CSE 546 — Best Dish Around You

Venkat Krishna Sai Kowkuntla
vkowkunt@asu.edu
1218482044

Pavan Kalyan Reddy Thota
pthota3@asu.edu
1218436466

Sai Teja Vishal Jangala
sjangala@asu.edu
1218332037

1. Introduction

This is a problem that we have adopted from the experiences and issues we have faced in the past. Sure, everyone might have faced it, and we thought solving it by an application will target large audience who have similar concerns. Whenever we visit a new city, we usually taste different kinds of food in different restaurants around the city.
However, it would be a very difficult task to guess the best dish in that area. We end up trying different dishes from various restaurants based on restaurant ratings. But there might be a scenario where the popular dish in that area is from a restaurant with a low rating. Because of the low rating, we might miss tasting the popular dish of that area.
In conclusion, Restaurant Ratings might not be the only reliable factor in choosing the best dish in an area. Our application will give the best dishes in the restaurant present in the restaurants nearby.

Existing technology: We researched online and found that we can rate only a restaurant online through Google, Yelp, Zomato, etc. There is no medium to rate a particular dish in a restaurant. There are no applications that can find the best dish among different restaurants that one can eat in a particular location.

Idea: To solve the above problem, we came up with a solution to automate dish ratings of each restaurant's dishes around a particular location by scraping user reviews from Google, Yelp, Zomato, etc. By this, we get popular dishes that are worth tasting in all restaurants around a particular location. We will further define some weights to compare the popular dishes from all restaurants rated above, thereby giving popular dishes available in that area. We want to build a mobile application that serves as Front-End for our app, that takes the user's current location and passes it to the Google Cloud where the Back end is implemented. Our Backend server fetches all the restaurants available at that location through REST API calls. We will further fetch the reviews of each restaurant and perform Analysis and score dishes in the menu according to reviews. At last, we weigh the scored dishes to output the best dish in that location which is worth trying.

Result: Through our analysis, we can find the best dish that a user can taste in any location just by turning on his/her GPS. Thereby, not just picking the best restaurant based on ratings, but by picking the best dish around his/her vicinity.

2. Background

Description of the background of the problem.

I. Most Relevant Technologies to the problem:

1. Our application requires a lot of reviews for it to be processed and should be always available. For the application to be always available and for it to provide it uses to all the users every time, it needs to auto-scale based input requests (input requests by the users) and the reviews that it must process to get the necessary information. To address all the mentioned issues, we have used

Google App Engine (as a part of Google Cloud Platform) to host the application and provide autoscaling.

2. The review processing is the main part of the application. The reviews that we have for every restaurant are processed by our model, (our model is an NER model, and the clear description of the model is provided in the later parts of this report). Our model gives out the best dish in the restaurants that are located within the range of 25 miles. To train the model we have used Google AutoML for food Named Entity Recognition (NER) and sentiment analysis on the food entities.
3. Since the application uses reviews as the main part that has to be process, we looked up for a platform that has most reliable reviews and also that has reviews for all the restaurants. We have found that Yelp has large number of reviews, and the reviews for all the restaurants. So, we have taken reviews of all the restaurants in Phoenix location from the Yelp and created a database using Atlas MongoDB which uses a google cluster. We have also used Yelp REST API to find the restaurants around the location of the user. Concluding, we have used Yelp reviews and used MongoDB to store these reviews, and we have used Yelp REST API in the real-time to get the restaurants around the location of the user.
4. Since the user of this application can be present everywhere and for the user to utilize the functionalities of the application properly at any time and anywhere, we thought of implementing a React Native application for user’s interaction. We chose a React Native application because it can be used on any Android, iOS and as a Web Application.

II. Why is this problem is important:

This problem is important and should be addressed because there might be people who go to different places, states, and countries. Each place has its importance and it culture. Food is important to understand the culture, but there might always be a question on what food to eat and which restaurant to choose. There might be restaurant that is very famous for a particular dish but least famous for all other dishes in that restaurant. When the users will post bad reviews for that restaurant and when very a smaller number of users will post good reviews for that one dish, the overall ratings for that restaurant will be reduced, and the dish will never be recognized.

