Applied Deep Learning Project
Personalized Meal Recommendation

Isabella Illes - 11771128

January 17, 2025

Contents

1 Introduction

In this project, the problem of finding meals that not only meet the dietary requirements but also satisfy the personal needs is addressed. Planning and finding recipes that not only suit your tastes but are also beneficial for your body and overall health takes time and effort. Ideally, recipes should also be fine-tuned to specific lifestyle goals, such as increasing protein intake or reducing saturated fat and sugar consumption.

To help facilitate the process of preparing recipes, I have implemented a **Personalized Meal Recommendation** system. The system is designed to recommend weekly meal plans tailored to user preferences, dietary restrictions, and nutritional goals.

For this system, the topic and approach of **Deep Reinforcement Learning** (DRL) has been chosen.

The idea of using DRL for meal planning was inspired by two studies:

- *Delighting Palates with AI*[Ami24]
- *Recipe Renaissance*[Mah24]

Both articles specifically discuss personalized meal recommendation. While there are traditional systems that do not use Deep Learning methods and adhere to nutritional and health needs, they struggle with the task of aligning meal plans to a user's specific dietary preferences and patterns. To address this challenge, both papers introduce approaches using Deep Reinforcement Learning methods to better accommodate and represent the complex relationships between food and user preferences.

[Ami24] [Mah24]

For the **Deep Reinforcement Learning** approach, the system uses **Double Deep Q-Learning** (DDQN). This project is categorized as **Bring Your Own Method**, as it takes advantage of existing datasets and adapts implementations to meet the unique requirements of personalized meal recommendations.

2 Dataset

In this section, I will briefly describe the dataset used for the recommendation system. During development, the recipe dataset underwent some changes. The final dataset is:

- Food.com - Recipes and Reviews

The dataset is an extensive collection of recipes with data crawled from **Food.com**. It contains several files; however, the file of interest for this application is **recipes.csv**. The other files are not included.

The file contains over 500K recipes and is approximately 700 MB in size. There are 28 columns in total, featuring both categorical values and some missing data. Each recipe includes a list of ingredients in text format as well as additional metadata about the recipe. The dataset also includes search queries that return the recipe and user-assigned tags. These tags provide additional dietary information, such as vegetarian, high protein, low carb, etc.

3 Algorithm and Approach

The initial approach for this project was to use Deep Reinforcement Learning with a **Policy Gradient Algorithm**, specifically the **Proximal Policy Optimization (PPO)** algorithm. However, it was replaced with **Double Deep Q-Learning (DDQN)** due to issues arising:

- Implementing PPO in the custom meal environment proved overly complex and infeasible. Adjustments for specific constraints and rewards were difficult to integrate.

The DDQN algorithm was implemented by creating an agent class that incorporates the main Q-network and a separate target network which is periodically updated with the weights of the main Q-network. The algorithm optimizes a loss function based on the **mean squared error** (MSE) between predicted Q-values and target Q-values.

Additionally, the agent leverages an **epsilon-greedy policy** for exploration and exploitation. The epsilon value is decaying over time to balance exploitation and exploration.

3.1 Network Design

A network class was implemented to support the DDQN with two available architectures to compare. Each architecture comprises an input layer, hidden layers, and an output layer, with sizes determined by the action and observation spaces.

The input layer encodes meal features, with its size matching the dimensions of the observation space.

The hidden layers: The first architecture employs fully connected layers using ReLU activation functions. **The output layer** computes Q-values for the available actions with its size corresponding to the action space.

In contrast, **the hidden layer** in the second architecture incorporates an LSTM layer. This enables the model to capture temporal dependencies and relationships across environment steps. The output layer acts the same in both architectures.

The goal behind the two architectures is to compare the performance of different architectures on the result. The architecture utilizing LSTM layer shows on average significant lower MSE loss. However, in terms of average return there does not seem to be a stark difference between those two architectures.

3.2 Environment

Since deep reinforcement learning was employed, a custom environment had to be designed for meal planning. The discrete action space represents the indices of all available meals, where an action corresponds to selecting a meal from the processed dataset based on this index. The observation space encompasses the features associated with each meal, such as nutritional content and other attributes.

