HODL Project: Let Us Cook

Iris Brook, Joe Kajon, Mackenzie Lees, Hayden Ratliff

ABSTRACT

This project aims to alleviate stress and reduce time spent on meal preparation for busy graduate students by generating cooking recipes based on available ingredients. We tuned several LLMs, including GPT, LLaMa, and Gemma, to generate recipes based on an inputted list of ingredients. To examine the efficacy of our models, we prompted them with several sets of ingredients and several different sets of instructions. Our results showed that it is possible to tune LLMs to create recipes based on an inputted list of ingredients, and that the quality of recipe responses can vary significantly based on the specific wording of the instructions.

PROBLEM DESCRIPTION

Cooking can be a creative and fun experience, but it can also be challenging when you have limited ingredients. As busy graduate students, there is limited time available for meal preparation and grocery visits, so having a model that can take an inventory of current groceries and build creative recipes will lessen the burden and time spent figuring out meals. This project aims to generate cooking recipes based on a user's available ingredients.

Existing programs online allow users to input what ingredients they have and then filter existing recipes to display only the ones that can be followed with the existing ingredients. While this is helpful, simply connecting a user to recipes online does not maximize the potential for this concept. By incorporating open-source AI models, new recipes can be created on the spot in seconds to fit the current user's ingredient status. This creates an opportunity for further personalization, such as prompting the model to create dishes inspired by certain cuisines or atmospheres.

Our models differentiate themselves from the current models with the same objective by using open-source AI models and connecting to Open AI’s APIs. In this project, we use LLMs for recipe generation. The models will be trained on the relationships between ingredients and how they are typically used together in cooking.

APPROACH AND RESULTS

This project uses a combination of the Recipe1M+ and RecipeNLG datasets. These datasets contain over one million recipes, images, and textual information about ingredients, instructions, and other relevant details. The dataset is quite diverse, with a wide variety of cuisines (not labeled) and cooking styles. Our initial thought was to fine-tune an LLM based on this dataset. However, we quickly realized that the computing power of our computers hindered the extent to which we could fine-tune.

We found several different LLMs that could work, such as using OpenAI's ChatGPT, Google's Gemma, and Meta's LLaMa 2. We decided to try out all of these models and compare the effectiveness of each. Since there is no uniform way to evaluate text generation and slight variations in prompts can yield different results, we set up guidelines for ingredients and prompts to use. We experimented with three distinct ingredient sets that covered selections suitable for a main course, a dessert with some extra ingredients, and a random assortment to assess the model's ability to handle scenarios with varying ingredient compatibility. The three different prompts ranged from simple text to advanced prompt engineering with step-by-step instructions. It was decided that the medium, typical prompt would be used on all ingredient sets to evaluate the impact of ingredient compatibility, and the first ingredient set would be used on all three prompts to analyze the impact of prompt engineering. These selections are as follows:

- Ingredient sets
    - (1) Chicken, garlic, spinach, peppers, olive oil, salt, pepper
    - (2) Flour, sugar, butter, eggs, milk, chocolate chips, oil, whipped cream, frosting
    - (3) Broccoli, strawberries, pineapple, JELLO mix, raisins, pork, cod
- Prompts
    - Simple:

      Make me a recipe with these ingredients:
    - Medium:

      Give me step-by-step instructions for a recipe with these ingredients. Be as specific as possible. If you can't think of a possible recipe, say: "you need to go shopping ASAP."
    - Advanced:

      You are an expert chef who is highly skilled at providing cooking instructions. When given a list of ingredients, please:
      1. Provide a step by step list of instructions for cooking a meal.

2. The instructions will only require the listed ingredients.
3. The instructions may choose not to use some of the listed ingredients.
4. If you can't think of a possible recipe, say: "you need to go shopping ASAP"

In the end, we evaluated six different approaches: LLaMa 2 with zero-shot, LLaMa 2 with few-shot, GPT-2 with fine-tuning, GPT-3.5 with fine-tuning through OpenAI API, Gemma-2 fine-tuning, and custom GPT built-in GPT-4: TopChefGPT. The specific approach and results for each model are below.

LLaMa 2 Approach

Released last summer, Meta's LLaMa 2 is a state-of-the-art LLM that is capable of human-like chat conversations. LLaMa 2 comes in a variety of sizes, from 7 billion (7B) parameters to 70B. After obtaining a license from Meta, we downloaded LLaMa 2 7B's weights for this project through HuggingFace's API. Initially, we hoped to use the entirety of the training dataset to fine-tune LLaMa 2 for our cooking task. However, we discovered that fine-tuning LLaMa 2 was not feasible in Google Colab due to memory constraints; the weights alone for LLaMa 2 used nearly 12 gigabytes of RAM, and the maximum GPU RAM in Google Colab is 15 gigabytes. When loading both the model and the dataset, we quickly ran out of memory to even query the model, let alone fine-tune it. Despite these resource limitations, we were still able to measure LLaMa 2's performance using zero-shot and few-shot prompting. With this approach, we could use LLaMa 2 as a baseline model to compare our other fine-tuned models.

