Popfood

COSC2626
Assignment 2

Phillip Guettaf
s3607153

Semester 2
2019

Table of Contents

Links

Live site
https://popfood.ap-southeast-2.elasticbeanstalk.com/

Git repository
https://github.com/rmit-s3607153-phill-guettaf/popfood

Introduction

This app was inspired by a similar app mentioned in the lecture, that shows and ranks local restaurants based on public Instagram posts in the area, and Steam's rating system, that shows how the overall rating of a product is different from the average of specifically recent ratings. After reading a number of restaurant reviews, you may notice a number of them give a rating at odds with their comment, e.g. a 3-star review with a long comment praising the restaurant, with one small thing that went wrong. This app is an attempt to normalise the review/comment disconnect.

Popfood retrieves a list of local restaurants from Zomato and checks these against Google. The Google and Zomato reviews are aggregated with a scaled sentiment analysis result from the last 5 reviews from both Zomato and Google and ranks restaurants accordingly. The results are saved to a database, and each restaurant's rating (if already in the database) is updated accordingly.

The database returns the top restaurants it has for the app to display, alongside the restaurants in the area, as well as a map showing the locations of the local restaurants.

Related work

Zomato

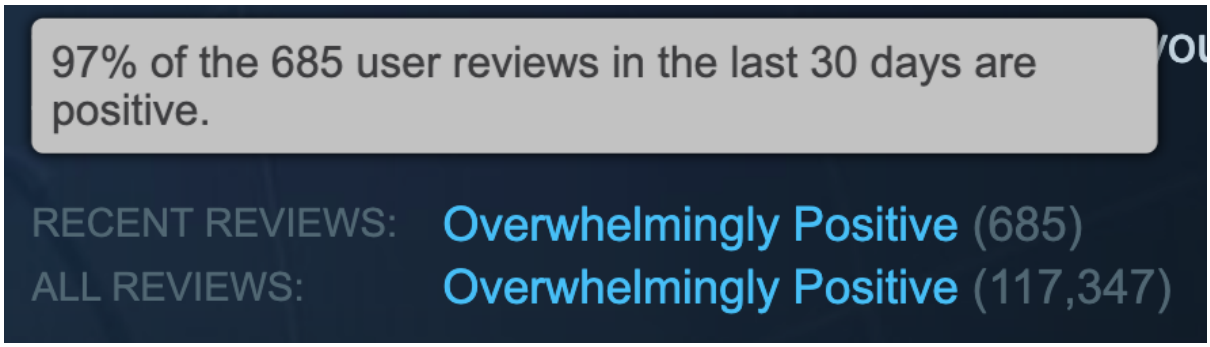
Zomato, by its nature is built around reviewing restaurants. Zomato do normalise their ratings over a scale that grants greater power to more frequent users however and doesn't check the content of comments for sentiment against the rating itself.

Google Reviews

Google's main similarity to this app is in its presentation of a map of local restaurants along with ratings. These ratings, as discussed above are differently calculated from the review system of Popfood.