A key constraint imposed on the environment is that meals cannot be repeated within the same week. Initially, the same action was repeatedly selected for the entire week. Introducing negative rewards for repeated meals proved ineffective in addressing this issue. Instead, an **invalid action mask** was implemented. This mask starts with all values set to 1, representing valid actions. Once an action is selected, its corresponding index in the mask is set to 0, effectively preventing its selection in subsequent steps. When actions are chosen, their probabilities are multiplied by the mask, ensuring that previously selected meals have a probability of 0 and cannot be chosen again.

3.3 Reward Function

The reward function plays a central role in guiding the agent to generate effective meal plans. It is defined in the custom environment, and it is designed to encourage the selection of meals that meet the nutritional target values. Furthermore, meals whose tags overlap with user-preferred keywords are also rewarded. During the development, encouraging behavior seems to be more effective in resulting in desired target intervals than punishing behavior.

Both episode and terminal rewards are calculated. The state is said to terminate after a whole week has been filled with selected meals. Then, the daily averages of the nutritional content of the week are computed. The agent is rewarded if the specific content values (protein, fiber and saturated fat) meet the target thresholds. Episode rewards are calculated by extracting the nutrient values of the current day and comparing them against the target thresholds. Additionally, rewards are added for each keyword of a selected meal matching keywords of the user preferred meals.

3.4 Evaluation Metrics

The evaluation metrics are primarily defined later in the post-processing stage. The model's success is primarily measured by how well the meals selected over the week meet specific nutritional requirements, focusing on protein, fiber, and saturated fat content. The evaluation process is simplified by concentrating on these three values.

The model's success is measured using:

- **Success Rate:** It measures the fraction of episodes where the average content of protein, fiber, and saturated fat across selected meals meets the user's dietary guidelines within a specified threshold. Ideally, the agent should achieve an 80% success rate for each nutrient, meaning at least 80% of episodes should see the average content fall within the required range. Protein and saturated fat content generally perform well, but fiber underperforms. This can be attributed to the average fiber content in the dataset being significantly lower than required for the dietary goal.

Further metrics that are used to help evaluate the model during experiments are the following:

- **Average Return:** This metric measures the overall performance of the agent by assessing the rewards accumulated across episodes. While helpful in evaluating the stability of the learning process, the average return does not play a significant role in analyzing the error metric. The agent is expected to converge to a consistent and high average return, but occasional spikes in reward indicate some instability in the network's learning.

- **MSE Loss:** Tracks the loss function of the DDQN agent during training.

4 Implementation

A complete pipeline, including tests, is created for the project. The pipeline first runs tests on individual modules, such as preprocessing, training, networks, and postprocessing. After the tests conclude, the pipeline preprocesses the dataset, which involves removing rows with missing values, hot-n encoding categorical variables, filtering out time intensive recipes, and standardizing numerical values.

Next, the training process begins, requiring initialization of the custom environment and DDQN agent. After training, postprocessing starts, where the agent and results are evaluated based on defined metrics, and the weekly meal plan is generated. The following plot shows the average nutrient content per meal over the episodes during training. Ideally, the values should converge to the target values, however, one can notice spikes and in general oscillating values, implying that the agent struggles with stability even with a separate target network.

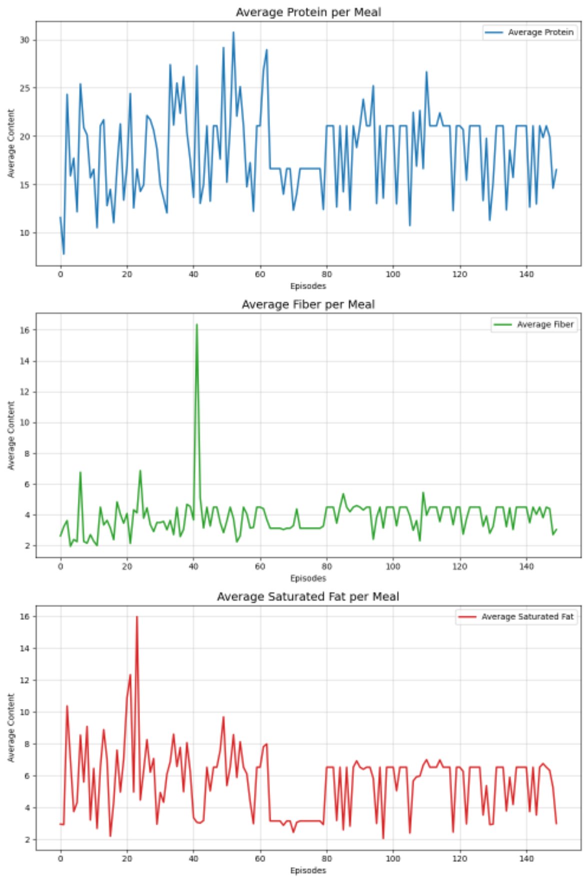

Figure 1: Average content per meal

Figure 2 shows the output of a recommended meal plan in regards to protein, fiber and saturated fat content for each day.

