Recipe Recommender Milestone 2

[UBC-CPEN291/project-team-apatosaurus]
Harrison Mitgang (hmitgang), Benjamin 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 the classifier and Harrison and Bowei worked on the recommender.

Classification Model

The classification model has been trained successfully and is complete, producing results with high accuracy. It involves a function (`identify_sample`) that takes in the path of the input image locations and outputs a tensor of the matching ingredient enumerated. This function will be used in later stages of the project for the recommender and the website.

To figure out which optimizer would be best to use for training, the training model was run using SGD, AdamW and Adam, and the best accuracy results were produced when using AdamW with `lr` = 0.0001 and `weight_decay` = 0.02. We experimented with the grayscale, horizontal flip, and ColorJitter transforms but, no major improvements to accuracy were produced.

For the next milestone, the model will be further improved by experimenting with other parameters such as learning rates, momentum etc and we will also begin integrating it with the website.

Recommender Model

We implemented two models, one to the spec of the original proposal using Surprise, and one using the techniques we learned in class using PyTorch. As will be outlined in the challenges section, we are not sure if either of these models will work the best for our project.

With that said, the recommender model using PyTorch has been trained and optimized successfully, producing a training loss of 0.27 and a test loss of 1.40.

In the stage of optimization, we tried multiple combinations of different optimizers with different parameters. In the beginning, we tested Adamax, ASGD, AdamW, and SGD. And we found out that SGD has the best efficiency and lowest loss. Then we adjusted the parameter in both optimizer and scheduler. Finally, we reached the lowest loss by setting `lr` to 1e-4 and `max_lr` to 0.4.

Since the model predicts a user, recipe pair, we have implemented a function (`get_recommendations_for_user`) that returns a list of tuples containing the recipe_id and predicted rating.

In the next milestone, we will ensure that the model actually works for our purposes, and begin integrating it into the website.

Feature Changes

No feature change is applied to the proposal during Milestone 2.

Challenges

- When writing the function that takes in the input image folder and outputs the predicted ingredient, Jane and Alina ran into a problem where the path of the images was not corresponding correctly with the function. To tackle this challenge, the members iterated through potential solutions and solved the issue by placing each image into a separate folder.
- When uploading the dataset using Google Drive mounting, the runtime for very few epochs was very long. To solve that issue, we copied the zipped dataset files to the VM and this resulted in much faster runtimes.
- When optimizing the recommender model, we found that the recommender model is very sensitive to the change of max learning rate. So we must adjust it very carefully to avoid overtraining.
- The recommender model has a few problems:
    - Both models we tried require all users to be inputted before training. That means if a user of our website rates a few recipes and wants recommendations based on that, we must first add those ratings (associated with the new user) to our dataset, train the model, and then produce the recommendations for the user. This method is slow and clunky. While this is certainly possible, the team will go to office hours to see if there is a better model we should be making use of.
    - To use the model, we must give the model a user, recipe pair. Assuming we've solved the first issue and that the user already exists in the model, we must calculate the predicted rating for each recipe, and then sort them by rating. This may be the best way to do it (and we've implemented this in a function called `get_recommendations_for_user`), although perhaps with a different model we can ask it for the top recipes immediately.

Member Tasks

<u>Classification Model</u>

- Both Jane and Alina were equally involved in implementing and training the classification model, all the code for the model was written in a pair programming format and they discussed solutions while debugging.

Recommender Model

- Harrison was in charge of building the recommender models and writing the function `get_recommendations_for_user` and Bowei was in charge of optimization.