Sprint#5 Progress report

Team6

1. What we have done

- When a user change the period of travel, existing day blocks which contains a day title should be changed. Also travel items like transportation items, restaurant items belong in day block should be moved to another day block. This functionality is implemented in `CreateTravel` Page according to our user story.
- Searching place information is important to manage user's travel items. Searching functionality is implemented with Google Maps API, but request fee is negligible to our service. We implemented internal cache to save all response of the Maps API and tried to reuse this cache in next user requests.
- We implemented recommendation feature, by hybrid method of content-based, and matrix factorization. We put additional block distribution similarity to improve recommendation. Feature is implemented in local, and will applied after backend is done.
- Content-based recommendation is done by calculating similarity. Universal sentence encoder calculate the embedded vector of plan, and plans with high cosine similarity will be recommended.
- Matrix factorization is implemented for implicit feedback, which there is no explicit score. We only see the user behavior like watching plan, like plan, comment plan, fork plan. This behavior will be reflected to recommendation by confidence, and matrix factorization.
- We think that block distribution of plan represent the travel's characteristic like activeness, or focus on eating, etc. So we add additional similarity score for this.
- We implemented UI for 'travel settings page'.
- Link functions are attached to buttons and travel blocks so that users can navigate our service with mouse clicks.
- We implemented backend for 'Travel' model and its children models like 'TravelCommit', 'TravelDay', 'TravelBlock' to provide commit and merge feature for mono-user and change of the settings for 'Travel'.
- We implemented backend API for travel which supports tag-searching and serves 'Travel' in order of popularity and recency.

2. What we are planning to do for next sprint

- Travel detail page will be similar with travel create page, but there would be no edit window. The travel detail page will be implemented similar with create page for next sprint.
- When a user create a travel, the user wants to check the path of his/her travel plan frequently. Google maps API served the routed image when we request the list of latitude, longitude. We will connect the maps api and our backend server to check the path of user's travel.
- We will implement actual recommendation on real service, and design the architecture for distributing model train, and model update on server.
- All other pages will be implemented, including 'edit user info page' and some additional pages for version controlling of travel plans.
- We will improve UI of our service to make it more user-friendly. Custom UIs for temporary use will be replaced with Material-UI components.
- We will make lots of fake data to make our service look more realistic for testing. Fake data are also needed for testing recommendation service.
- Backend for 'Travel' model will be completed to supports version control and sharing features among multiple users as well as forking.

- We are planning to write tests for backend API for ‘Travel’ with various user stories.

3. Not implemented features

- Some pages like user setting page or create travel page should be implemented with save image. Django supports to save image in database through pillow library, but we did not implement this feature yet.
- When a user create a new travel plan, we want to add the plan in google calendar. This feature requires the google login feature and access authorities about google calendar service. The usage of google calendar api would be explored the possibilities.
- Functions for connecting frontend components to backend requests should be implemented.
- Backend for ‘Travel’ model with multiple collaboraters should be implemented.

4. Organization issues

- We need to distribute the calculation for the model update and inference for the recommendation engine on the server. Model update take some minute so it would be hard to concurrently run on same server with service.
- We managed our backend server and frontend server in azure server, but the token was expired. The MySQL database was served in those server, so we tried to move the new database, bit this is hard to move in order to azure service.
- We have to manage many issues for designing backend for ‘Travel’ model to support version control. We have to resolve issues like detecting merge conflicts between two commits. So, we are trying to resolve them with the tree structure that keeps track of past version.

5. What tests did we prepare and not testing yet

- We have only written minimum tests for each page, so we have to write full-featured test for all pages.
- We need to test for the various type of plans for the recommendation. There could be blanks on the plan, and incomplete plan, etc. We need to check recommendation feature gives some result on these plans. It would be hard to measure the recommendation quality, but it always need to give some result without error.
- We are preparing tests for backend API for ‘Travel’ model such as commit changes and merges.

6. Coverall Report

Overall 72%

https://coveralls.io/github/swsnu/swpp2019-team6?branch=master

The tests for frontend pages with various user stories will be added to lift up the overall coverage.