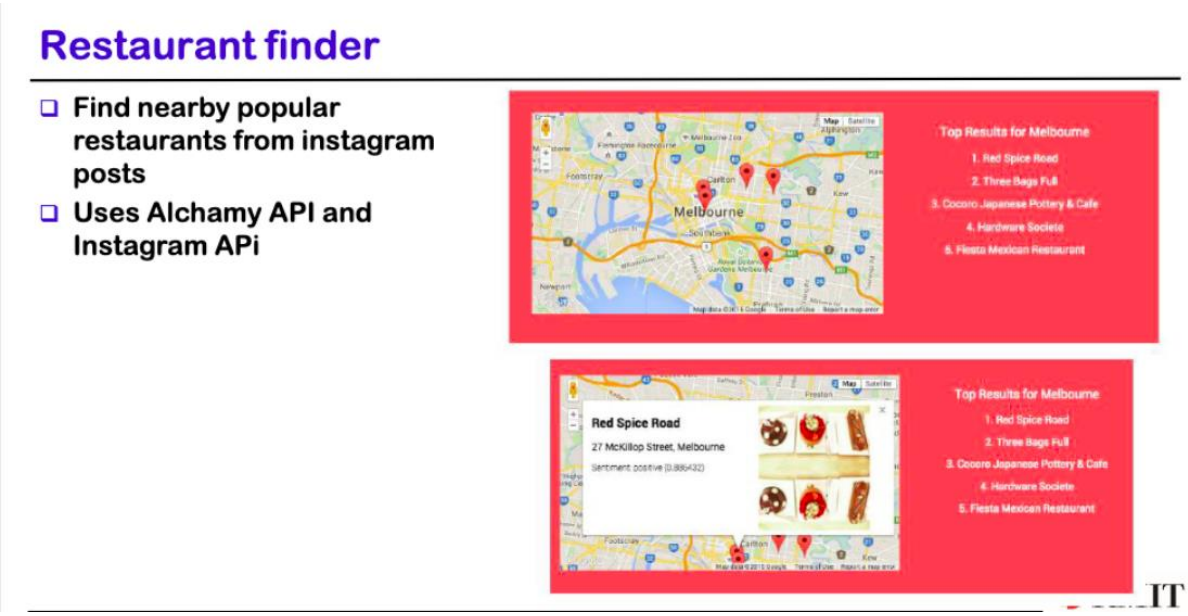
Steam Rating System

Steam's review system for games is the only other found that includes a modification of ratings, specifically showing the most recent reviews, as below:

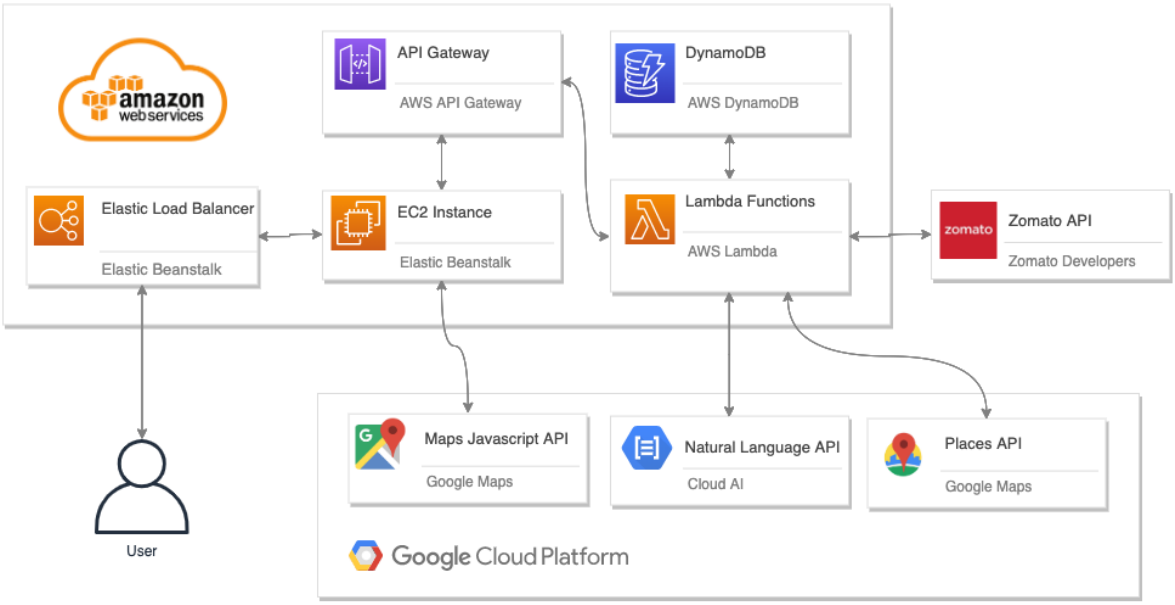


Instagram Ranking

The Instagram Restaurant Finder shown in the Week 8 Lecture was the prime inspiration for this project, and has a similar basis:



Software architecture



Components

Elastic Load Balancer

The Elastic Load Balancer is the load balancer sitting above the EC2 instance, distributing traffic to the app.

EC2 Instance

The EC2 instance is where the actual app runs.

API Gateway

The API Gateway, linked to the Lambda Functions, handle traffic from the app to the API written for Popfood: most data requests flow through here to various servers online to collect data for processing, and send data for saving.

Lambda Functions

The Lambda Functions are where the actual API functions written for Popfood run.

DynamoDB

This is the database of restaurants processed by the application.

Zomato API

The Popfood API requests data from the Zomato API, which returns an array of restaurants, including locations, ratings and reviews from the Zomato servers

Maps Javascript API

The Maps Javascript API is the only API accessed by the application without the API Gateway: this is to display the map. Direct access was used to speed loading time, so the user wasn't left with a blank screen while other data was loaded.

Cloud Natural Language API

The Cloud Natural Language API receives a text block of comments and returns a Document score for the text ranging from -1.0 to 1.0, indicating the sentiment associated with the text.

Places API

The Places API collects place information from Google servers based on the location of the restaurants first grabbed from Zomato.

