Step 3: Project Proposal

Tyler Freckmann, 2024

Background

Many organizations have a wealth of text data at their fingertips. Organizations would like to be able to make the content of these documents easily accessible to potential users.

For example, perhaps product managers would like to be able to easily extract insights about their products from the feedback that customers provide - through online comments, support tickets, customer call transcripts, or user research interviews.

Perhaps the technical writing team at a company wants to make it easy for users to get answers to product questions without having to manually search through lengthy product documentation.

Perhaps the publishers of academic journals would like to provide a way for researchers to find information relevant to them and their research beyond what can be found through traditional key-word search.

Each of these examples requires the automatic extraction of user-relevant insights from large collections of documents - not something that would be possible to do manually. This is where machine learning techniques such as natural language processing come in.

The Use Case

For this project, we will focus on the third example described above. Let's say you work on the publishing team for an academic journal and you'd like to provide researchers a way to find information that is relevant to them - beyond what they could find through key-word search.

You would like to provide readers with a bot that allows them to ask a question and receive an answer powered by the papers in your journal. The response would contain a summary of the information from the relevant papers as well as excerpts from the papers themselves.

This would allow readers to quickly find relevant information without having to comb through tons of abstracts from a key-word search.

The Data

The collection of academic papers we will use is from the NeurIPS Conference.

All papers from 1987-2023 have been downloaded and stored in a CSV file on AWS S3. The CSV file is ~1 GB in size and contains ~20,000 rows. Each row contains three columns:

- year - the year the paper was published
- name - the title of the paper
- text - the full text of the paper (including the authors, abstract, and citations)

I wrote a script to scrape this data from the NeurIPS Conference website. Information about that script, and the script itself can be found in this GitHub repository.

Proposed Solution

This project will leverage Retrieval-Augmented Generation. The technique has two steps:

1. (Done once during preprocessing): Compute the embeddings of all the papers.
2. (For each question):
    a. Compute the embedding of the question
    b. Search for the papers whose embedding is most similar to the question
    c. Generate a response using the relevant paper(s) as context

Compute Resources

The embedding computation and text generation both rely on Large Language Models. We will leverage OpenAI's LLMs for each of these tasks. Since this is a demo project and we want to minimize compute costs, we'll focus on models available in OpenAI's free tier. For the embeddings, we'll use *text-embedding-3-small* which will allow us to process a lot more papers per dollar. For answer, we'll use *gpt-3.5-turbo* - the most advanced text generation model available in the free tier.

Since most of the compute work will be done by OpenAI - our application just needs to be able to orchestrate the preprocessing and host and run the webapp for users to interact with the RAG system. A general purpose Amazon EC2 instance such as the T3.medium instance should be sufficient for this task.

Alternative LLMs may need to be explored if the rate limits for the OpenAI models don't scale well given the size of the data and the financial resources available. Other LLM services may be explored (e.g., Anthropic, AWS, etc.). Additionally, using a local model for embedding might be an option as well (since embedding has the highest amount of data that would need to be processed). The trade-off for using a local model is that GPU resources might be necessary to run the local model.