AI Dining Suggestion App

By Bao Pham
CS 297 Fall 2018

Possible routes

- Start collecting data from users with a simple UI app similar to Tinder
- Define what data needs to be collected and how to connect them
- Once data is enough to train, start training a model using Tensorflow (more on this later once get to training part)

- Collect data available with Yelp and Google APIs. Specially focusing on reviews data
- Start training a model with that data
- Define what data needs to be collected and how to connect them
- Make a simple UI app to collect more data from users and update them as users have more inputs

Deliverable 1

Create an API to query and process restaurant related data from Google and Yelp given a specific GPS location such as names, price levels, food types, images, etc. (possibly with Facebook authentication system for users if time permits).

Simple UI is needed to show results.

Architecture

1. Get GPS location when app opens
    a. Later can ask some filtering questions to filter restaurants
2. Assign unique string to each device
    a. Later can implement authentication system
3. Make queries to Yelp and Google APIs to query list of restaurant nearby
    a. Remove duplicates
    b. Show images from Google (10) Yelp (3)
    c. Possibly use paid services to get more images

Implementation 1

1. Acquire tokens from Google and Yelp APIs

```
module.exports = {
    MAPS: {
        API_TOKEN: 'AIzaSyOk3UsX_dUc_IXiWi_oRnemBEN38LFs3Ik',
        END_POINT: 'https://maps.googleapis.com/maps/api/place'
    }
}
```

```
module.exports = {
    CLIENT_ID: "WCXgJybBBoQN0PJeH47AMQ",
    API_TOKEN: "2PhEBDExL44JbeZB5exT1qEz-0bvK4rAEorQWlnCPjetnplDN_rGFl_oou_NHsDddzYeJ8fgTLikHh2cjG
    URL: "https://api.yelp.com/v3"
}
```

2. Create service classes to handle requests to Google and Yelp for restaurant info

```
getPlaces({lng, lat, radius, minPrice, maxPrice}) {
    let type = "restaurant";

    let query = {
        key : apiToken,
        location : `${lat},${lng}`,
        radius,
        type,
        keyword : "",
        minPrice,
        maxPrice
    };

    let url = `${apiEndPoint}/nearbysearch/json?${querystring.stringify(query)}`;
    fetch(url)
        .then(res => res.json())
        .then((responseJSON) =>{
            let photoQuery = {
                key: apiToken,
                maxwidth: 1000,
                maxheight: 1000,
                fields: 'photo'
            };

            Promise.map(responseJSON.results, function(restaurant){
              photoQuery.place_id = restaurant.place_id;
              let photoUrl = `${apiEndPoint}/details/json?${querystring.stringify(photoQuery)}`;
              console.log(photoUrl);
              fetch(photoUrl).then(photos => photos.json()).then((photoResults)=>{
                console.log(restaurant.photos.length + "======" + photoResults.result.photos.length);
```

```
searchForRestaurants({lng, lat, radius, minPrice}){
    let query = {
        latitude: lat,
        longitude: lng,
        radius: parseInt(radius),
        term: "restaurants"
    }

    if(minPrice > 0) {
        for(i = 0; i < minPrice; i++) {
            query.price += '$'
        }
    }

    const url = `${config.YELP.URL}/businesses/search?${querystring.stringify(query)}`;
```

Implementation 2 : Make additional queries to get the images

```
getPhotoUrls({restaurants, maxWidth, maxHeight}) {


    let query = {
        key: apiToken,
        maxwidth: maxWidth || 1000,
        maxheight: maxHeight || 1000,
        fields: 'photo'
    };

    var promises = restaurants
        .filter(function(restaurant) {
            if (restaurant.place_id == null) {
              return false; // skip
            }
            return true;
        })
        .map((restaurant) => {
            query.place_id = restaurant.place_id;
            let photoUrl = `${apiEndPoint}/details/json?${querystring.stringify(query)}`;
            return fetch(photoUrl)
            .then(photoResults => photoResults.json())
            .then((response) => {
              if(restaurant.image_url != null){
                restaurant.image_url = response.result.photos;
              }else{
                restaurant.photos = response.result.photos;
              }
              return restaurant;
            })
        });

    return Promise.all(promises).then(results => {
      return results;
    });
}
```

Implementation 3: Merge results from both APIs. Remove duplicates and set up routes

```
//Libs
const express = require('express');
const router = express.Router();

const restaurantController = new require('./restaurant.controller');

//Controllers
const controller = new restaurantController();

router.get('/search', controller.getRestaurants);
router.get('/:id', controller.getRestaurant);

router.post('/like/:id', controller.likeRestaurant);
router.post('/dislike/:id', controller.dislikeRestaurant);
router.post('/favorite/:id', controller.favoriteRestaurant);

module.exports = router;
```

```
searchForRestaurants({lat, lng, radius, minPrice, maxPrice, maxHeight, maxWidth}) {
    let googleRestaurants = googleService.getPlaces({lat, lng, radius, minPrice, maxPrice})
        .then(json => json.results)
        .then(restaurants => googleReduceRestaurants(restaurants, maxHeight, maxWidth))

    let yelpRestaurants = yelpService.searchForRestaurants({lat, lng, radius, minPrice})
        .then(results => results.businesses)
        .then(yelpReduceRestaurants)

    return Promise.all([googleRestaurants, yelpRestaurants])
        .then(mergeSearchResults)
}
```