Midterm Challenge Written Report
Mike Dean (miketraindoc@gmail.com)
September 2024

The text below is reproduced in the main Jupyter notebook that I used to develop the initial app and carry out the tasks. The notebook is located at this address if you want to see the code that was used for each task.

https://github.com/mdean77a/AIE4/blob/main/Midterm/MidtermChallenge.ipynb

---

Task 1. Dealing with the Data

(Role: AI Solutions Engineer)

Task 1 Deliverables:

1. Describe the default chunking strategy that I will use:

The default strategy will be to load the two PDF files using `PyMuPDFLoader` just as we have previously done. This results in each PDF page being its own document. I have checked sample pages with `tiktoken` and the token count per page is <1000, so these are small enough to just embed without further splitting. I saved these embeddings in `page_split_vectorstore`.

2. Articulate a chunking strategy that I will also test:

The disadvantage of the default strategy is that there is no chunk overlapping between the pages, and this might worsen the ability connect two pages that are both relevant to a query. So I recombine the page_content of all pages into a single string, convert it into a document, and split it with a chunk size of 800 and an overlap of 400 (the default settings used by OpenAI). This strategy allows chunks to overlap, pehaps adding semantic continuity between adjacent pages. These were embedded with the same embedding model and saved in `chunk_split_vectorstore`.

3. Describe how and why I made these decisions:

The default behavior of `PyMuPDFLoader` is not bad and I have been using it for several months. However, I was splitting each of the documents created, not thinking through that if I had chunk sizes greater than the page itself, this was meaningless. I also had chunk overlap, but had not thought through the implications of each page being a separate document. So I made these decision for this Midterm Challenge so I can later compare the performance using RAGAS in Task 5.

Task 2. Building a Quick End-to-End Prototype

(Role: AI Systems Engineer)

Task 2 Deliverables:

1. Build a live public prototype on Hugging Face, and include the public URL link to my space.

Here is a one minute Loom video that demonstrates the prototype running in Hugging Face. https://www.loom.com/share/70b741d3e4e14af792572b3aa9106463?sid=5aeb2e51-0f75-4c74-9f8c-6bc6d14aaa52

I used the page split retriever for this prototype, meaning that the documents were broken by page, not recombined, and were chunked as whole pages without overlap.

2. How did I choose my stack, and why did I select each tool the way I did?

My stack consists of the following:

- Hardware is Apple Mac Studio
- Editor is VSC as recommended, and I have grown to like it very much because it includes everything.
- Qdrant is the vector store. I have used FAISS and Chroma, but Qdrant is fantastic. I run it locally as a server, though for this Hugging Face situation, I am using a memory-based implementation.
- ChainLit is the interface, as recommended. Notably, the current version of ChainLit does NOT work on Hugging Face, and the version needs to be locked (chainlit==0.7.700).
- RAGAS is part of my stack for purposes of later evaluation.
- LangChain is used to help simplify the code.

I learned an important lesson about software engineering - notebooks are NOT good ways to organize! They are great for teaching. So I reverted to my last 30 years of software development, and started refactoring code OUT OF THE NOTEBOOK, and then calling it inside the notebook. So you (or others) can use the notebook as a navigation tool, but it doesn't become so unwieldy that you can't figure anything out.

Task 3. Creating a Golden Test Data Set

(Role: AI Evaluation and Performance Engineer)

Task 3 Deliverables:

1. Assess my pipeline using the RAGAS framework including key metrics faithfulness, answer relevancy, context precision, and context recall. Provide a table of my output results.

	Metric	separate_pages	total_chunked
0	faithfulness	0.870491	0.899317
1	answer_relevancy	0.922842	0.905958
2	context_recall	0.860000	0.915000
3	context_precision	0.893333	0.898333

I did the evaluation of my prototype model which was based on the PDF documents being divided by pages, but while I was here, I compared this with the other strategy, which was to recombine all the text and then split by chunks with overlap.

When the PDF document is separated by page, and then those pages are INDIVIDUALLY chunked or embedded, the context recall is somewhat less, but other parameters are not really striking. Both strategies need to be assessed later when we use the fine-tuned embedding model.

2. What conclusions can I draw about performance and effectiveness of my pipeline with this information?

- *Faithfulness*: Measures whether all claims or statements in the answer can be completely inferred from the context that was provided. The value is the percentage of claims that can be inferred over the total number of claims.
- *Answer relevancy*: Measures whether the answer is relevant to the question. It does not matter if the answer is actually correct - only that it directly answers the question without redundancy.
- *Context recall*: This measures whether the facts that are in the ground truth reference answer can be inferred from the context that was provided to the LLM. In a perfect situation, every statement in the ground truth should be able to be linked to the context.
- *Context precision*: This measures whether all the elements in the ground truth are in the highest ranked parts of the context.

Not really any significant differences here, though I suspect that letting the pages be kept as separate documents is going to be inferior in the long run.