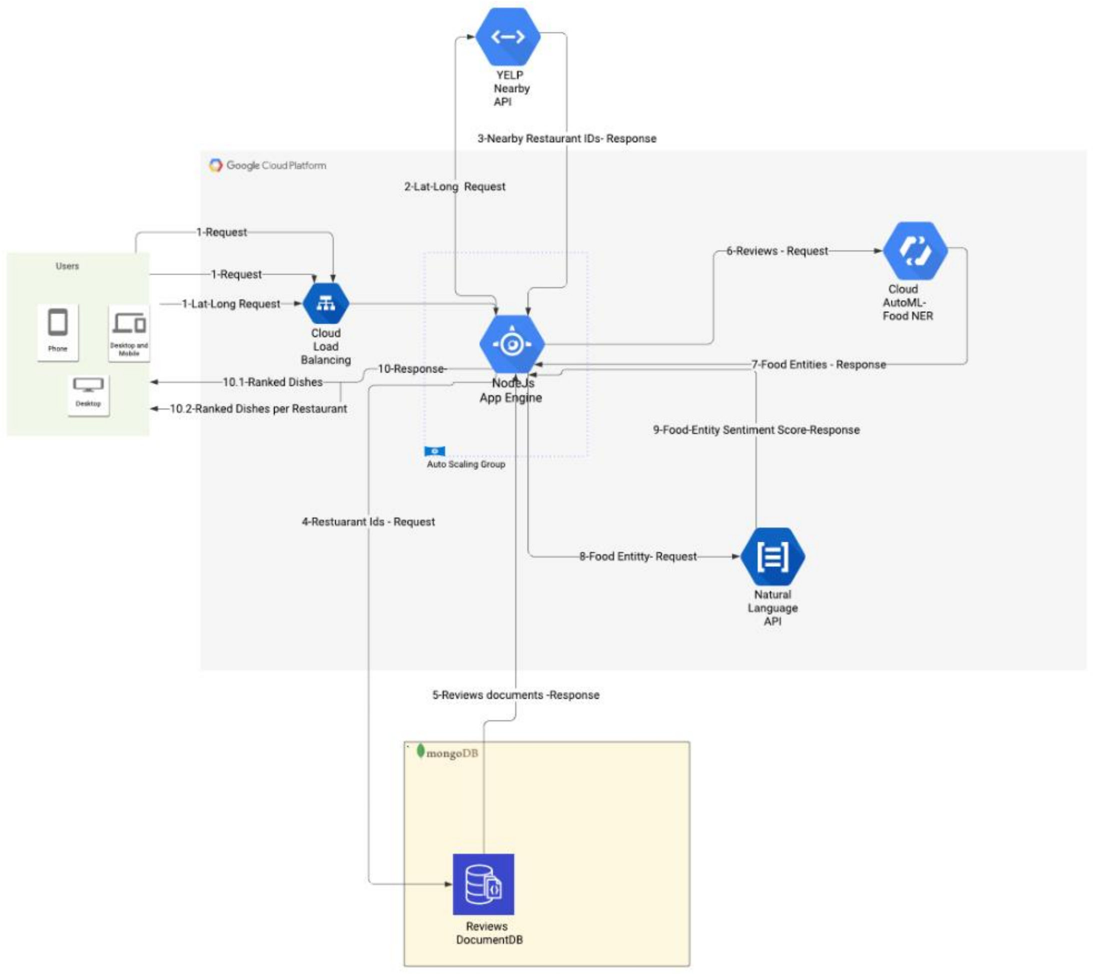
Our application here will take out that one popular dish in that restaurant, ignoring all other reviews through Food Named Entity Recognition and sentiment analysis. This problem will be a common problem for a user who is new to the place or the user who is travelling or also for the users who has stayed in a place all their life but never looked a restaurant beyond it reviews, because our application suggests dishes in that restaurant. We have investigated it and considered to address it. Additionally, this problem has not been addressed anywhere before by any other application.

III. Why are the existing solutions not sufficient:

All the existing applications now only concentrate on the reviews and rating given by the user, but we never get time to see what is that one aspect that is bad or good in those bunch of reviews. However, our application concentrates on processing those reviews to get useful insights in the data and provide it to the users. This application has been inspired from the problems that we see every day, and we thought implementing it might target very large crowd who are looking up for good food. This might also help many local small restaurants to increase their value of business. Since there is no proper solution for this problem, we have implemented it.