Popfood

Popfood is a web application written in React, run on Node.js. The app was bootstrapped using Create React App [1] and uses the Google Maps React package [2].
On loading, Popfood completes the following tasks:

1. Get the user's location
2. Send a request to Zomato for restaurants in the area
3. Use the data from this to send a request to Places for the Google information on these restaurants
4. Aggregate the reviews for these into a single text block and send to Cloud Natural Language for a sentiment analysis
5. Calculate the "Average Rating" across these platforms
    a. This is done by scaling the Document score to sit in a range of 1.0 to 5.0, then taking the average of the Google Rating, Zomato Rating and this score for the Average Rating of the restaurant
6. Save each restaurant, with its Google, Zomato and Average Rating information to the DynamoDB
7. Retrieve the list of most popular restaurants from the DynamoDB
    a. This step is done asynchronously, and as it is fast, often happens before the database is updated with new entries. Refreshing the page may change this list.

Data

Restaurants

For ease of access to data, the restaurant data is stored as an array of JSON objects:

```
[
   Restaurant1: {
         averageRating: Float,
         googleInfo: {
name: String,
location: {},
...
         }
         zomatoInfo: {...},
         reviews: [{...}]
   },
   Restaurant2: {...}
]
```

API

The API written for this is broken down into six functions, written in Javascript and run on AWS Lambda. The functions are accessed through an AWS API Gateway.

/getZomatoRestaurants

This function gets a list of Zomato Restaurants in the area

/getZomatoRestaurantReviews

This function uses this information to get more detailed information on the Zomato servers about the restaurants, including the last five reviews.

/googleRestaurants

This function takes a Zomato restaurant as input, and returns the Google Place result for the restaurant, including the last five reviews

/getCommentSentimentReviews

This function takes the aggregated comment reviews from a restaurant and sends it to the Natural Language API, which returns a document score.

/saveRest

This function takes an array of restaurants and saves them to the DynamoDB

/getDynamoRestaurants

This function scans the DynamoDB, sorts the table by average rating and returns (up to) the five highest scored restaurants.

DynamoDB Table Structure

Table attribute	*Name*	*Description*
Primary key	`zomatoID`	This is the zomato place ID for the restaurant, used to ensure a unique identifier at the top level of the table.
Secondary (sort) key	`aveRating`	This is the calculated Average Rating of the restaurant, used as the sort key to enable low-nested searching for top-rated restaurants
Info	`info`	This is the complete restaurant object used by the application stored as a JSON string. This was used to simplify loading and saving and reduce the table overhead.

Developer manual

API Keys

The first step to running Popfood is to get some API keys for Google, Zomato and AWS!

Google

Create an API key for your project:

1. Create a new project in Google Cloud Console
2. Navigate to the APIs & Services console
3. Click "Credentials"
4. Click "Create new credentials"
5. Click "API Key"
6. Save the key!

Enable the Maps Javascript, Places and Natural Language APIs

1. Click "Library" in the sidebar of the APIs & Services console
2. Search for "Maps Javascript API" and click "Enable"
3. Repeat steps 1 and 2 for "Places API" and "Cloud Natural Language API"

Paste your API key into the value field wherever "gApiKey" is marked as a variable in the project

Zomato

Create an API key for the project

1. Go to developers.zomato.com/api
2. Click "Generate API key"
3. Log in with your Google Account

Paste the API key into the value field wherever "zApiKey" is marked as a variable in the project

AWS

AWS CLI

Follow the instructions online to install the AWS CLI [3]

Elastic Beanstalk

Use the GitHub scripts to install the Elastic Beanstalk CLI. [4]

Credentials

Create security credentials for your app

1. Navigate to the AWS console
2. Click your email in the top left, and click "Security Credentials"
3. In the left sidebar, click Users
4. Create a new user
5. Assign the user to the group
6. In the left sidebar, click Groups
7. Create a new group

8. Assign the user we made earlier to this group
9. Click on this user
10. Click "security credentials" in the top tab bar in the user screen
11. Click "Create Access Key"
12. Save these to your computer and keep them safe
13. Put the access key and secret access key in the lambda function "getDynamoRestaurants"

Lambda

Set up the Lambda functions

1. Navigate to AWS Lambda
2. Click "Create new function" > "From scratch"
3. Paste the text from the appropriate file in the lambda directory of the source code folder into the window.
4. Click Save!

API Gateway

Set up the API Gateway for Popfood

1. Create a new API in the API Gateway, with the name of the API you are adding
2. Create a new resource and method for this resource (POST for all methods except /getDynamoRestaurants, which is GET)
3. Add your lambda function for this method. Be sure to enable CORS under Actions.
4. Click Deploy and follow the prompts!
5. After creating a stage and deploying the API, copy the invoke link of the stage and paste into src/api.js in the apiPOST and apiGET functions where a URL is needed.

