LLMs for Google Maps

Stanford CS224N Project

Sukrut Oak
Department of Computer Science
Stanford University
`sukrut@stanford.edu`

Abstract

Our work describes how large language models (LLMs) can be integrated with Google Maps to provide users with a more intuitive experience with their maps. Our goal is to design a model in which a user can ask any question related to maps to an LLM assistant, and for the model to answer correctly using relevant information. Our main contribution is a finetuned model that lets the user ask any question, whether it be about a route that they plan to take or retrieve information like air quality in a certain area. For all of these possible questions, the LLM can interpret the request, find the correct information via APIs, and report the answer back to the user. Our main finding is that well constructed synthetic fine-tuning data is crucial to fine-tune the LLM assistant, and that synthetic fine-tuning data from GPT 4 enables the model to perform the best in almost every single category of requests.

1 Key Information to include

- Mentor: Yuhui Zhang

2 Introduction

Integrating an LLM for Maps for public use has never been done before. Currently, there are no methods for users to ask questions about a map in an intuitive and natural language oriented way, but we hope to change this. For example, a user near Pacifica, California could ask the LLM to find a route to the famous Taco Bell on the beach, summarize the route, ask questions about the air quality in the area, and ask for specific health recommendations for their young child and elderly parent in case of detrimental environmental conditions. In this work, we design and build a system that can accomplish all of these tasks by enabling the LLM assistant to understand each question, categorize the requests into specific categories, call the correct API, and successfully return the best responses to the questions. The key method we employ is fine-tuning an LLM model on synthetically generated data, and the model performance improves as the synthetic dataset becomes more precise, thorough, and diverse in examples.

While we were working on this task, Google announced generative AI for Maps on Feb. 1st, although they haven't yet deployed any models to the general public (Daniel, 2024). Other work has been done in the tangential field of cartography by allowing users to generate completely new maps based only on natural language inputs (Feng et al.). However, no work directly integrating LLMs with maps has been released before in a way that has these features openly available to the public, and this enables incredibly useful features for users. In a world where there is more and more emphasis on personalization and LLM assistants, this feature is crucial to satisfy user requests and can be a valuable feature for maps and voice assistants.

3 Related Work

There has been very little work done in the field of LLMs for Maps, and at the time of beginning this project, the only similar work done was a study on "Geographic Question Answering with Maps Leveraging LLM and Open Knowledge Base," in which the authors studied the limitations of using LLMs to generate maps (Feng et al.). In this work, the authors present an innovative workflow for users to generate completely new maps based only on natural language inputs. One key insight was that LLM data and prompts are unstructured while map data and formats are very structured, so it is difficult to have the two types of systems interact in a meaningful way. The authors attempt to solve this problem by using a 3rd party tool that can provide the data requested by the LLM, and output the data in a way that can be formatted back into a map since the LLM agent is not able to do this process itself. The main goal of their paper was to have the LLM be able to understand map data and produce its own visualizations of maps based on user inputs, which current day LLMs and AI agents cannot yet do on their own. ChatGPT, other LLMs, and even image generating AIs like Midjourney do not have the capabilities to produce correct maps that are true to the data and to standards that are commonly used in these applications related to geo-spatial data.

In the example the paper provides, a user asks the model to create a map of all the universities in Germany and China, and the model, through the process of using ChatGPT, SPARQL, and Wikidata, is able to generate a map completely from scratch that provides the correct information back to the user (Feng et al.). This is a very novel process that has not been done before, since the only previous attempt at this required the user to provide their own dataset to the model. In this new process, the user does not need to provide any such dataset to the model, and the model is able to access the information it requires on its own.

On Feb. 1st, 2024, Google announced their plans for deploying a generative AI model for their Google Maps service in an article titled "A new way to discover places with generative AI in Maps," which they have begun rolling out to select local guides (Daniel, 2024). In this article, they describe how users can ask natural language questions, and their LLM model will be able to understand the queries and respond using the existing Google Maps information including businesses, places, public photos, ratings, and reviews. Currently, the examples given don't have too much additional context, with examples like "places with a vintage vibe in SF" and "activities for a rainy day." However, this is incredibly exciting work and very similar to the long term goal we had in mind when starting the project at the beginning of the quarter.