3. Design and Implementation

I. Architecture Diagram of our Application:



Description of the design and implementation of our solution:

Major Components in the Architecture:

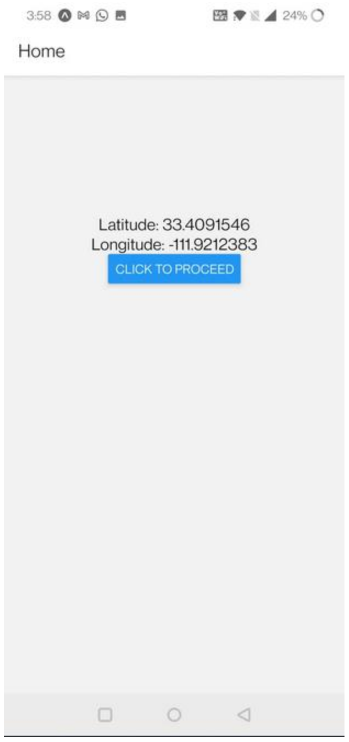
1. React Native:

We have built the front-end of our application using React Native. We have used this React native because the application should be very handy for the user to check the restaurants nearby. For the application to be very handy, the best way is a Mobile Application. For this mobile application to work in all mobile operating systems, we have used React Native to build it. Additionally, this React Native will also enable us to use this application as a web application.

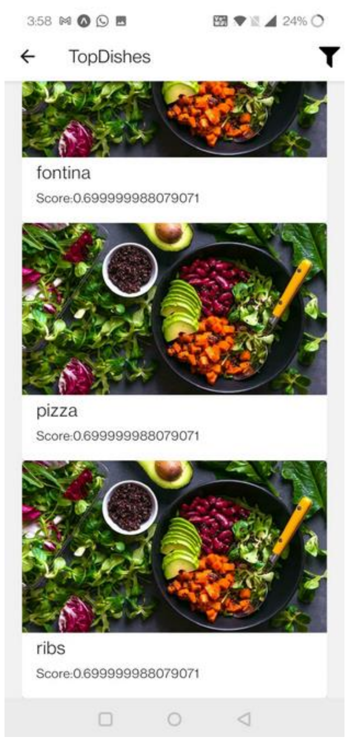
There are three parts of our React Native application, the first part is Home Page, this home page will read the latitude and longitude values of the device's location and send it for further processing while displaying it on the mobile. Based on this Lat-Long values, the restaurants in radius of 25 miles of the user are considered.

From these restaurants, the top dishes are shown in the second page of the React Native application, which is a TheBestDish page, here the Best Dishes within 25 miles of the user will be shown.

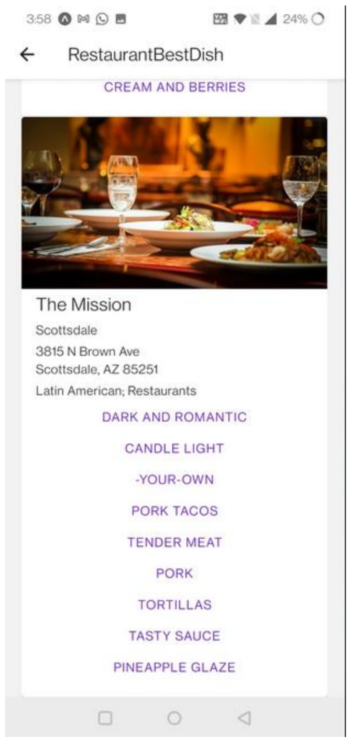
The third page of the application is TopDishesAmongRestaurants page where the names of the Restaurants are shown, along with the names of the Restaurants, the top dishes in each of the restaurant will be shown.



First Page (Home Page)



Second Page (TopDishes)



Third Page(RestaurantBestDish)

2. NodeJS App Engine:

As this application is hosted in Google Cloud Platform, in which we used Google App Engine (GAE) which has PaaS resources to make our application available and to perform auto-scaling. In this GAE, we installed a NodeJS server to host our application, making it a NodeJS app Engine. We have provided the backend services using this NodeJS App Engine, where we have written the code in NodeJS to implement our features and to fetch the required restaurant details and reviews from the MongoDB Atlas cluster which we used to store all the restaurant reviews.