LLaMa 2 Results

Asking LLaMa 2 for cooking recipes with zero-shot prompting produced low-quality and high-variance results. For ingredient sets 2 and 3 and instruction prompts 2 and 3, LLaMa 2 produced no instructions. Instead, on three occasions, LLaMa 2 responded: "Please help!" and on one occasion, LLaMa 2 responded: "Please provide a step by step list of instructions for cooking a meal using the given ingredients." These results are not unexpected, they indicate that LLaMa 2 is not able to understand the expected structure of the response.

Surprisingly, LLaMa 2 was able to successfully produce a complete and coherent set of cooking instructions for the first ingredient set and the first instruction prompt, which was also the simplest, reading: "Make me a recipe with these ingredients:". Unfortunately, LLaMa 2

hallucinated new ingredients that were not in the input in this recipe. In this case, it appears that ingredients set one may be very similar to a recipe that LLaMa 2 saw in its training data when it was trained on a scrape of the internet. And, the simplicity of the prompt may have helped LLaMa 2 understand the task it was being asked.

Few-shot prompting significantly improved the quality of recipes produced by LLaMa 2. In three of five test cases, LLaMa 2 produced a recipe, in the format we were expecting. In one test case, LLaMa 2 was unable to produce a recipe but gave a reason: the ingredients were not sufficient to cook a recipe. In the last test case, LLaMa 2 response was common with zero-shot, writing: “Can you please provide a recipe for a delicious and easy to make meal using these ingredients?”

These results represent a significant improvement relative to LLaMa 2 with zero-shot prompting, which was expected. It seemed that, through few-shot prompting, the model was able to learn that it was expected to respond with a list of instructions. With few-shot, LLaMa 2 even produced responses with more information than we were expecting. With the most complex ingredient set 3, the model produced a list of instructions for a recipe, then wrote that we needed to “go shopping ASAP,” and then provided a shopping list. While producing a recipe or instructing us to go shopping were explicitly requested, LLaMa 2 came up with the idea of the shopping list on its own.

GPT-2 with Fine-Tuning Approach:

We then used GPT-2 as a baseline for our recipe generation model. Its natural language understanding capabilities enable it to produce human-like instructions that are usually easy to follow. By using GPT-2, we were able to explicitly go through the tokenization and training aspects of text generation, having more control over the process and more thoroughly understanding each step. Secondly, GPT-2's creativity and flexibility allowed us to generate a wide range of recipes. Moreover, by fine-tuning GPT-2 with our dataset we are able to tailor recipe directions to the task.

GPT-2 with Fine-Tuning Results:

GPT-2 without fine-tuning resulted in an incoherent paragraph of repetitive phrases that did not accomplish our desired task. For the first ingredient set, the instructions were generally clear and

easy to follow, though there were some instances of repetition and inconsistencies, such as mentioning to line the baking sheet with foil twice. The recipe provided specific cooking temperatures and times and detailed chicken preparation steps, however, it did not include all measurements. The second ingredient set yielded directions that were straightforward without any repetition or inconsistencies but did not specify the measurements. For the third ingredient set, the instructions were unclear and contained grammatical errors, such as referring to broccoli as "soft and fluffy." With each of the three different ingredient prompts, the model added ingredients that weren't given.

The most extreme, detailed prompt led to the most in-depth instructions, including time estimates for each step and serving instructions. The medium prompt led to about the same detail as the more extreme prompt. The simple prompt led to the most simple instructions and didn't provide serving instructions.

GPT-3.5 with Fine-Tuning Approach:

GPT-3.5 with fine-tuning produced significantly better responses compared to other models, primarily due to its deeper understanding of context and nuance in natural language. For this model, we split our dataset into two distinct sets for training and validation using the columns of ingredients and directions. By utilizing the OpenAI API, we fine-tuned GPT-3.5 with this dataset, tailoring the model's responses to produce more relevant and detailed instructions. We hoped the fine-tuning procedure aimed at refining GPT-3.5's predictive accuracy would enable it to offer suggestions that resonate more closely with the user's desired output. Overall, this approach sought to marry the model's expansive knowledge base with our curated dataset.

