Andrea Dunn Beltran, Noah Weaver, Calvin Lee
Comp 560: Artificial Intelligence
March 8, 2024

Project Proposal: Culinary Insights

Problem Introduction

Imagine it's getting towards the end of the week. Dinner time rolls around and you look in the cupboard where a spattering of odd ingredients wait. No matter how hard you stare at the curious shmorgishborg no recipes come to mind. It's this problem that we aim to fix: given an input of ingredients, provide a recipe while minimizing the number of ingredients you'd have to buy.

We hope to approach the problem in two vastly different ways in order to compare and contrast methodologies. First we will use an A* search algorithm to sort through a database of recipes according to a function of our own devising. Then we will train a language learning model to generate responses for this specific food related task.

As we work, we may add advancements to the search functionality. Accounting for nutritional factors and taking in cuisine preferences are two improvements to the base design that we've considered.

There are various similar services available over the internet. Dishgen is an existing application of language learning models for a similar purpose. It takes in prompts flexibly and creates new recipes that aren't necessarily preexisting. Another notable service is MealPractice, where users select one of multiple preset attributes ranging from protein options to dietary limitations. It then gives three options for users to choose from and generates a recipe.

Existing Resources

We will use the provided Food.com dataset which includes 180K+ recipes and 700K+ recipe reviews covering 18 years of user interactions. Given our recourse and experience levels we plan on fine tuning an existing GPT model for this specific task. We will also look further into the Keras library as a relatively user friendly tool for neural network creation, utilizing python to act as an interface for the TensorFlow library.

Envisioned Approach

The true story of our project lies in the comparison and integration of two methodologies. The **search algorithm** which offers a direct pathway to existing recipes, optimizing for practicality and immediacy. In contrast, the **generative language model** generates novel recipes

Part 1: Search Algorithm Development

To efficiently match users' available ingredients with recipes, we propose utilizing an A* search algorithm, which is suited for searching through large, complex spaces to find an optimal sequence of actions (in this case, ingredient combinations)

- **Search Algorithm for Recipe Matching**: A* search can be applied by defining the state as the current combination of ingredients available. The goal is to reach a state where a complete recipe can be produced with the available ingredients. The cost function to minimize could combine factors such as the number of missing ingredients and possibly the cost or difficulty of acquiring those ingredients. Heuristics could be based on the similarity of the available ingredients to those required by recipes, guiding the search towards recipes with ingredients closest to the available set.

Part 2: Language Model Training

For generating new recipes based on a set of given ingredients, we aim to utilize advanced language modeling techniques. Our approach will include:

- **Model Selection:** GPT-2: Given the task's nature, transformer-based models such as GPT (Generative Pre-trained Transformer) or BERT (Bidirectional Encoder Representations from Transformers) offer significant advantages due to their ability to understand context and generate coherent text. Since we would like to use a free and publicly available model, GPT-2 is a natural option. While it may not be as large as its successor, GPT-3, GPT-2's architecture is sufficiently capable of generating contextually relevant text, making it sufficient for recipe generation. Its open-source availability allows for unrestricted experimentation and fine-tuning.
- **Data Preparation and Preprocessing:** Similar to the previous approach, the dataset will be preprocessed to align with the input-output format expected by GPT-2. This involves creating a clear delineation between the list of ingredients (input) and the recipe instructions (output).
- **Fine-Tuning**: Leveraging transfer learning, GPT-2 will be fine-tuned on recipe dataset. The fine-tuning process will adapt GPT-2 to better understand culinary terminology and recipe formats, enhancing its ability to generate relevant and potentially novel recipes from input ingredients.

Evaluation Plan

Our model's success in generating realistic recipes with easy-to-understand instructions will depend heavily on our evaluation plan. Setting aside a part of our dataset as a validation set, we will then fine-tune our model using some commonly used technical metrics such as ROUGE and BLEU score to focus on the overall text quality. ROUGE and BLEU will be used to compare the fraction of shared n-grams between the generated and validation recipes.

Another technical metric, perplexity, will be used to evaluate how well our model understands the structure and content typical to recipes by measuring its uncertainty in predicting the next words in a sequence.

Lastly, we know we will want to use a custom relevance metric to evaluate how well our model reduced the number of ingredients bought and increased the number of ingredients used that we already own. Preliminarily we have 2 options:

A score that weighs buying an ingredient and not using an ingredient as equally detrimental:

$$Score \ = \frac{\#ingredients\ used - \#ingredients\ bought}{\#ingredients\ owned}$$

A score that weighs buying an ingredient as more detrimental (from -1.0 to +1.0):

$$Score \ = \left(\frac{1}{3}\frac{\#ingredients\ used}{\#ingredients\ owned} - \frac{2}{3}\frac{\#ingredients\ bought}{\#ingredients\ owned}\right) \times 3$$

These scoring evaluations are subject to change as we develop our model but will be useful in determining how well the model solves our problem, as well as for comparing the search results and the model results. In the case of our search algorithm, we will want to measure its time and space efficiencies.