DynamoDB

Create a DynamoDB table to store data

1. Navigate to DynamoDB on the AWS console
2. Click "Create new table"
3. Name the table "rest"
4. For the Primary key, name it "zomatoID", with type String
5. Create a sort key by clicking the check box, and name it "aveRating" of type Number
6. Click create

The database will automatically create new fields for "info" when data is pushed to the table.

EB CLI

Follow the instructions [5] to set up your project configuration for EB CLI
Follow the instructions [6] to create your environment and deploy the code.
Then you can use 'eb open' to go directly to your live project!

User manual

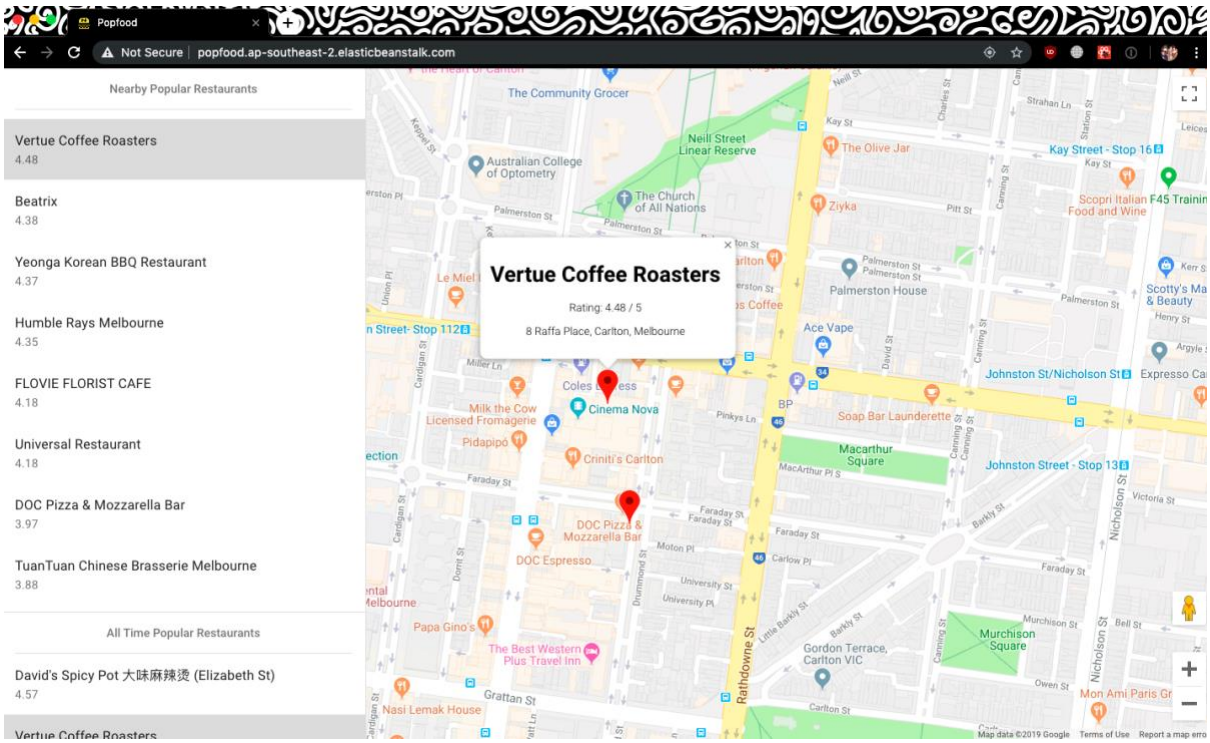
Go to the URL of the live site:

https://popfood.ap-southeast-2.elasticbeanstalk.com/
Be sure to connect via https, not http. If there are security warnings, it means the SSL certificate has expired. You can continue safely, although your browser may warn you not to.

Click "Allow location" when your browser prompts you.

The map will auto-populate with markers, and the sidebar with list items.

You can click on a list item in the sidebar to centre the map on that restaurant, and click on a marker to show the rating, name and street address of the restaurant.



References

[1] Facebook, "Create React App," [Online]. Available: https://create-react-app.dev/.

[2] Full Stack React, "Google Maps React," [Online]. Available: https://www.npmjs.com/package/google-maps-react.

[3] Amazon Web Services, "Installing the EB CLI," [Online]. Available: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html.

[4] Amazon Web Services, "AWS Elastic Beanstalk CLI Setup," [Online]. Available: https://github.com/aws/aws-elastic-beanstalk-cli-setup.

[5] Amazon Web Services, "Configure the EB CLI," [Online]. Available: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-configuration.html.

[6] Amazon Web Services, "Managing Elastic Beanstalk Environments with the EB CLI," [Online]. Available: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-getting-started.html.