4 Approach

In this project, we are attempting to have an LLM answer queries about maps and its associated data. To access this information, we use data from Google Maps accessible through APIs, including air quality, directions, locations, and other types of API information (Maps). The main goal of this project is to have the LLM understand different user queries and the type of information the user is requesting, choose the correct Google Maps API to call, format the Google Maps API call correctly by extracting the correct information from the user's query, and return the formatted response to the user. This is an important problem because users interact with maps frequently, and having an LLM to understand information about maps would be a helpful feature for many users.

The main problem we focused on is fine-tuning an LLM on Google Collab to choose the correct API to call based on the context around the user's query, and extract the information that is necessary for the API call parameters. For example, a user may ask about the route to a given location, but there are multiple Google Maps APIs for that topic, including routes API and places API. The LLM needs to choose the correct API depending on the information needed and call the API with the correct information. The goal of this project is to allow users great flexibility with their queries, enabling them to include added relevant and potentially superfluous information to the query, and still receive the correct information. Another example, which was used in the actual test set defined later in this paper, is "My friend and I are arguing about the fastest way to get to a new Italian restaurant downtown. Can you settle this with a quick summary of the route?" This user query has a great deal of contextual data, some of which is unnecessary, yet our fine-tuned models are able to interpret the type of information requested, extract the information needed for the API call, and return the correct type of information, which is not the time it takes or the location of the restaurant, but an actual summary of the path.

To achieve this solution, we first chose 10 categories of information that the user queries would ask about, forcing the LLM to choose between the categories. The 10 categories were Locations, Distance to Place, Type of Place, Summary of Route, Duration of Travel, Air Quality Category, Air Quality Recommendation, Air Quality Child Recommendation, Air Quality Elderly Recommendation, and Dominant Pollutant. Next, we generated three sets of synthetic data to fine-tune an LLM model. The first synthetic dataset was generated by hand, the second synthetic dataset was generated using GPT 3.5, and the third synthetic dataset was generated using GPT 4. The specifics of how these datasets were generated can be found in section 5.1, but as a brief overview, they were generated using API documentation and examples of how to format the prompt to the API correctly, specifically on the 10 categories. For the experiments, we chose GPT 3.5 as our baseline model, and we fine-tuned three additional GPT 3.5 models on the three synthetic datasets.

This approach is entirely original and the code was written by the author. For this work, we used the GPT 3.5 model and fine-tuned versions of GPT 3.5 and GPT 4 models (OpenAI).

5 Experiments

5.1 Data

Our project has two forms of data. The first is the original data available through the Google Maps API, and the second is the synthetic data used to fine-tune a LLM model on the data from the Google Maps API documentation.

Our project uses various points of data from the Google Maps database, and this project relies heavily on the Maps API and API documentation (Maps). The Google Maps API feature hosts a wide variety of data, especially about route planning and environmental conditions, and the tests all fell largely into these two broader categories. The input of the API calls were the parameters needed for the specific API call, and the output was a JSON file with a large amount of information provided by the API. The JSON output was parsed to find the information requested by the user, and returned to the user in a well formatted answer.

Our project also makes use of a large amount of synthetic data to fine-tune the GPT 3.5 models. We created 3 sets of synthetic data: 100 example queries and answers generated by GPT 4, 100 example queries and answers generated by GPT 3.5, and 20 example queries and answers generated by hand. To generate the two datasets using OpenAI's GPT Models, we prompted GPT-3.5 and GPT-4 with API documentation from Google Maps and asked the models to generate the datasets of queries and answers that could be answered by data available through the APIs. The third dataset of 20 custom examples was generated by hand.

5.2 Evaluation method

The metrics we use evaluate if the LLM correctly responded to the user query. If the LLM correctly categorizes the request by the user and calls the API using the correct parameters extracted from the user query, then the answer is marked correct. Otherwise, the answer is marked incorrect.

For this project, we created a test set of 100 queries, which each belonged to one of the 10 specified categories. All 4 models were tested on this test set, and the number of queries answered correctly was evaluated, manually, for each of the models. Further, the number of queries answered correctly was broken down into the 10 categories to understand for which categories the models were improving. For example, if a model answers a query in the the category of 'duration' and incorrectly categorizes the question into the category of 'distance', we marked the answer as incorrect for the 'duration' category.

With this quantitative evaluation metric, we can understand how the different fine-tuned models performed, and specifically for which categories and types of questions they struggled on. Through this evaluation method, we hope to understand where fine-tuning can improve results, and where fine-tuning has lower impact on correctly answering the user query.