Every backend service that we have implemented in this application is hosted and executed in this NodeJS App Engine. Additionally, this NodeJS App Engine makes intermediate calls to get the required results from Yelp REST API, Cloud AutoML and Sentiment Analysis API.

3. MongoDB Atlas Cluster:

We have used MongoDB Atlas Cluster as the Database of our system, in which we stored around 2,29,000 reviews given by user on different restaurants in the Phoenix location.

This is small overview of our database. Our Database contains the yelp reviews in the Phoenix location, these are the reviews given by all the users and all these reviews have a unique “review_id”. All these reviews will be present in a CSV file, and each row will have the details of the reviews given by the users. A user can give multiple reviews and there will be multiple restaurants, also a user can give multiple reviews for a same restaurant. However, for each review, has a unique review_id. Our data (in CSV format) is converted to json format and uploaded to MongoDB as documents.

Now this database is loaded into a MongoDB Atlas Cluster which uses Google Cluster, and NodeJS App Engine will extract the restaurant information and reviews as per the Latitude and Longitude values obtained by the React Native application and as per the Restaurants obtained by the Yelp REST API.

4. Yelp REST API:

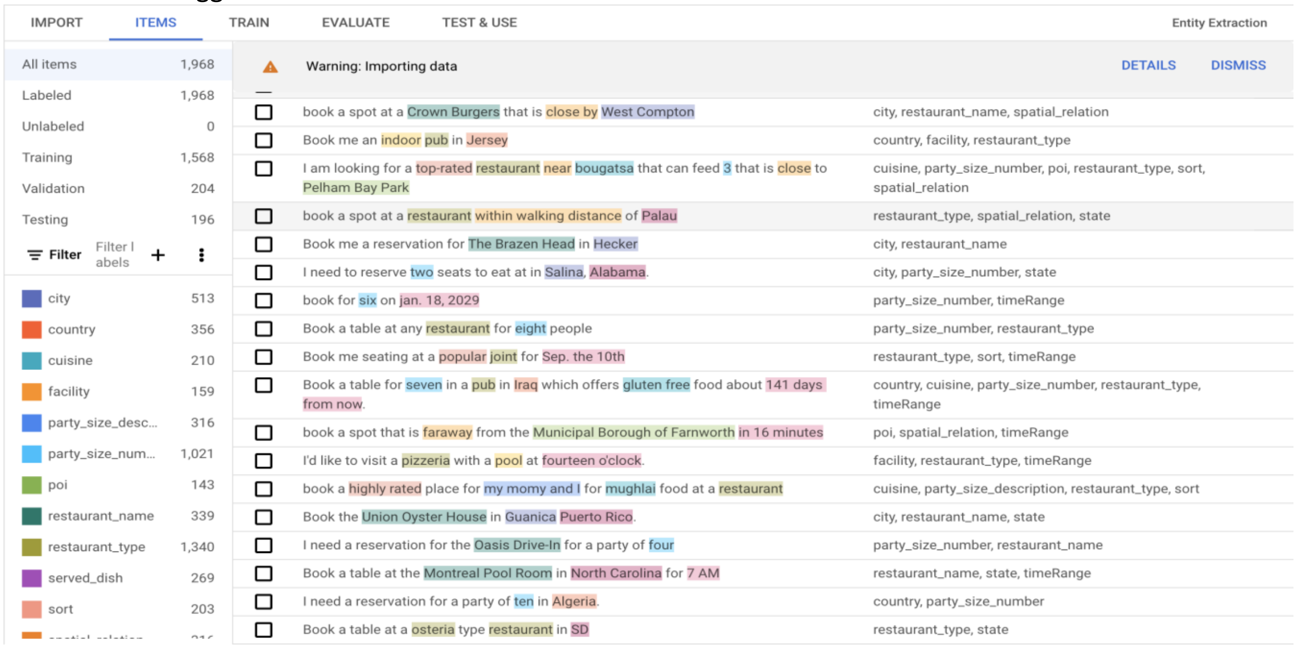
We have used Yelp Nearby REST API to find the nearest restaurants for a location (which is given by the React Native Application). We get the names of all the restaurants within 25 miles radius using this Yelp REST API provided by the Yelp. This Node-JS server hits the yelp-API with the Latitude-Longitude coordinates as parameters. The yelp-API will give out the top 5 restaurants around the location (latitude-longitude) of that user, and from these restaurants, top dishes are produced. Using this restaurants names, we get all the details of the restaurants for the further processing on cloud AutoML and sentiment analysis. These results are fetched and further display it to the user through the React Native Application.

