Recipe Recommender Milestone 4

UBC-CPEN291/project-team-apatosaurus

Harrison Mitgang (hmitgang), Bowei Ren (boweiren), Alina Shabanova (alinas2000), Jane Wu (jwu611)

Current State

The classifier, recommender, and website work!

Deployment

As outlined in “Feature Changes” and “Challenges”, we struggled to deploy our app to Heroku or Google Cloud services. Instead, we will only run the website locally.

Website

The focus of this milestone was combining the classifier and recommender webapp. This was done by merging the server.py files and serving the recommender webpage after the classification was completed. The ingredients were passed along to the recommender, and recipes were filtered out based on which ingredients were received. In other words, if a user uploaded a picture of salmon but not olives, then the recommended recipes will only include recipes that *do* include salmon, but *do not* include olives.

RF-Recommender

The secondary recommender system is built based off this paper: https://www.researchgate.net/publication/224262836_RF-Rec_Fast_and_Accurate_Computation_of_Recommendations_Based_on_Rating_Frequencies

To quote the paper, the algorithm “generates predictions simply by counting and combining the frequencies of the different rating values in the usual user-item rating matrix.” Experimentally, we found that the model is much quicker than matrix factorization. The paper suggests it is an efficient and effective algorithm for creating predictions on sparse datasets like the recipe interactions dataset. We implemented this not to replace the matrix factorization, as it has not been tested extensively, but merely as a “bell and whistle” to make using the webapp quicker.

Feature Changes

We are no longer deploying our web app to an external hosting site. Instead, it will only be run locally. See “Challenges” for more details.

Challenges

- Deployment
    - When deploying to Heroku as outlined in our proposal, the “slug size” of our project was too big. This is likely due to the vast size of the ML models and libraries. Even when offloading the training data (required for the recommender) to an AWS S3 bucket, the slug size was still too large.
    - We then tried Google Cloud App Engine and even after hours of finiking, we ran into a similar memory error.

    - Given this seems like a common problem amongst hosting services, including AWS, we have decided the most effective way to deploy this web app for now is locally.
- Retraining the recommender system takes a *long* time. To help supplement this, we’ve added a second, less tested algorithm called RF-Rec to make the UX better for standard users.

Member Tasks

- Jane and Alina worked on upgrading the website to allow inputs of more than 1 image, and on combining the classification and recommender websites into one final one.
- Harrison continued Jane and Alina’s work in combining the websites, and ensured the two components worked together seamlessly. He also implemented a secondary recommender system based on rating frequency for quicker recommendation and implemented recipe filtering based on what ingredients were uploaded. He also attempted (without success) to deploy the web app to Heroku and then Google Cloud App Engine. He also created a readme on deployment instructions, and a blurb on the website with instructions and purpose.
- Bowei worked on the style optimization and css file of the final web app.