5.3 Experimental details

For our experiment, we had 4 different LLM models and evaluated each of their performances. The baseline model for comparison was GPT 3.5, and specifically the "gpt-3.5-turbo" model. The other 3 models also use GPT 3.5 and the "gpt-3.5-turbo" model, but each of these models is fine-tuned on 3 different datasets of synthetic data. The model fine-tuned on custom synthetic data was fine-tuned with 5 epochs and roughly 13,000 training tokens, the model fine-tuned on GPT 3.5 generated synthetic data was fine-tuned with 3 epochs and roughly 38,000 training tokens, and the model fine-tuned on GPT 4 generated synthetic data was fine-tuned with 3 epochs and roughly 38,000 training tokens. The details of each model fine-tuning can be found in Table 1.

Model	Number of Epochs	Training Tokens	Training Time
Fine-Tuned on Custom Data	5	12,700	5:17
Fine-Tuned on GPT 3.5 Synthetic Data	3	37,779	15:10
Fine-Tuned on GPT 4 Synthetic Data	3	37,980	14:58

Table 1: Fine-Tuning Details of GPT 3.5 Models

Each of the GPT 3.5 models being fine-tuned achieved a training loss less than 0.55. The training accuracies, which were graphed on the same chart with values from 0 to 1, all achieved a value greater than 0.88. The details of the fine-tuning process can be found in Figures 1 - 3.

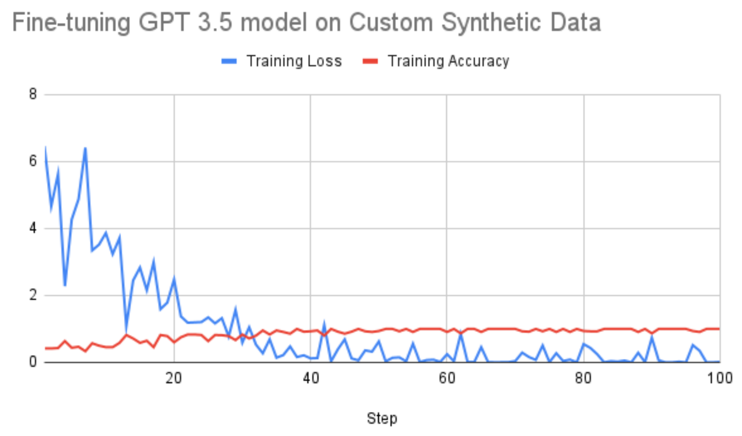

Figure 1: Fine-Tuning Details for Custom Synthetic Data

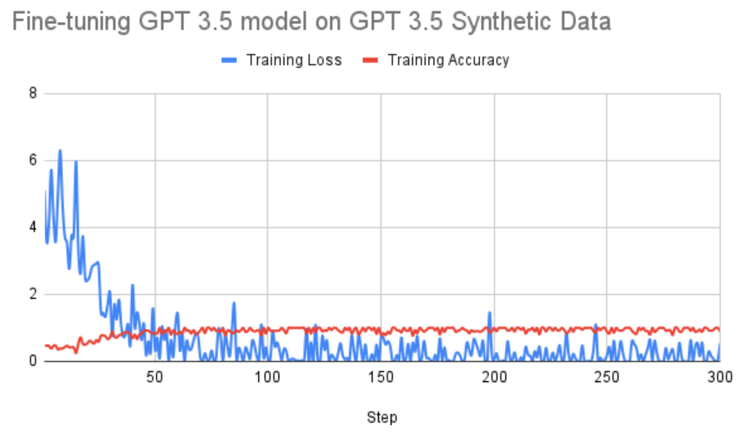

Figure 2: Fine-Tuning Details for GPT 3.5 Generated Synthetic Data

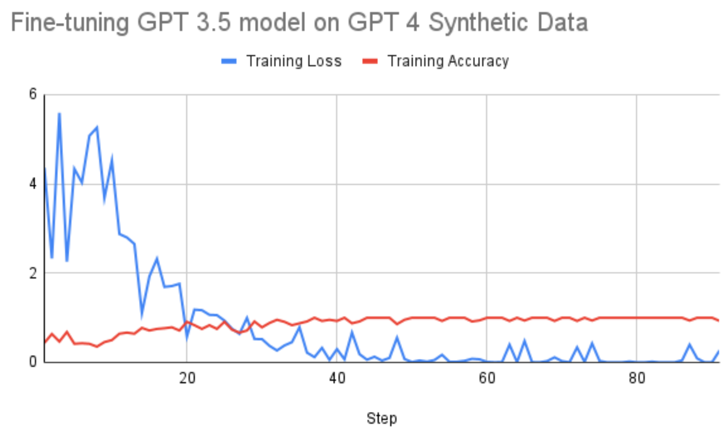

Figure 3: Fine-Tuning Details for GPT 4 Generated Synthetic Data

After these three models were fine-tuned, they, along with the GPT 3.5 baseline, were used in the existing framework and tested with the test set of 100 queries. The results of how each of these models performed will be displayed in the next section.