5. Cloud AutoML:

After finding the restaurant reviews using the Yelp Nearby REST API and MongoDB cluster, we then sent the reviews of these restaurant to the Cloud AutoML API in the Google App Engine for the NER (Named Entity Recognition) training, where we will know all entities in the review. However, the prime focus is to find out the names of the dishes, we extract the Food entities in the reviews. Using this Google Cloud AutoML, we have found the food dishes present in the reviews of that restaurant. Now after knowing the food dish names, we need to find the sentiment of each review and the entity associated with it, which will be done by Cloud Natural Language API for sentiment analysis.

In the Model, the NER tasks are performed on these texts to identifies the entities (dish names) in the text (reviews). For this NER task, we have trained our model on Google AutoML to generate entities (the word span of dish names).

AutoML Food NER: This project needs extraction of the dishes from a review, known as Named Entity Recognition. As currently there is no NER task on food dataset, we created Food NER on food dataset found on Kaggle.



All items	1,968
Labeled	1,968
Unlabeled	0
Training	1,568
Validation	204
Testing	196

🟦	city	513
🟥	country	356
🟦	cuisine	210
🟧	facility	159
🟦	party_size_desc...	316
🟦	party_size_num...	1,021
🟩	poi	143
🟩	restaurant_name	339
🟩	restaurant_type	1,340
🟪	served_dish	269
🟥	sort	203

book a spot at a Crown Burgers that is close by West Compton	city, restaurant_name, spatial_relation
Book me an indoor pub in Jersey	country, facility, restaurant_type
I am looking for a top-rated restaurant near bougatsa that can feed 3 that is close to Pelham Bay Park	cuisine, party_size_number, poi, restaurant_type, sort, spatial_relation
book a spot at a restaurant within walking distance of Palau	restaurant_type, spatial_relation, state
Book me a reservation for The Brazen Head in Hecker	city, restaurant_name
I need to reserve two seats to eat at in Salina, Alabama.	city, party_size_number, state
book for six on jan. 18, 2029	party_size_number, timeRange
Book a table at any restaurant for eight people	party_size_number, restaurant_type
Book me seating at a popular joint for Sep. the 10th	restaurant_type, sort, timeRange
Book a table for seven in a pub in Iraq which offers gluten free food about 141 days from now.	country, cuisine, party_size_number, restaurant_type, timeRange
book a spot that is faraway from the Municipal Borough of Farnworth in 16 minutes	poi, spatial_relation, timeRange
I'd like to visit a pizzeria with a pool at fourteen o'clock.	facility, restaurant_type, timeRange
book a highly rated place for my momy and I for mughlai food at a restaurant	cuisine, party_size_description, restaurant_type, sort
Book the Union Oyster House in Guanica Puerto Rico.	city, restaurant_name, state
I need a reservation for the Oasis Drive-In for a party of four	party_size_number, restaurant_name
Book a table at the Montreal Pool Room in North Carolina for 7 AM	restaurant_name, state, timeRange
I need a reservation for a party of ten in Algeria.	country, party_size_number
Book a table at a osteria type restaurant in SD	restaurant_type, state

6. **Cloud Natural Language API for Sentiment Analysis:**

Here is the catch, we will get the food dish names mentioned in the review, but we need to find if there is a positive review or a negative review. We have Google Cloud service to perform this Sentiment analysis and then assigned a score to each review. This score will tell how positive or how negative it is written about a dish in that restaurant. We have calculated a cumulative score of this dish and then displayed it to the user using our front-end application (React Native Application). Please note that, with the name of the dish, we are also displaying the name of the restaurant from which this dish has been picked.

II. <u>Cloud Services utilized:</u>

The below are the Cloud services that we have utilized to build our solution:

1. **MongoDB Atlas Cluster:**

We have used MongoDB Atlas cluster to store all the restaurant reviews, there are around 2,29,000 reviews by different users. We have stored all these reviews in this MongoDB Atlas Cluster, and we provided a connection to it for the NodeJS App engine to access all the reviews. These reviews are then further utilized for Named Entity Recognition and Sentiment Analysis. In out application, MongoDB served the purpose of storing our data, and this MongoDB Atlas Cluster is one of a Cloud Service we used.