In addition to the pipeline, a web application has been developed. This application recommends recipes for at least a week based on stored personal preferences, fetching necessary data and the trained agent from a GitHub release if no local data is available.
The web-application provides options for users to declare favorite recipes and adjust target nutrient content. Using this information, the model can be re-trained and fine-tuned to suit individual preferences.

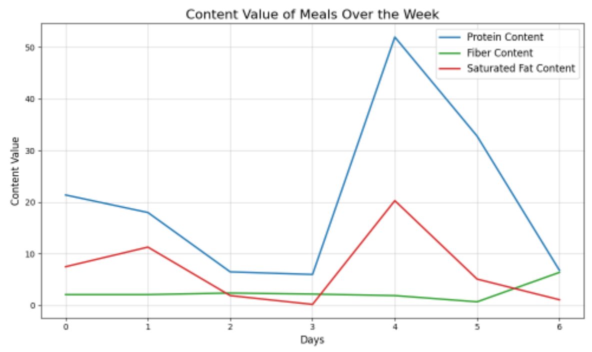

Figure 2: Content values of a recommended meal plan

5 Work-Breakdown

In conclusion the overall breakdown and time spent on individual tasks in comparison to the initial time estimates were roughly similar:

- Dataset collection: 1 hour (estimate: 1 hour)
- Dataset preparation, processing and transformation: 10 hours (estimate: 10 hours)
- Designing and building an appropriate network: 22 hours (estimate: 28 hours)
- Training and fine-tuning that network: 14 hours (estimate: 20 hours)
- Building an application to present the results: 15 hours (estimate: 20 hours)
- Writing the final report: 10 hours (estimate: 10 hours)
- Preparing the presentation of my work: 7 hours (estimate: 4 hours)

6 Challenges and Conclusion

During the development of The Personalized Meal Recommendation System several challenges were encountered, which influenced the overall design and outcomes.

6.1 Dataset Limitations

The initial dataset lacked essential nutritional information, prompting a switch to a more suitable dataset. Additionally, the plan to incorporate budget constraints was abandoned due to several issues:

- **Inconsistent Pricing:** Variability in product prices, including multiple price options for the same item created difficulties.
- **Unit Mismatch:** The recipe dataset did not include ingredient quantities in standardized units (e.g., kilograms), while the price dataset did. Resolving discrepancies between ”2 apples in a recipe” and ”1 kg of apples costs X” would introduce impreciseness, making the results unreliable.

When working with the final dataset further shortcomings appeared. The tags and keywords were not always accurate or meaningful, making them unreliable for dietary constraints like vegetarian or diabetic diets.

Looking back, it might be more suitable to create different datasets for dietary constrains (vegetarian dataset, diabetic dataset, etc.) and to only refer to the filtered one rather than letting the agent learn how to select for such strict requirements.

6.2 Reward Function

In the early stages, the agent would tend to select the same action throughout the entire week. Negative rewards were ineffective, and the solution that worked best was invalid action masking. Additionally, one of the major hurdles was consistently meeting nutritional targets, particularly when trying to balance high protein and high fiber content. It was extremely difficult finding reward functions and fine-tuning them to meet the required performance.

Designing and tuning the reward function was one of the most time-consuming aspects of the project, and it was challenging to identify effective measures.

6.3 Network and Hyperparameter Tuning

Experimenting with different network architectures and hyperparameters resulted in marginal improvements. However, these adjustments did not lead to significant breakthroughs. The differences between various network architectures were less impactful than initially expected.

6.4 Future Considerations

Looking ahead, generative models could offer adaptive recipe recommendations by generating novel combinations based on user preferences and constraints. This would provide greater flexibility in combining individual foods.

References

[Ami24] F.; Li J. Amiri, M.; Sarani Rad. Delighting palates with ai: Reinforcement learning's triumph in crafting personalized meal plans with high user acceptance. *Nutrients*, 2024.

[Mah24] Neha Tyagi; Vijay Maheshwari. Recipe renaissance: Leveraging deep reinforcement learning for food recommendations systems in culinary innovation. 2024.