Task 4. Fine-Tuning Open-Source Embeddings

(Role: Machine Learning Engineer)

Task 4 Deliverables:

I fine tuned the model in a separate notebook because I thought I would need to use Colab to process it (I did not need to as the training proceeded on my Apple Silicon - to my complete surprise.). The notebook is located here:

https://github.com/mdean77a/AIE4/blob/main/Midterm/FineTunePartTwo.ipynb

1. Swap out my existing embedding model for the new fine tuned version. Provide a link to my fine-tuned embedding model on the Hugging Face Hub.

https://huggingface.co/Mdean77/finetuned_arctic

2. How did I choose the embedding model for this application?

I selected Snowflake/snowflake-arctic-embed-m because it has reasonable performance and improved dramatically in our previous exercise with it.

Task 5. Assessing Performance

(Role: AI Evaluation and Performance Engineer)

Task 5 Deliverables:

1. Test the fine-tuned embedding model using the RAGAS frameworks to quantify any improvements. Provide results in a table (combined table below).
2. Test the two chunking strategies using the RAGAS frameworks to quantify any improvements. Provide results in a table (combined table below)

	Metric	TE3 paged	TE3 chunked	ARCTIC paged	ARCTIC chunked
0	faithfulness	0.885625	0.870754	0.900033	0.900109
1	answer_relevancy	0.961167	0.909941	0.909930	0.947806
2	context_recall	0.850000	0.825000	0.870000	0.831667
3	context_precision	0.898889	0.874444	0.875556	0.862222

Previous results earlier with different chunk sizes (using the TE3):

	Metric	separate_pages	total_chunked
0	faithfulness	0.870491	0.899317
1	answer_relevancy	0.922842	0.905958
2	context_recall	0.860000	0.915000
3	context_precision	0.893333	0.898333

3. The AI Solutions Engineer asks me "Which one is the best to test with internal stakeholders next week, and why?

The results in the tables above show significant variability. Of these four metrics, the faithfulness metric is probably the most important, and the fine-tuned Snowflake outperforms TE3. My gestalt from looking at these tables, and also common sense, suggests that combining the PDF pages and THEN chunking them is sensible. I may have compromised performance here compared with the separate pages because I made the chunks small - the separate pages were embedded together and have more content.

If you compare the TE3 chunked with the ARCTIC chunked, the ARCTIC model is clearly better (though context precision was the same). For our company, faithfulness is critical in this application that will be used by our employees. So my recommendation is that we use the fine-tuned model, but that we experiment with chunk sizes. The results of some preliminary experiments are here:

	Metric	600/100	800/400	1200/400
0	faithfulness	0.892048	0.931814	0.883423
1	answer_relevancy	0.935133	0.931497	0.951927
2	context_recall	0.915000	0.915000	0.920000
3	context_precision	0.882778	0.908333	0.901111

The other variable to consider is the k number of chunks returned by the retriever; faithfulness might be better improved by using small chunk sizes but returning a larger k number of chunks.

Task 6. Managing Your Boss and User Expectations

(Role: SVP of Technology)

Task 6 Deliverables:

1. What is the story that I will give to the CEO to tell the whole company at the launch next month?

Here is the script for our beloved CEO to deliver with feeling at the launch:

> “All of you all know, our company has been at the leading edge of software development for many years, and unless you have been living in a sound-proof closet, you are aware of recent developments in so-called artificial intelligence. Your kids are probably using ChatGPT to help them do their homework!

> "As we move forward with our software development, it is very clear that we need to incorporate some of these advances in our products, or we will fall behind our competitors. But these tools are new, and we have to incorporate them into our ecosystem carefully, or we will find ourselves on the bleeding edge instead of the leading edge. There are ethical considerations about these tools, as you have certainly heard about on the news. The Senate and the White House were in near competition several years ago to identify important issues and potential strategies to assure that these tools are used to improve the lives of Americans without causing harm to any of our citizens.
>
> "It therefore gives me great pleasure to announce that our software engineering team has developed an application that can help all of you think about AI, its implications, and how we might think about it as we move forward with new products. This application is called Responsible AI, and it will discuss the topic with you like a friendly colleague. It is familiar with the most important Federal government written documents on the subject, and can give you an opportunity to find out the latest thinking from our Federal partners. You will all receive an email with the link to the application so that you can try it out today!"

2. There appears to be important information not included in our build. How might we incorporate relevant White House briefing information in future versions?

The testing app that we are posting will help us determine the features most important to our employees, but I have used some simplified approaches in order to expedite deploying it. These approaches include using an in-memory database for the two documents that we have used for this roll out.

For our long term maintenance of this application, we will utilize a server based vector database that maintains information about the uploaded documents, including their source and dates of upload. When new information is made available, we will upload it into the vector database, which will be file-based and persistent. This will allow new information to be added without having to re-process the old information. To help assure that users receive the most up to date information, the retrieval mechanism will pull the dates of publication and upload as well as the relevant context, and the most recently updated documents will be prioritized for inclusion in the context.

If a completely new version of a previously uploaded document is published, then we will mark the old version as deprecated, and our retrieval mechanism will be able to ignore deprecated items. We could erase them, but this would not be ideal since there may be important older information that has not been superseded in the new document.

This maintenance of our vector embedded knowledge base is likely to be the main cost driver for the app after the initial few months.