2. **Google App Engine (NodeJS):**

We have used the PaaS Resources from the Google App Engine to build our application. We have installed a NodeJS server in this Google App Engine (GAE), made it a NodeJS App Engine. This is a Cloud service we used to for all backend execution and intermediate calls between Cloud AutoML, Cloud Natural Language APIs.

3. **Cloud AutoML (for NER):**

This is a Cloud service we have utilized in our application for the Named Entity Recognition Task. This Cloud AutoML will take the reviews and find the entities within the review. The food dishes in the review are recognized using this Cloud AutoML around which we calculate sentiment analysis.

4. **Cloud Natural Language API for Sentiment Analysis:**

We have used a cloud service called Cloud Natural Language API in our application to find the score of the review. The score of each dish is calculated and this score will determine if this dish is a good dish or bad dish, also will tell by how much extent the dish is good/bad, by displaying its score.

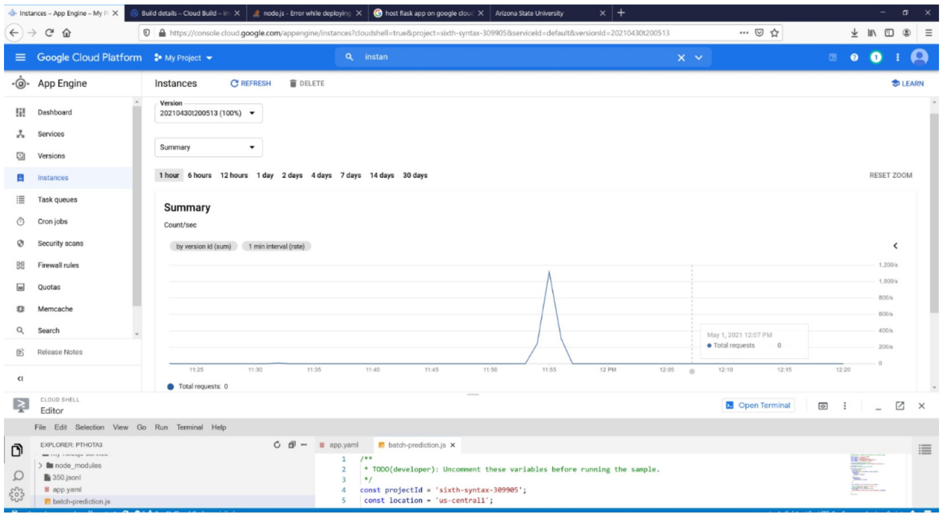
III. <u>Role of Google App Engine:</u>

We have written our code that runs on Nodejs server and it needs to be deployed on Nodejs runtime. Google App Engine (GAE) which is a PaaS, provides a Nodejs environment for the application. Entire backend of the project is deployed on this Google App Engine which makes intermediate calls to other cloud services - Cloud AutoML (for NER), Cloud Natural Language API, Yelp Nearby API. GAE also provides an autoscaling feature, which auto-scales number of instances based on number of requests and the time taken for processing the reviews of the restaurant.

IV. Autoscaling:

To increase the performance of the application, we are auto-scaling it as the number of reviews that the Cloud AutoML has to process increase and also, as the number of user requests increase. Based on these factors our application is scaling in and scaling out automatically. This is improving the time the application is taking to complete the task given by one or many users. We have observed that as the number of requests and reviews the system has to process increases (either by one or more users), the number of instances are increasing and when the requests to process are completed, the instances terminate themselves, proving that the application is Auto-scaling.

The bottleneck we observed is that, since our application runs Named Entity Recognition and Sentiment Analysis on each review, the time required by Cloud AutoML to process all these reviews take a lot of time. Based on the review count, Cloud AutoML scales in and out automatically and gives out very good results in comparably very less time, therefore Auto-scaling has improved the performance by a large extent.



```
/**
 * TODO(developer): Uncomment these variables before running the sample.
 */
const projectId = 'sixth-syntax-309905';
 const location = 'us-central1';
```

V. How our application solves the problem in section 1:

As the problem mention in the section 1 is about having the names of the best dishes around the restaurants with the location of the user, and through our front-end application we are reading the Long-Lat coordinates through the Home Page. We are getting the Top Dish names in that region by getting all the restaurant information in that region around 25 miles using Yelp API, and then after getting the reviews in these restaurants, we send it to our model for Named Entity Recognition and sentiment analysis. We get the scored result for every popular dish (above the threshold) and display it in our second page of the application. In the third page of the application, we further use the results obtained in the second page to display the names of the restaurants that has these most popular dishes available.

