Recipe Recommender Milestone 3

UBC-CPEN291/project-team-apatosaurus
Harrison Mitgang (hmitgang), Bowei Ren (boweiren), Alina Shabanova (alinas2000), Jane Wu (jwu611)

Current State

We split up into teams of two, where one team would work on the classifier and the other would work on the recommender. Alina and Jane worked on further improving the classifier and Harrison and Bowei worked on the recommender. Both groups also began investigating and testing how to integrate the classifier and recommender with a website.

Classification Model

The classification model is complete and outputs an accurate result when an image of a food item is uploaded. This week the model settings were further optimized to obtain the values for optimum results. To figure out the best settings, we used the settings from milestone 2 that gave the best results and tried combining them with different learning rates, weight decay, step size and gamma values. However, the changes did not result in any improvements. In conclusion, the new tests run in milestone 3 and the overall best accuracy results are summarized in the ingredient_classification notebook. The model was also trained using a single step scheduler, but did not improve the results.

Classification Website Integration

The trained classification model has been integrated into a test website built with Flask and HTML. The flask-ngrok package was used to quickly and easily demo the website from colab. Currently, the website supports a single image upload and is able to output the predicted label for the ingredient in the image using the trained ingredient classifier from milestone 2.

User Input

In another web app, six recipes are initially picked from the 20 recipes with the most ratings. Users will be asked to rate them. The results are sent to the model as user input for training. This website can be easily combined with the classification website to form the final website.

Recommendation Website

Using Flask and SocketIO, the user is able to rate a few random recipes (this UI will be replaced by the part described in “User Input”), wait for the model to train with a progress indicator, and view the top recommended recipes. Clicking on a recipe will bring you to a more detailed description on the recipe.

Feature Changes

The only feature change applied to the proposal during Milestone 3 is to utilize SocketIO. Since we must retrain the recommender with each user, SocketIO allows the user to track the progress of the training, and receive the results without waiting a few minutes for the page to load. SocketIO allows for seamless communication between the browser and the backend.

Challenges

- When integrating the classifier with the website, we had to investigate how to make predictions with the model without having to retrain each time. This was resolved by pickling the model
- When integrating the classification model with the website prototype, there were some issues with file directories. This was resolved by making sure the input image was saved in the appropriate folder and that the right directory was being accessed as the path to the upload_image function.
- When designing the questions, the original plan was to first filter the recipes according to the ingredients, and then dynamically generate six recipes. However, after testing several inputs, it is found that the recipe generated in this way has greater uncertainty, and it is also prone to the situation where there are fewer than six screening results. So temporarily use six fixed recipes instead, and then decide which plan to use based on the test situation.
- Retraining the recommender for each user takes a long time. In order to provide the user with details on the progress of the training, we will use SocketIO so that the training can happen in the background while the user waits on a waiting page.

Member Tasks

- Both Alina and Jane worked on optimizing the classification model, trying out different values and schedulers. All the code and debugging was done using pair programming. Alina and Jane also worked on integrating the classification system with the test website using Flask. Jane was more involved with writing the code whilst Alina did research on pickling, debugging was done using pair programming.
- Bowei built and debugged a draft recommender website using Flask. It takes a user's response and will be combined with the classification website to form the final website.
- Harrison built a light front end that will be replaced with Bowei's part, as well as the backend to process recommendations, and provide detailed recipes for each recipe.