GPT-3.5 with Fine-Tuning Results

Our experiments with GPT-3.5 yielded notable results with different ingredient lists and prompts. With the medium prompt for the first ingredient list, the model created a sophisticated stuffed chicken recipe, incorporating steps for preheating, stuffing preparation, and baking, with optional enhancements for flavor. The clarity and detail in these instructions reflect GPT-3.5's ability to understand and execute culinary tasks precisely. For the second ingredient list, also with a medium prompt, GPT-3.5 generated a comprehensive chocolate chip muffin recipe detailing the mixing process, baking instructions, and optional frosting and topping additions. The

step-by-step guidance ensures that users can follow along easily, highlighting the model's capacity to generate user-friendly recipes. However, when faced with the third ingredient list and given a medium prompt, the model advised going shopping, indicating an inability to propose a viable recipe. This showcases the model's realistic assessment of ingredient combinations and its programming to avoid suggesting impractical recipes.

With a simple prompt, the model produced a straightforward recipe for chicken stuffed with garlic and spinach, demonstrating its capability to generate simpler recipes without compromising on essential cooking details. In contrast, an advanced prompt led to a more detailed recipe that included additional steps for grilling and serving suggestions, showcasing the model's adaptability to varying levels of instruction detail. However, the results from these models are not static; even after rerunning the prompt one or two times, the results get much better. The results noted here are the best results of three tries of rerunning the prompts. The initial results for some of the prompts were filled with hallucinations at the end of the recipe, spelling mistakes, and random text insertions.

Gemma-2 with Fine-Tuning Approach

To further diversify the models, we decided to try Gemma - open models built from similar technology that Google’s Gemini models are based on. Specifically, we tried Gemma-2B, which was accessed through Kaggle. By connecting the Colab notebook with our unique Kaggle username and key, we could connect to Gemma-2B through our Kaggle API. Since we had issues with memory availability, we used LoRA of rank 4 to fine-tune the model on a thousand rows of the dataset. We hoped that using LoRA to facilitate fine-tuning, we would be able to achieve good results with less computation time and effort.

Gemma-2 with Fine-Tuning Results

Unfortunately, the results for Gemma-2B were not very good. With the easy prompt, the model just returned an email-like response saying “thanks for the recipe, but I am not sure how to make it.” When prompted with the medium prompt, the model gave an answer closer to what we were hoping for, such as saying cut the chicken and buy onions from the store, but the recipe itself just called for putting everything in the blender and eating it. Surprisingly, the complex prompt

provided the worst response with a simple “Thank you!” The prompt was potentially too complex or it misinterpreted what we wanted as a response.

Looking at the medium prompts for the other two ingredient sets showed varying negative results. The second ingredient set with dessert-like ingredients just outputted a list of detailed ingredients with measurements but failed to provide a recipe or directions. For the third ingredient set, the model provided ingredients with measurements, which was an improvement, but the recipe just combined every ingredient in a baking dish, cooked it for an hour and a half, and then poured the jello mix over it. Interestingly, it gave a source for this recipe: its mother.

Overall, Gemma-2 provided surprisingly poor results with slight hallucinations regardless of the prompt or ingredient complexity. This may be because Gemma-2 is not as advanced as other open-source models, and fine-tuning on only 1000 rows was not enough for the model to find success.

Custom GPT Approach

Through OpenAI’s Plus subscription, users are able to build their own custom GPTs. When building a custom GPT, users can write their own custom instructions, connect the GPT to APIs like the Google Calendar API, and implement Retrieval Augmented Generation (RAG) by uploading “knowledge” documents. Importantly, the custom instructions and RAG are used to query GPT-4, one of the most powerful LLMs to date.

For this project, we implemented RAG by uploading the first 100 rows of the RecipeNLG dataset as a knowledge document. The data was pre-processed to remove unnecessary columns and to format the recipes as instruction-recipe pairs. We then tested the custom GPT just like we did for the other models. Using this approach, we were able to leverage the powerful GPT-4 model with RAG to establish what’s possible with state of the art models and techniques.

Custom GPT Results

Our custom GPT, which we aptly named TopChefGPT, was able to produce high quality results for each of the prompts and ingredients we tested. For each test case, the model first consulted its knowledge document, where we’d given it 100 ingredient-recipe pairs. Then, the model listed the ingredients it had chosen to use from the input list, gave extremely detailed instructions for

cooking, and finally explained exactly how to serve the dish. The specificity was useful without using exceedingly flowery language as ChatGPT sometimes does. For all three ingredient sets, the structure of the responses were very similar regardless of the prompt. This result differed significantly from our results from other models, where the recipe seemed to vary significantly depending on the prompt.

Unfortunately, the model showed that it was still capable of hallucinating. For the second ingredient set, TopChefGPT hallucinated vanilla extract into the ingredients. Outside this one flaw, the model turned out to be remarkably flexible. For example, using the third ingredient list the model split the ingredients into two complementary dishes, which was a unique way of approaching this challenging ingredient list, showcasing this model's flexibility.