5.4 Results

After evaluating the test set on the 4 models, we marked how many of the 100 queries in the test set each model answered correctly. The GPT 3.5 baseline model answered 76 correctly, GPT 3.5 fine-tuned on custom synthetic data answered 79 correctly, GPT 3.5 fine-tuned on GPT 3.5 generated synthetic data answered 87 correctly, and GPT 3.5 fine-tuned on GPT 4 generated synthetic data answered 90 correctly. These results are summarized in Figure 4.

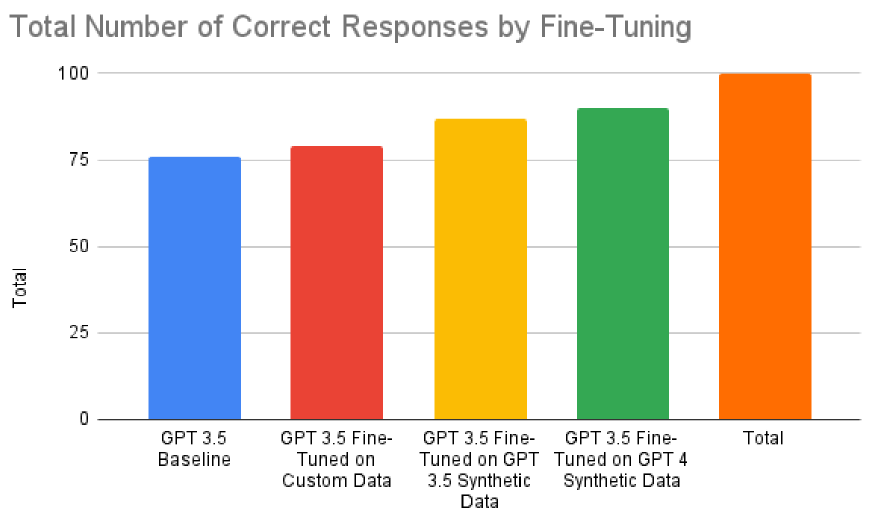

Figure 4: Total Number of Correct Responses by Model

Next, we broke down the correctly answered responses by the 10 categories for the 100 queries in Figure 5. The Orange bar represents the total number of questions corresponding to that associated category, and the four other colors represent the performances of the four different models we evaluated.

