Musubi: LLM Coding Assessment

Blaze Kotsenburg
Github link

Overview

I began by reading the requirements and started with the synthetic data. Here, I thought that providing a detailed prompt to the LLM would help generate some synthetic data to use for the assessment. I did zero shot here as I didn't want to provide an example for every single classification type since I was also including the Policies in the prompt.
I used some langchain libraries and a Mistral model via Nvidia because I was already working on a side project messing around with it and that's what was fastest for me to use.

Classification

You can see in my code that I load the policy file and begin to parse it into chunks. It's not pretty and it's definitely brute force because I experienced some issues with my API keys in the beginning that burnt a good chunk of time.
The reason I broke it into chunks for each policy was so that I could embed each policy and store it in an in memory vector db that I could use for each sample of data. I figured in the real world, simply adding the policy in the prompt isn't very realistic as policies will probably contain a lot more tokens.
So, given an input, I use the similarity from the input against the vector db to get the best "document" back. I use the retrieved policy in a prompt with the user-generated (synthetic) data to pass along to the LLM.
I do think I could have done better enforcing structure on this, but was running low on time.

Evaluation

The evaluation portion is pretty straight forward. I iterate the synthetic data and compare the true label to the predicted label from my "classifier" function. One thing I would have liked to do is probably do a step to actually evaluate the quality of the synthetic data, and then at the end, possibly add a "judge" LLM call comparing the predicted output to the true label to reassess the final prediction.