VI. Why our application is better than State-of-the-art:

As all the applications that we use now only display the reviews and display the average ratings it received by the user, as we have discussed in the Section II why it is not an effective way to judge a restaurant by the overall ratings. Every popular application we know will just display the reviews but does not understand the sentiment in those reviews, just viewing these reviews and rating given by the user will not be effective, so we need further information to judge a restaurant. Also, due to many bad dishes the rating of the restaurant will be low, and that one best dish in that restaurant will never come into light because the present applications show the overall ratings, and the user will not consider the restaurant because of its low rating obtained by the bad dishes. There will be many such local small-scale businesses effected for this reason too.

Now, we have designed an application that will give out the best Dishes by NER and Sentiment Analysis on the reviews the restaurant received. When we know the Top Dishes around that region, we will know what dish to taste and what not to taste even, which will be very handy even when we are going to completely new place. This type of approach has not been used before and we strongly feel this will be an effective method that should be adopted, to identify best dishes, which will help both the users and the restaurants.

4. <u>Testing and evaluation</u>

After the deployment we have tested the functionalities of the application. Since the application is based on location of the user, we have tested the application on multiple locations and in the location where there are less restaurant reviews in our dataset.

Current GPS: We have tested our application in the current location i.e., Tempe. Since the dataset contained lot of reviews for Tempe restaurants, NLP model has taken time to process the reviews and we have observed a delay for the results to appear on the user screen.

Mock GPS: To test the application's performance on various locations, we have used Fake GPS application which mocks the current location and we observed interesting results. As we moved away from Phoenix (dataset contains only Phoenix's restaurants), we observed the latency was very less and results are not that accurate. The reason was since the reviews to be processed are very less, time taken to show the results is short and results are based on only those reviews, thereby decreasing the correctness of the result.

Autoscaling: We have tested the application with multiple user requests. We used Apache Benchmark for load testing. We have tested the application with 50000 requests with 1000 concurrent request. The instances are scaled up to 10 instances and it took about 10 minutes to process the request and scale down the instances.

5. <u>Code</u>

Server.js: The whole code is deployed on NodeJS sever. This file is the start script on deploying. This file contains the code for integrating all the components. It receives the latitude and longitude co-ordinates as the request. This is then sent to Yelp Nearby REST API as request and receives the restaurant ids as response. These IDs are then sent as request to MongoDB to retrieve the reviews and then sent to AutoML food NER API. The food entities are then sent to Cloud NLP for sentiment analysis and then scored. These results are evaluated and ranked on specific metrics which is sent as Json response to the end user.

testRun.jsonl: This file a json Line format which is an intermediate temporary file generated by server.js after Entity recognition by AutoML API and Cloud NLP sentiment analysis for every review.

counter_score.py: This code is to evaluate the scores of all the reviews and rank based on a metric defined. This is a child process spawned by Server.js file. It reads each line from testRun.jsonl file and maps it to a counter and order them by score which gives the best dishes in that region. To find best dishes in an individual restaurant it applies the same metric but grouped by restaurant reviews. The result is then sent as a json object which is a combination of two results.

Ner-json-format.py: To train the food NER on AutoML, the dataset needs to be in jsonl format. This code loads the complete dataset and formats the dataset as per AutoML requirements and writes it as jsonl file. This file can then be used for training the model.

Upload_reviews_mongo.py: This code is to upload the yelp reviews into MongoDB Atlas cluster. It uploads review as document into yelp-reviews collection defined in the documentDB.

Sentiment.js: It takes a single review as text and makes a call to Cloud NLP API for Sentiment Analysis and receives the sentiment score of the review.

Package.json: Contains all the dependencies and start scripts used to download the node modules and start the application.

App.yaml: This file is to mention the runtime of the app engine which is Nodejs14. Here we can set the number of instances limit for autoscaling.

Batch-prediction.js: This code is to take a batch of reviews and send it to AutoML NER - API for entity extraction and write the output in a jsonl file.

