Retrieval Augmented Generation

Mollie Hamman

Retrieval-Augmented Generation

Retrieval Augmented Generation is a process through which someone gathers a bunch of source material, which are usually medium-length texts, then they try to retrieve information from them through a process of embedding the source material, storing it in a vector database and then asking specific questions about that source material. It is a very interesting process, and may Large Language Models use this technique in their chatbot systems when doing question and answer processes. This way the system is able to computationally give you a reasonable answer that can help the user ultimately retrieve information about a specific, chosen topic.

Glove

This is an embedding model which takes in a very large amount of words and finds linear sub-structures throughout the embeddings. It was very interesting to learn about and work with this embedding model because it taught me about how computers are able to digitize language and then find patterns within them that can translate to mathematical understandings of the relationships between the words. Learning about how these relationships work and form reminds me of how I learn concepts in my life, as connections between words are a constant part of the learning process for me. I loved working with this embedding model, and I hope to use it in future projects.

SBert

For this embedding model Sentence-BERT, I learned about positional embeddings, which was a very interesting concept for me. I liked learning about the word embeddings because they help the computer give meaning to each individual token, but allowing the computer to understand context seems like a challenging problem for me. Solving that problem by using something like a position embedding is such an interesting solution, as it helps the computer computationally store the information for how the actual position for each word or token influences the rest of the words in the sentence or text. I can see this model being better for large texts which are very context dependent, like medical information or law-related documents.

OpenAI

Working with OpenAI all semester, I can see that using this API is relatively easy and efficient if you are familiar with the process. Since I have been using this throughout different projects, it made using this model much easier. I can also see the downside, as this is not really a model suitable for using for offline processes.

QDrant

I used QDrant as a vector database. This was my first experience with QDrant, so it was interesting to become familiar with it this semester. When I downloaded it onto my computer, there was a bit of a learning curve with uploading and separating into collections, but once I got the hang of it, it seemed to be a very efficient and useful tool for storing information.

Comparing the Models

To compare the models, I would say that the biggest differences is the positional embeddings. Integrating that into SBERT really changed how that model performs overall. It was interesting to learn about Glove's linear placement of words and to visualize it through their website. [1] Although, I think without the positional context it can be difficult for a model to understand how each individual word or token acts in the greater context of the text. OpenAI uses positions as well, so it was interesting to learn about how it differs too. I think that their biggest advantage is the wide reach it has with their audience, as very many people are familiar with Chat GPT. Overall these models all have their strengths and weaknesses and are each good at certain tasks. Given this understanding, I now have a better understanding of when to use each embedding model in the future.

References

[1] https://nlp.stanford.edu/projects/glove/
[2] https://tinkerd.net/blog/machine-learning/bert-embeddings/
[3] https://openai.com/index/introducing-text-and-code-embeddings/
[4] https://dev.to/simplr_sh/comparing-popular-embedding-models-choosing-the-right-one-for-your-use-case-43p1