User:

Hi, nice to meet you.

Assistant:

Hello! It's nice to meet you too. Thank you for taking the time to speak with me today about the Entry-Level RAG AI Engineer position. To start off, could you tell me about your experience with retrieval-augmented generation (RAG) pipelines?

User:

Sure, I'd be happy to. While I was at my previous job, I was in charge of developing a retrieval augmented generated pipeline, which involved taking in user submitted documents and allowing them to search through it. So the way that I did this is, I started off by having all the documents be accepted. I type checked them to make sure that the documents were of the correct LEC file format, which we accepted PDFs, Word documents, and as well as a couple of LEC open standards. And I would then extract all the text from these, have them all be vectorized and placed into the vector database, for which we use Chroma. And we then had an embeddings model compare similarity to the input query to the LLM, to retrieve the most relevant pieces of the document to then allow the user to view that part of the document.

Assistant:

Thank you for sharing that experience. It's great to hear about your hands-on work with RAG pipelines. Can you tell me more about the specific LLM you used for this project and any challenges you faced in optimizing its performance?

User:

Sure. For this task I made use of OpenAI's GPT-4 because this would not only allow the model to load in a large amount of data into its context window, as GPT-4 has quite a large context window, this would also allow the users to interact with a powerful model being to do complex advanced reasoning with their data. To do this, a particular challenge to optimize its output performance was we needed to generate a very high quality prompt that would allow it to not only understand the actual data added into its context window, but also be able to do complex reasoning with the user in a safe and controlled manner. So to do this, we structured the prompt into several parts that were short, but concise, where we first informed it that appending to the message that would be given as the system prompt, there would be additional information for it to look through and use accordingly depending on the situation when conversing with the user. We then had the LLM be prompted into a bunch of safety things, is the best way I can put it I guess, where we told it to never be rude, to always be understanding of the user's perspective and to avoid discussing certain topics such as politics and religion. And this way the LLM would be a much safer environment to converse with for the user.

Assistant:

I see. That's a comprehensive approach. How did you evaluate the effectiveness of your RAG system, particularly in terms of retrieval accuracy and relevance of generated responses?

User:

Sure. Very quickly on into the development cycle of this RAG system, we realized that there were two key points that led to the major contributing factor of the accuracy and relevance of the general responses. The first one is the quality of the embeddings model that is able to search through the vectorized database to find the most relevant content. For this we ended up just using OpenAI's own embeddings model instead of our own locally ran one because we initially attempted to save cost but it was not worth the drop in retrieval accuracy. And second of all, the relevance of the generated responses was mainly improved through higher quality prompting into the LLM's system prompt.

Assistant:

Thank you for that explanation. One last question: Given your experience, how would you approach optimizing the speed and cost efficiency of a RAG system in a production environment?

User:

Sure, the first and most important thing is to understand the specific needs of the production environment. If the environment in which the system will be used requires real-time conversation or real-time responses from the ILM, then I would give a bigger priority to smaller LLMs such as GPT 3.5 Turbo or even GPT 4.0 Mini, as these models are able to generate their responses a lot quicker even if the accuracy of the generated content is not as good. Whereas on the other hand, if the production environment was not very time-sensitive, then I would use larger models. However, the trade-off between the size of model is inversely proportional to the quality of responses that it generates and directly proportional to the cost inefficiency of the model. So for example, if I use a large model, it will generate a text slower and cost more, whereas if I use a smaller model, it will generate text faster and be more cost-efficient. So it's about understanding the right balance of performance to speed that is needed for the application and then using a corresponding model.

Assistant:

Thank you for your time and detailed responses. We appreciate you sharing your experience and insights. You'll hear back from us soon regarding the next steps in the interview process. Have a great day!