May 9, 2021

Milestone 3 - Team Ornithomimus

State of the Project

The Flutter application is approaching its final form, with all major features implemented. On the profile screen, the user can now select their preferred distance, elevation gain, and difficulty using range sliders and toggle buttons. On the hikes screen, hikes are shown in a tinder-card format. Each hike has multiple photos which can be scrolled through, and swiping a card to either side marks it as “liked” or “disliked”; all rated hikes are moved to a list to send them to the backend, and liked hikes are added to the matches page. As the user swipes through the available cards, an API call is periodically made to add more hikes to the stack. On the matches screen, the user can view their liked hikes in a list format. Tapping on a hike brings up more details, including larger images, hike information, and a link to the AllTrails page. For a preview of the application, see reports/M3/app_showcase.gif. In addition to the visible changes, we have made temporary methods to simulate API calls; these methods return JSON-formatted strings, which are then handled by the application as if they came from the backend.

The backend and API are now functional. The API is able to create new users, recommend hikes to users, and receive reviews that will refine the recommendations shown to the user. When the API is running, it receives a request and the user’s username to locate the user in the system to manage that user’s profile specifically. When a request is made for a user to get a hike, it returns a list of hikes which were prepared by the machine learning models.

The recommender system has been implemented using cosine similarity to find hikes that are similar to the user’s “ideal hike”. Every user has a preference vector that contains 19 encodings for different hike attributes, such as length, difficulty and select keywords. This vector is updated after a hike is “liked” by taking the information from the hike. Similarly, each hike has an attribute vector of the same size, containing the same information. When a hike is requested, the cosine similarity between the user’s preference vector and each hike’s attribute vector are calculated and the hike with the highest similarity is returned.

The image classifier has been updated to accept hike names as inputs. It now finds all images associated with a hike, gets a score for each image, and makes a prediction for the hike based on the mean of all its image scores.

An important note: the work done in M3 builds on top of the work started in M2, so all of the code for our project is in the M2 folder. Please see the commits made after M2 to branches *back-api*, *flutter-app-M3*, *recommender-model*, and *image-classifier* for the most recent updates.

Feature Changes

In our past milestone report, we had stated that we were switching to a collaborative-filtering recommendation system that would recommend hikes that similar users have liked. We faced challenges getting the user data from AllTrails, so we switched back to a content based recommendation system. This recommendation system does not require other user data and uses a preference vector with embeddings to find hikes that are similar to hikes the user has liked in the past.

Current Challenges

Currently, our biggest challenge is figuring out how to combine our two machine learning models to produce a single output. Since we have an image classifier which produces a score for two categories, and a recommender system which produces a cosine similarity between a hike and preference vector, we cannot directly combine the two models. Our plan to solve this problem is to connect the models in series, producing a shortlist with the recommender which gets approved by the classifier. When the application calls the API to get more hikes, it also posts the list of hikes that were rated by the user; this list of ratings is used to train both the recommender and the image classifier. Then, based on the updated model, the recommender creates a list of the hikes with the best cosine similarities to the user's preferences. This shortlist is then given to the image classifier, and each hike is either approved or rejected. The resulting approved list is then sent back to the application in response to the API GET request.

Team Contributions and Upcoming Tasks

Andrew Forde

Contributions: Updated profile page of app with user-selected preferences. Updated Matches screen to list form with selectable hike pages. Added multi-image view of Tinder cards. Added temporary methods to simulate API calls.
In-progress: Combining ML models, integrating components for unified testing.

Taylor Mcouat

Contributions: Updated the back-end of the API to create new users, recieve reviews based on previously seen hikes, and recommend hikes based on past reviews. Built the recommender system.

In-progress: Adding the image classifier to the API to refine the recommendations further. Testing and refining the API.

Leo Sun

Contributions: Refined the backend in terms of user settings.
In-progress: Recommender model and testing preparation.

Jayden Wong

Contributions: Changed app to support user authentication using firebase.
In-progress: Storage of user preferences, account details, and matched hikes using firestore cloud.