Neuro 240 Final Project: Metric-Based LLM-Router

Gary Wu

Harvard Department of Neurobiology

1 Abstract

Inspired by the recent launches of numerous Large Language Models, as well as various projects from industry, I'm interested to see how feasible Large Language Model routers can be. An LLM-router can essentially be defined as an intelligent system that can recommend or "route" the user towards the best LLM given a task. I start small in this paper with just prompts, and build a KMeans Clustering Model to predict which LLM will have the best response given a certain prompt.

I have made my modeling code publicly available here: https://github.com/garywuuu/neuro240.

2 Introduction

With the enormous current interest in Large Language Models, I can imagine a world in which there are hundreds of large language models all serving different users for different purposes. It then becomes extremely difficult for everyday consumers to know which LLM's to prompt for a given use case, given that different LLM's will give better responses than others given a prompt.

Thus, I consider the possibility of creating an efficient 'LLM router' that can learn which LLM's might provide the best outputs for certain prompts given some criteria, and 'route' the user towards the correct model provider API. For this project, I will use the criteria of toxicity as measured by Meta's RobertaHateSpeech API. Additionally, I will only be comparing OpenAI's GPT and Anthropic's Claude for this project, and the model will just choose between the two models.

The overall user flow from start to finish from the user perspective is as follows:

1. User begins with some prompt or question that is of somewhat controversial nature (i.e. "should marijuana be legalized?") and wants to find the LLM that will give the least toxic response.

2. User enters prompt into some system that utilizes our clustering model, and the model redirects the user towards the LLM that it predicts will have the least toxic response given the specific prompt.

3. User receives output from the LLM that was chosen by our LLM-router.

I believe it is almost inevitable that more complex LLM-routers will exist in the future. Since users care about quality of responses, speed of responses, and price of responses for their LLM's, it makes sense that the most efficient way of interacting with LLM's will have to involve some sort of routing. The only world in which this doesn't exist is if there is just one LLM that wildly performs all others in every category and use case. This is highly unlikely in my opinion given how competitive the current top LLM's are, and if a super-powerful LLM like the one described existed, I could probably consider it an AGI - in which I would have much bigger problems than LLM-routing.

3 Related Work & Literature Review

This particular field of interest has little academic presence which makes sense: it largely revolves around the comparison of industry LLM models rather than foundational model improvement and other hot topics of the moment such as open-source. Nonetheless, this problem is important because as the number of LLM's increases, people will have to choose between more and more LLM providers. Making this process more efficient and productive is a key usability factor.

3.1 Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

This paper introduces Sentence-BERT, a model designed to generate semantically meaningful sentence embeddings [1]. It utilizes a Siamese architecture with transformer-based models to learn sentence representations by maximizing the similarity of semantically similar sentences and minimizing the similarity of dissimilar ones. This allows it to capture fine-grained semantic information and outperform traditional methods in multple natural language processing tasks, such as semantic textual similarity and paraphrase identification.

3.2 A Brief Review of Nearest Neighbor Algorithm for Learning and Classification

This paper provides an overview of how to use the KMeans-Clustering technique for classification and prediction [2]. KMeans does well when there is no previous information known about the dataset, and does well in simple environments and general purposes. The main idea is to find the correct placements of "clusters" in the data, meaning which data-points are similar to each other. The model trainer can decide how many clusters they wish to have in the end, and the model will be trained to determine k clusters. In order to predict classifications for new data-points, you can calculate which cluster a new-datapoint is closest to.

3.3 Evaluating Large Language Models: A Comprehensive Survey

This paper discusses the evaluation of LLMs across various dimensions, including alignment, safety, and reasoning tasks [3]. It highlights the importance of assessing LLMs' factual consistency, especially in text summarization, as LLMs may struggle to maintain factual accuracy in their summaries. The paper also presents several benchmarks and datasets used to evaluate LLMs' performance in these areas, such as the PersonaChat data for assessing consistency in conversational QA pairs and the XSumFaith benchmark for evaluating factual consistency in summarization.

3.4 Martian: Startup working on an LLM-Router

My project is most inspired by Martian, an LLM router [4]. By dynamically routing between multiple models, Martian can beat GPT-4 on performance, reduce costs by 20%-97% and simplify the process of using AI. Notably, Martian uses a combination of LLM's in creating their output, rather than choosing one specifically for the output (my focus).

4 Methods & Approach

I can break down this project into multiple steps:

1. **Dataset Creation**

   A few key points I put a lot of effort into thinking about included how to format the data, how to generate the prompts, and how to store the data in various forms (plain text versus embeddings).

2. **Training the Model**

   Given the dataset, a big question was what type of model I should be using. Additionally, I had to tweak the dataset in order to accomodate the end model chosen.