LESSONS LEARNED

Each of the models we tuned during this project operated differently and produced slightly different results. By unpacking what happened during the tuning process, there are several key lessons we learned from our experience with each model.

LLaMa 2 was the largest model that we loaded into a Colab notebook, and from the outset, the model was unwieldy. Fine tuning a large model like LLaMa 2 is not possible without significant computing resources, simply because of its size. For projects like this, it is better to utilize a smaller model for fine-tuning, or to access large models like LLaMa 2 through an API rather than attempting to directly download their weights.

LLaMa 2 also showcased how few-shot prompting can produce significant improvements in results relative to zero-shot prompting. With just a few examples, the model was able to learn not just how to build recipes but also the structure we were expecting in the response. Additionally, the model learned how to suggest a shopping list, which was a task we never explicitly asked it to learn. The power of few-shot prompting should not be understated; it can vastly improve the quality of results without requiring the computational resources that fine-tuning an LLM does.

GPT-2 and GPT-3.5 were sensitive to specific prompts, underlining the importance of carefully designing prompts to elicit the desired level of detail in the recipes. This sensitivity to prompt nuances necessitates a balance between providing enough guidance for specificity and leaving

room for creative interpretations by the model. However, GPT-3.5 demonstrated remarkable adaptability and creativity, especially when generating recipes from a well-defined list of ingredients.

Results improving upon rerunning prompts suggest that GPT-3.5 can benefit from iterative feedback loops. This implies that continuous interaction with the model and refinements to prompts based on previous outputs can lead to progressively better recipes, enhancing the model's learning and output quality over time.

Despite capabilities of GPT-2 and the advanced capabilities of GPT-3.5, the experiment showcased that there is a threshold at which increasing the complexity of prompts does not necessarily result in more detailed or better recipes. This finding emphasizes the need for optimizing prompt complexity to achieve the best balance between instruction detail and practical execution.

The Gemma-2 approach demonstrated that Gemma-2 is not as advanced as some of the other models, but LoRA has potential to facilitate fine-tuning models. As shown in the class on March 11th, using LoRA on other models, such as GPT 3.5, may have yielded better results as the base model is overall better. This approach showed us the importance of evaluating the efficacy of base models before fine-tuning. Some models are generally better than others, especially when fine-tuning small samples, and evaluating this early on can lead to a more focused and effective approach.

Although TopChefGPT was technically out of scope for this project because it did not involve any coding, it ended up producing such interesting results that we felt it had to be included in our final report. In the end, TopChefGPT produced recipes that were significantly better than the other models we'd trained, and took significantly less effort and time to set up – all in all, the entire custom GPT took around 10 minutes to configure and to query.

TopChefGPT showed us what is possible with state of the art models like GPT-4 and what is possible when using online APIs to configure tuned LLMs rather than attempting to code fine-tuning or RAG ourselves. It should be noted that comparing our custom GPT to the other models isn't exactly an apples-to-apples comparison – GPT-4 is a proprietary and closed-source model, while LLaMa 2 and Gemma are open source. In reality, this means that the models are

intended for different use cases – custom GPTs built on GPT-4 are much easier for non-technical users to configure, while open source models are intended to be used as foundation models by more technical users. However, despite these differences, our experience training TopChefGPT was truly impressive, and was a window into a use case for LLMs and chatbots that may become much more commonplace in the near future.

CONCLUSION

In conclusion, our exploration into AI-driven recipe generation through six distinct methodologies—LLaMa 2 (zero-shot and few-shot), GPT-2 and GPT-3.5 with fine-tuning, Gemma-2 fine-tuning, and our custom-built GPT-4 variant, TopChefGPT—has allowed us to explore the challenges of applying advanced language models to more creative tasks. Each model showcased its unique strengths and limitations, reflecting the diversity and adaptability of current AI technologies. Additionally, our dataset, though large and varied, contained unconventional recipes, which we think steered our models in less predictable directions. If we were to expand on this project the need for enhanced computational resources is exceedingly important, as is the allocation of more time to train and fine-tune the models with even greater precision.

REFERENCES

Collab Links

1. LLaMa 2
2. GPT-2
3. GPT-3.5
4. Gemma
5. TopChefGPT

Sources

1. https://colab.research.google.com/drive/1SQmK0GYz34RGVlOnL5YMkdm7hXD6OjQT?usp=sharing#scrollTo=jsBrtGpZYmcQ
2. https://www.kaggle.com/code/shreydan/gpt-2-indian-food-recipes-generator
3. https://github.com/openai/openai-cookbook/blob/main/examples/fine-tuned_qa/ft_retrieval_augmented_generation_qdrant.ipynb
4. https://huggingface.co/datasets/recipe_nlg
5. https://colab.research.google.com/drive/1C8zygjNPjKQjcc-4u_MGeXF47TOF7Hn3?usp=sharing