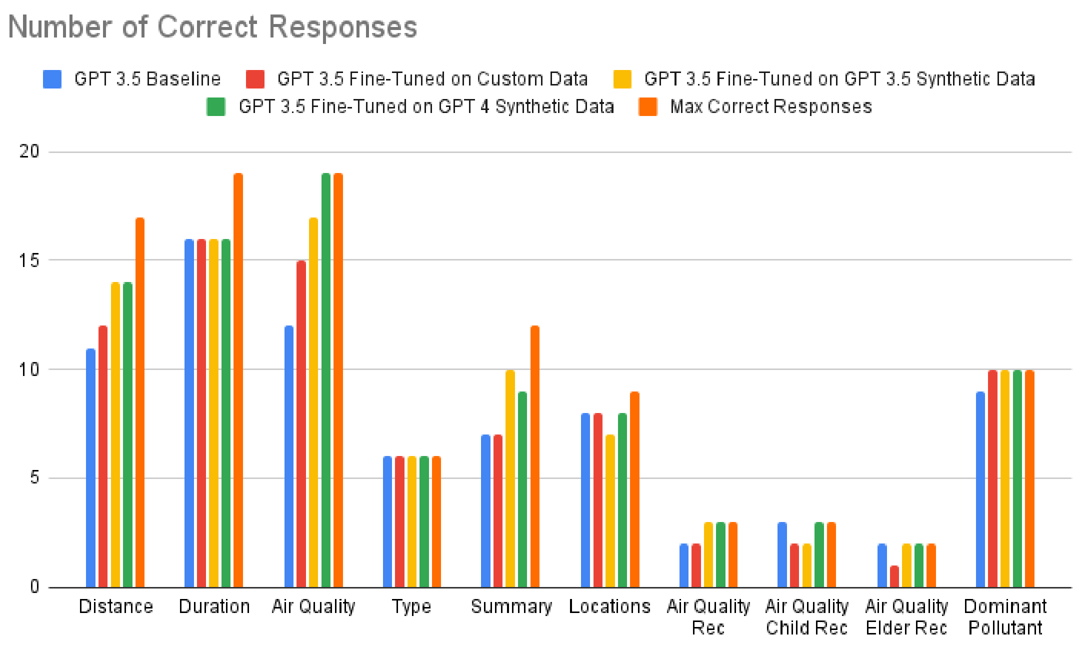

Figure 5: Number of Correct Responses by Category

These results are better than what we expected, especially given the difficulty of some of the questions in the test set. The pattern in the results is roughly what we expected, with most of the models performing better than the GPT 3.5 baseline across almost all the categories.

The model fine-tuned on custom synthetic data is 3.95% better than the baseline, and improvements maily stem from the distance and air quality related questions. The model fine-tuned on GPT 3.5 generated synthetic data is 14.47% better than the baseline, and the model fine-tuned on GPT 4 generated synthetic data is 18.42% better than the baseline, performing the best overall.

This pattern likely emerged because GPT 3.5 and GPT 4 are quite powerful in terms of summarization and generalization, and their synthetic datasets incorporated more niche examples that enabled their fine-tuned models to better answer the test queries. The custom synthetic dataset also likely performed worse because its dataset only had 20 queries and responses, whereas the GPT generated synthetic datasets both had 100 queries and responses. This was done so that we spent the same amount of time generating all three synthetic datasets, which allowed for fair treatment in terms of time taken to create the datasets. More details about why the overall patterns emerged and what they tell us about our approach can be found in the analysis section.

6 Analysis

Across almost all of the categories, the baseline performs the worst, fine-tuning with custom data performed better, fine-tuning with GPT 3 data performed even better, and fine-tuning with GPT 4 data performed the best.

We believe the GPT based fine-tuned models performed the best because the GPT generated synthetic data was created using context from the specific API documentation. Thus, the queries and responses in the synthetic data were able to incorporate more information from the API, including more niche API use cases. Also, it was likely able to understand what types of problems the API can answer and how to format the API calls correctly. By covering more edge cases, the synthetic data, generated by both GPT 3.5 and GPT 4, could better prepare for questions that were difficult to answer by extracting only the information it needed from the query and calling the API on that critical information. The baseline model and model fine-tuned on custom synthetic data had a weaker version of that capability, and extracted other, likely superfluous, information from the query and called the API, which often resulted in an incorrect response or a failed API response. The fine-tuning on GPT synthetic data, which itself was generated using documentation from the APIs, was able to understand what the critical information to extract from the query was, and thus answer correctly with a higher probability.

For example, the baseline model and custom data fine-tuned model failed on the following query, while both GPT data fine-tuned models were able to successfully answer the question: "After watching a lot of cooking shows, I'm excited to try making sushi at home. How far is the nearest Asian grocery store where I can buy seaweed?" This question asked for the distance, but the baseline incorrectly classified the request into the locations category. The custom fine-tuned model said the context as "nearest grocery store," which the API was not able to interpret. The GPT data fine-tuned models were able to identify the correct category of distance and the correct context of "grocery store," and the API was able to return the correct result. This tells us that fine-tuning using examples generated using the API documentation is critical for understanding and preparing for untraditional and difficult-to-answer queries from users.

7 Conclusion

The main findings of our work is that with carefully generated synthetic data for fine-tuning an LLM, it is possible to design a highly customized and personalized experience with maps. With further development, it will be possible for users to ask any question they choose, along any category related to maps and the environment, and the LLM will be able to answer. This paper focuses on this task for the 10 categories defined at the beginning of this paper, so there are limitations on what types of queries our LLM models can correctly answer. Incorporating more API and data source options would expand these capabilities and is an avenue for future work that we are excited about.

References

Miriam Daniel. 2024. A new way to discover places with generative ai in maps.

Yu Feng, Linfang Ding, and Guohui Xiao. Geoqamap - geographic question answering with maps leveraging llm and open knowledge base (short paper).

Google Maps. Google maps platform documentation | google for developers.

OpenAI. OpenAI Models.