3. **Evaluations**

   It's pretty ambiguous as to how to evaluate the performance of this model. Explored different approaches.

4.1 Dataset Creation

There is no widely available LLM prompt input/output dataset for controversial questions, so I will create one as follows. I used the list of questions provided here: Controversial Questions. I then generate the responses from both GPT and Claude for each prompt (since they are questions) and store them both in .txt files.

Then, I run both the GPT and Claude outputs on the toxicity score API which can be found here (https://huggingface.co/facebook/roberta-hate-speech-dynabench-r4-target).

[prompt 1, 0 or 0.5 or 1]
[prompt 2, 0 or 0.5 or 1]
[prompt 3, 0 or 0.5 or 1]

...

Where 0 or 1 corresponds to whether GPT or Claude performed better on the given prompt, and 0.5 if the toxicity score was the same.

4.2 Model

To build this model, there are a few major steps to consider. Since I are trying to predict the best model to use (GPT or Claude) given a prompt, I need some type of system that can provide recommendations. The assumption here is that GPT responses are similar to each other and the

same applies for Claude. Thus, a clustering model should work well here, given that if a certain cluster predicts better for a certain model, you can feed a new prompt into the model and see which cluster it is closest to.

An industry standard for clustering models is the KMeans-Clustering technique. I decided to use a KMeans-clustering model for this project, by forming 2 clusters based on the data, and then finding which cluster a new data-point is closest to for prediction.

In order to work with a KMeans model, I need to pre-process our dataset slightly. First, I know that KMeans models typically work well with tuple-like data, i.e. in the form of (x, y) where x and y are both scalar values. Thus, I need to transform our dataset of prompts into two-dimensional values. In our literature review, I cite the Sentence-BERT model, which I use in our code to transform each prompt into a high-dimensional vector (I believe it is initially 11-dimensional).

However, performing KMeans on data of such high dimension is unfeasible given current techniques. Thus, I perform Principal Component Analysis (PCA) on the high dimensional vector embeddings to reduce the dimension of the data from 11 to 2. This is done with sklearn's PCA package.

I then train the KMeans-clustering model with the 2-dimensional vectors, again using the sklearn library. This gives us the location of the cluster centroids, which I can then compare distances with new datapoints. A tricky question is how to assign GPT or Claude to each cluster. To do so, I iterate back through each of the datapoints and see which cluster each datapoint is closest to. I then associate each cluter with the model that a majority of the datapoints are closest to.

4.3 Evaluation

After I have a fully trained KMeans-clustering model, I can assess the performance of our model on new test data. Since I don't use the entire dataset of controversial questions for training, I use the remaining for testing. I take the test dataset of questions and again run them through GPT and Claude and score them using the toxicity score API. I store the objective less-toxic model for each and then compare that to the model that our clustering model recommends. This is done through a euclidean distance measurement between the new prompt's embedding and the two clusters.

5 Model Results

Following all of the steps mentioned above, I obtain dimensionality-reduced embeddings and our trained clustering model:

```
1. Loading llm dataset



2. Reducing via PCA

Explained variation per principal component: [0.08135121 0.07024865]
Cumulative variance explained by 2 principal components: 15.16%


           0         1         2         3         4         5         6    \
PC_1  0.005087  0.042430  0.040403  0.008836  0.051632  0.025555  0.088601
PC_2  0.012122  0.027371  0.008571  0.001170  0.012711  0.040842  0.066688

           7         8         9    ...       758       759       760  \
PC_1  0.006648  0.005440  0.077636  ...  0.002880  0.055138  0.023199
PC_2  0.029500  0.021797  0.022329  ...  0.035396  0.020329  0.027152

           761       762       763       764       765       766       767
PC_1  0.090592  0.015942  0.003908  0.053799  0.035203  0.010521  0.007950
PC_2  0.057278  0.016125  0.038585  0.030538  0.063836  0.038757  0.041111

[2 rows x 768 columns]
```

