, it’s difficult to locate in right chunks, so I add the text “title” and “author” at the beginning of the text for better localization.

4. After all cleaning and adding, the raw text is divided into small chunks and construct a data frame. each chunk has 300 tokens with the first 50 tokens overlapped with previous one. Overlapping design can solve the problem that if the separate part is inside a sentence, make each chunk more coherent. 300 token is a good size, a normal sentence has around 20-40 tokens, 300 tokens can express a piece of idea and not too long.

5. All the chunks are put into a data frame with index. For each chunk, text pre-processing is applied. Pre-processing part including:

- Remove punctuations.
- Convert into lower case.
- Remove stop words.
- Tokenization.
- Lemmatization.

2.3 Construct raw_citation_context dataframe.

Because direct judging whether a reference is important or not is difficult, but the context of the how it is be cited can give us some hint. For each of the in-text citation that I extract previous, get the context of it. The context starts from the first "." before the in-text citation to the first "." after the in-text citation. Save the in-text citation and its corresponding context into raw_citation_context dataframe for later use.

3. QUERY CLASSIFICATION

Depending on which query it is, the system has different way to deal with that. Here is the workflow:

1. apply text preprocessing on query (same as previous text pre-processing)

2. remove tokens "paper", "text", "article" and "thesis". These words appear in questions very often, but it does not contain meaningful information, remove them to reduce noise.

3. check if the query contains tokens like "reference", "references", "citation" and "citations". If yes, raw_citation_context data frame is used for searching and matching for next step.

4. simple filtering if the chunk contains any of the tokens from query, we called these chunks as `selected_rows`. It's a way tell whether the query's answer can be directed found in text or not. Filtering some chunks before embedding can improve the efficiency.

5. Expanding the query can to better similarity matching. The system use `wordnet.synsets ` to expand the query. Expand query is to find the synonyms for each token of query and add some of the synonyms together as a fixed list of tokens to do similarity matching. For example, for the sample question "What performance did BERT achieve on natural language processing tasks?"

Before expanding the query, here're the query tokens:

['performance', 'bert', 'achieve', 'natural', 'language', 'processing', 'task']

After expanding the query, here's the query tokens:

['performance', 'bert', 'achieve', 'natural', 'language', 'processing', 'task', 'carrying_out', 'attain', 'instinctive', 'language', 'treat', 'project']

4. SENTENCE EMBEDDING

From previous assignment, we found out the sentence embedding and similarity matching based on paper <A SIMPLE BUT TOUGH-TO-BEAT BASELINE FOR SENTENCE EMBEDDINGS > gives good performance on large scale data. Therefore, this system applied this method to do embedding and matching.

This system is using the pre-trained word-embedding from glove. To save the computing, the version Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d) has been used.

After sentence embedding, cosine similarity is used to count the degree of similarity of 2 sentence embeddings.

Compared with averaging sentence embedding, there are 2 innovative points of the new embedding method. The first is using SIF (Smooth Inverse Frequency) to mitigate the impact of high-frequency words. It helps emphasize the importance of less frequent and more meaningful words in the sentence representation and good at handling out-of-vocabulary words by ignoring them during the weighting process.

The second is using SVD (Singular Value Decomposition). SVD is a matrix factorization technique that helps extract meaningful information from high-dimensional data. It subtracts the projection of sentence embedding onto the first singular vector which capture the most dominant meaning of the whole question library, which efficiently remove the global sentence-level information, allowing the sentence embedding to focus more on local and context-specific information.

The final system uses top 5 similarity matchings to return the key references of the paper for reference related query. And top 2 similarity matchings to send to GPT API to generate answers for other queries.

5. KEY REFERENCE FINDING

If the query is reference related, the similarity matching would apply on raw_citation_context, there are 2 columns "in-text citation" and "context", through the matching of "context", get the top5 in-text citation. Some of the citation is single, some are more than one, for example (Peters et al., 2018a; Rad- ford et al., 2018). If it is more than one, use regular expression to strip it, then for every unique citation count the frequency it appears in top 5 citations, return the top 5 unique frequent citations in order as return (Note: after stripping the grouped citation, there might be more than 5 single citations).

6. GPT API USE AND INTERFACE BUILDING

The model the system used is "gpt-3.5-turbo". And here's the chat history:

```
chat_history = [
    {'role': 'system', 'content': 'You are a helpful assistant.'},
    {'role': 'user', 'content': 'Here are some texts from a paper: ' + paragraph},
    {'role': 'assistant', 'content': question}
  ]
```

For the interface building, it's an interactive interface in terminal. There are 2 parts, greeting part and conversation loop part. For greeting part, there are some welcome messages, and system will ask a pdf file choosing, in order to make sure it's a valid type of file, the file name has been provided with `.pdf` suffix. Also make sure the pdf file is in the path of the python file, otherwise, it will return `file doesn't exit` warning. For the conversation loop, it will keep asking "what is your question" and provide answer till user put `quit` to quit. Normally the response time won't be too long, if sometimes it stuck, it might be stuck with the GPT API, and you need to restart the terminal to use the system.