Image 1: Dataset Prep using PCA & Calculating Explained Variance

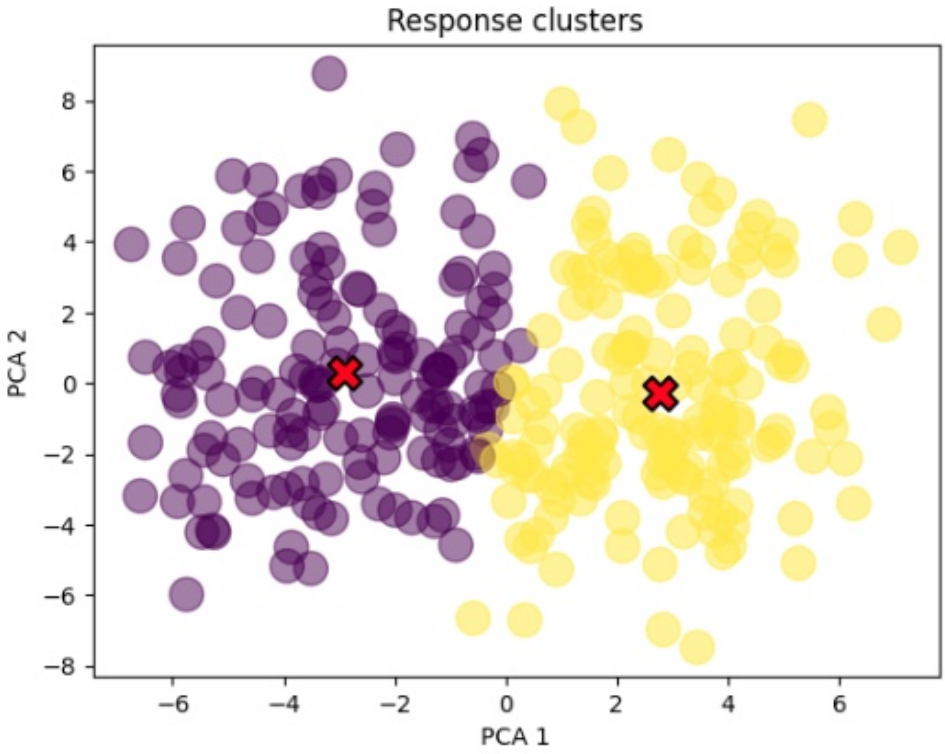

Image 2: Clustering Results & Centroids

Then, evaluating our K-Means Clustering model on our test dataset, I get the following results:

	Score after 10 examples	Score with 50 examples
LLM-router	60%	56%

Figure 1: Final Results

Given the nature of this project and the results, it can be a bit difficult to interpret these findings. Essentially, I find that over these 50 test examples, the LLM-router chooses the better-performing LLM-provider 60% of the time on the first 10 examples I had been able to calculate values for before deliverable 4, and after getting my hands on more credits I was able to run the evaluation on 50 more examples, with a 56% accurate performance. This is slightly better than random choice (50%), and if I consider using this type of model over a large number of prompts (say hundreds of thousands), then our model would have provided substantial improvement compared to random choice and saved users a large amount of time choosing.

6 Conclusion & Next Steps

Through this project, I have successfully created a clustering-based LLM router for guiding users to the best LLM for a specific purpose given some prompt. I first started by creating a novel dataset of controversial questions. Then I gathered responses from GPT and Claude and scored the responses based on a toxicity score API. I pre-processed our data by turning the prompts into vector embeddings using SentenceBERT such that they would be compatible with a K-Means clustering model, and then performing dimensionality reduction using PCA and taking 2 principal components. After, I trained a clustering model on this data and found the centroids. Through evaluating our model on novel test data and comparing our LLM-router recommendations with the true scores of both models, I found a noticeable improvement over random chance.

I now discuss a few potential next steps and potential improvements to this LLM-router system.

1. **Incorporating Multiple Metrics and More LLM's**

   This is an exciting introduction to the notion of LLM-routing, and a truly useful project or product would have to involve a broader set of metrics (apart from toxicity) and comparing many more models than just GPT and Claude, although they happen to be the current leaders among certain measures. Number of models supported would be an important consideration, as the number of models compared increases, the more design decisions would have to be made in regards to model type/architecture, etc.

2. **Explained Variance in Embeddings**

   One of the most difficult aspects of this project was increasing variance explained by 2 principal components. It's very, very difficult to represent the semantics of a sentence in vectors, which is why the initial dimensionality was so high. However, as I are trying to

compress that information into smaller and smaller vectors, it's harder to distinguish between sentences and also to capture the meanings of them. Thus, an interesting improvement to the system would be finding a method to maintain a higher percentage of variance explained by the PC's.

3. **Creating Larger Datasets**

   My dataset curation was heavily limited by cost concerns and as such, I think a continuation of this project could include a larger and richer dataset perhaps not limited to controversial questions. In order to curate these datasets I actually signed up for a hackathon that provided a small amount of GPT credits. With some more funding I, believe I can see even better results.

References

[1] **Reimers, Nils, and Iryna Gurevych**, "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084 (2019).

[2] **K. Taunk, S. De, S. Verma and A. Swetapadma**, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

[3] **Guo, Zishan, et al.**, "Evaluating large language models: A comprehensive survey." arXiv preprint arXiv:2310.19736 (2023).

[4] **Increase AI performance and reduce cost with the LLM router. Martian. (n.d.-b). https://withmartian.com/**