7. EXPERIMENTS

6.1 Different size of tokens.

Each chunk has 250 tokens, I did experiment on whether different size of tokens sent to GPT API would affect the result and how much it would affect the result.

Here's a sample question based on BERT.pdf:

Q: What performance did BERT achieve on natural language processing tasks?

250 tokens * 1: The text does not provide specific information on the performance of BERT on natural language processing tasks.

250 tokens * 2: BERT achieved an impressive performance on various natural language processing tasks. For example, on the CoNLL-2003 Named Entity Recognition (NER) task, BERT achieved an accuracy of 88.4% on the development set. Additionally, on the MNLI-m dataset, BERT achieved an accuracy of 79.8%, and on the MRPC dataset, it achieved an accuracy of 82.2%. BERT also achieved an accuracy of 90.7% on the SST-2 dataset.

250 tokens * 3: The paper does not provide specific performance numbers for BERT on natural language processing tasks. It mentions various tasks such as natural language inference, machine translation, and named entity recognition, but the results are not provided in the excerpts you provided.

300 tokens *2: According to the provided texts, BERT achieved the following results on different tasks:

- On the MNLI (Multi-Genre Natural Language Inference) task, BERT achieved a performance of 84.2% when fine-tuned, 95.4% when fine-tuned with the MASK strategy, and 94.9% when fine-tuned with the feature-based approach.

- On the NER (Named Entity Recognition) task, BERT achieved a performance of 84.3% with the MASK strategy, 94.9% with the same strategy, and 94.0% when using the RND strategy.

- Ablation experiments were performed on different masking strategies, and it was found that fine-tuning was robust to different strategies. However, using only the MASK strategy was problematic for the feature-based approach in NER.

These results were obtained after fine-tuning BERT on various NLP tasks.

From above, we can see one chunks of 250 tokens is unable to catch the information we need because its capacity is low. Multi-chunks can better combine information even if the answer is located in different paragraphs. But 3 chunks of 250 tokens doesn't give good performance as well, it might because, too many chunks would add not useful information and lower the weight of the real useful information. Therefore, 2 is the best number of chunks to send to GPT API.

We also see that 2 chunks of 300 tokens perform slightly better than 2 chunks of 250 tokens because for each chunk, the information it contains is more complete. So, the final system will use 2 chunks of 300 tokens to send to GPT API.

6.2 Expand query.

Based on 2 chunks of 300 tokens, here's the sample test on the difference of using expand query or not.

Here's a sample question based on BERT.pdf:

Q: What performance did BERT achieve on natural language processing tasks?

Without expand query: According to the provided texts, BERT achieved the following results on different tasks:

- On the MNLI (Multi-Genre Natural Language Inference) task, BERT achieved a performance of 84.2% when fine-tuned, 95.4% when fine-tuned with the MASK strategy, and 94.9% when fine-tuned with the feature-based approach.

- On the NER (Named Entity Recognition) task, BERT achieved a performance of 84.3% with the MASK strategy, 94.9% with the same strategy, and 94.0% when using the RND strategy.

- Ablation experiments were performed on different masking strategies, and it was found that fine-tuning was robust to different strategies. However, using only the MASK strategy was problematic for the feature-based approach in NER.

These results were obtained after fine-tuning BERT on various NLP tasks.

With expand query: BERT achieved new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement), and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

From the test sample, we can see using expand query can improve the accuracy of matching the answer related chunks. Because expand query add more flexibility while matching. It more focused on the meaning than the token itself.

6.3 Key references with GPT API

Here's a sample question based on BERT.pdf:

Q: what is the key reference of the paper?

Without API: The top 5 important references are:

(listed in order)

['Peters et al., 2018a', 'Radford et al., 2018', 'Bowman et al., 2015', 'Williams et al., 2018', 'Vaswani et al., 2017']

With default API prompt setting: The key reference of the paper seems to be the BERT model, as it is mentioned multiple times throughout the text.

I changed a little bit on the prompt design as following:
chat_history = [

{'role': 'system', 'content': 'You are a helpful assistant.'},

{'role': 'user', 'content': 'Here are some texts from a paper, the format is "in-text citations: the content of that citation;"' + paragraph},

{'role': 'assistant', 'content': question}

]

With designed API prompt setting: The key reference of the paper is Peters et al., 2018a.

Form above, we can see by using default API prompt settings, it's more likely to return the paper itself as the key reference of the paper. After designed a little on chat history, it can return a proper answer, but the answer it provides is still within the chunks we sent to API. Also, because the idea of `key reference` is subjective, it's better to give a flexible ordered list than one reference. Plus, without API using can reduce the cost, the system will directly using the top 5 similarity matching of the citation context to provide the reference related queries.

6.4* Other attempts

There are 2 more small attempts that I tried, the first is trying to use some open-source language model from hugging face instead of GPT API such as BERT and T5. As expected, the performance of these large language models is not as good as GPT.

Another attempt is for the query expand part, I tried to use the model 'tuner007/pegasus_paraphrase' from hugging face to paraphrase. Then add paraphrased query and original query together as the query to do similarity matching. The performance is similar with using `wordnet.synsets` to find the synonyms of each token of the query and using tuner model slow down the speed. So, the system sticks to the simple way to expand the query.

8. CONCLUSION

The cost of the system is: for reference related queries, it won't use GPT API, for other queries, each queries the system will send 600 tokens (2 chunks) to GPT API.

8.1 Highlights

- Using multi chunks, be able to catch the answer even if the answer is in different paragraph. Also 600 tokens compared with the whole pdf save a lot of cost from GPT API using.

- filter before similarity matching. Set a simple filtering before similarity matching can improve the efficiency and accuracy.

- use SIF and SVD to do sentence embedding and similarity matching. Compared with averaging sentence embedding and matching, it emphasizes the importance of less frequent and more meaningful words in the sentence representation and the sentence embedding is more focus on local and context-specific information, which improve the accuracy of matching the chunks where answers from.

- expand query to improve accuracy. There are many different expressions when users make queries. Using expand query can reduce the bias caused by the different expressions the user used and more focus on the meaning of the query and can matching the correct answer more accurate.

- be able to find the top 5 key reference of the paper. The system uses the in-text citation and its context and sentence embedding to find out the key references of the paper without using GPT API, which is something new and useful for searching AI papers.

8.2 Limitations

- graph and table information is not captured properly. The system can't extract graphs from pdf, it can extract tables, but the extracted tables are not in good format. These will make the extract text lose many useful information of the paper.

- the system uses the pre-trained glove embedding, but glove embedding is not specific trained for AI research paper area, so the top few similar chunks can not necessarily represent where the answers from. If the future work can train an embedding specifically for AI research paper area, that would improve the system performance.

- for some of the complex query, answers are located in more than 2 paragraphs, but the system can only get the top 2 chunks, which may lead to incomplete answers.

- the system is using GPT API, and there are some limitations of that. For example, it requires cost and sometimes it will have runtime error or crush. And for some query that need deep understanding of whole paper, the answer it provides might not be accurate.

- can only process one pdf at a time. The system is designed to process one pdf file at a time, if you need to process second pdf, you need to rerun the code.

9. APPENDIX

lima.pdf (https://arxiv.org/pdf/2305.11206.pdf)

rwkv.pdf (https://arxiv.org/pdf/2305.13048.pdf)

are used as new test samples for the system.

The test questions for the project are as follows:

Question type 1: (test on lima.pdf)

(1) What is the hypothesis about alignment in this paper?

A: The hypothesis in this paper is that alignment can be a simple process where the model learns the style or format for interacting with users, exposing the knowledge and capabilities that were already acquired during pretraining.

(2) What is the experiment setup of this paper?

A: The experiment setup of this paper involves comparing the performance of different models by generating a single response for each test prompt. Crowdworkers are then asked to compare the model outputs and label which one they prefer. The experiment includes comparing the LIMA model to five baselines: Alpaca 65B, OpenAI's DaVinci003, Google's Bard, Anthropic's Claude, and OpenAI's GPT-4. Responses from all baselines are sampled throughout April 2023. Nucleus sampling and a repetition penalty are applied during response generation, and the maximum token length is limited to 2048. Inter-annotator agreement is computed using tie-discounted accuracy, and comparisons are made between human, crowd, and GPT-4 annotations.

Question type 2: (test on lima.pdf)

(1) What is the main discovery of this paper?

A: The main discovery of this paper is that fine-tuning a strong pre-trained language model on carefully curated examples can produce competitive results on a wide range of prompts. However, there are limitations to this approach, such as the significant mental effort required to construct such examples and the potential for weak responses in certain cases. Nonetheless, the evidence presented in the paper demonstrates the potential of tackling complex alignment issues with a simple approach.

(2) How to explain the phenomenon observed in this paper?

A: The paper describes how a diverse set of prompts, including those mined from online communities and manually authored by the authors themselves, were used to train a model for generating helpful responses as an AI assistant. The authors hypothesize that setting a consistent format for the answers, acknowledging the question before providing the response, improves the model's performance. Additionally, the inclusion of prompts with some degree of toxicity or malevolence helps the model learn how to reject certain commands and explain why it will not comply. Furthermore, the authors sample examples from a natural language generation task dataset to add diversity to the training examples and increase the model's robustness. The paper emphasizes the investment in diversity and quality in creating prompts and responses, rather than focusing solely on quantity.

Question type 3: (test on rwkv.pdf)

(1) find the key reference for the following paper

A: The top 5 important references are:

(listed in order)

['Hendrycks et al., 2021', 'Hochreiter, 1998', 'Le and Zuidema, 2016', 'Biderman et al., 2023', 'Tay et al., 2020']

- Screenshots are provided to proof the above answer is generated from this system.