React Native: This is for build the front of the Application. It sends latitude and longitude as request to NodeJS App Engine and receives the Json response.

a) HomeScreen.js: It contains the code to retrieve the current location of the user in terms of latitude and longitude. On clicking the button, it sends the request with latitude and longitude as query parameters.

b) RestaurantBestDish.js: When query is solved it receives the json response. This code parse the response to display only the top dishes among all the restaurants and also shows the overall score of the dish.

c) TopDishAmongRestuarants.js: Upon on clicking the filter button it moves to the next screen. This code to parse the json response to view top dishes in individual restaurant. It also displays the Name of the restaurants, Address and City in React native badges,

Steps to run the Code:

Preprocessing Folder:

We need to train the food NER on Google Cloud AutoML, for this our dataset should be in jsonl format. Run NER-json-format.py to convert reviews into jsonl format. The code required is found in this Preprocessing folder. Run this code on the reviews.

Database Folder:

This folder has code required to upload the dataset to MongoDB Atlas Cluster. Run upload_reviews_mongo.py to upload the reviews (2,29,000 entries) to MongoDB document database.

Front-End Module:

Download React Native code from front-end folder and run npm install command. Then run the line “npm install --global expo-cli” in that folder through Command Prompt. To start deploying the application run “expo start” command, which will generate a React Native App on your connected phone.

Backend Module:

Setup NodeJS environment on Google App Engine, and upload the files app.yaml, batch-prediction.js, counter-score.py, package.json, sentiment.js, server.js and testRun.jsonl file to the GAE.

Deployment:

Your application is ready for deployment. After completing the above steps, run “gcloud app deploy” in the Google Command Shell in the folder where the backend module is hosted.

View the results now on Expo Go application, which is installed in your mobile device, to see the best dishes in restaurants around your location.

6. <u>Conclusions</u>

Summary of what we have achieved and learnt:

Through this project we understood the working of a PaaS application, Google App Engine. We got a deep understanding of Cloud services like Cloud AutoML and Cloud Natural Language API, and understood the effectiveness of these resources, while also knowing the advantages and disadvantages. We also understood how Google App Engine (GAE) is used to make robust applications as it allows to implement backend services through its runtime and deploy it on cloud.

For choosing the databases for our application, we have tried many cloud databases like Google Firebase, Google Cloud SQL, and many to connect to our application. In this process of finding a perfect database of our needs, we understood the limitations of each database, then chose to go with MongoDB Atlas Cluster which also uses Google Cluster.

To increase the reachability of our application, we need this application to make it handy and available across all the platforms, we then aimed to make it a mobile application as mobile phones will be handy for the user and the GPS of the system will provide the Lat-Long of the user. We looked up to make a mobile application in both Android and iOS, and we also wanted to provide web access to our application. We then came across React Native, and we have created a UI using this React Native and Expo Go. We got a deep understanding of React Native, and the possible applications we can build/scope using React Native.

We got a clear idea on how to integrate the above mentioned front-end, back-end and database modules to create a completely loosely coupled system, that can also Auto-scale based on the requests it receives. It was an ultimate fun working on an application that we initially planned of and getting it live by implementing it as a Full-Stack application using Cloud resources across different platforms.

Improvement in the Future:

Below are a lit of features that can be implemented to improve this application.

1. Our application only works in the region of Phoenix as the database is loaded with all the reviews of all the restaurants in the Phoenix region. We can expand this and make this application work in all the other regions across the United States by improving the dataset.
2. Our application currently uses the reviews present in the Yelp; however, we can also expand it and make our application work on reviews that are posted on other websites like Google Reviews, Trip Advisors etc.

3. Currently we have utilized the free credit of 300 USD given to us by the Google Cloud Platform, we can improve the performance and reduce the latency by expanding our Cloud resources.
4. We can further increase the scope of this application by implementing it in other countries like in Canada, India, China etc., by sending the reviews collected from the popular websites in the respective countries as our application is Crowd sourcing application.

7. Individual contributions

Team Member: Venkat Krishna Sai Kowkuntla

Contribution to the Project: My contribution towards the project comes under implementing various different components of the application. I have my part working on the project's Frontend, Backend, Database and Testing. I played a key role in Integration of all key components and tested the functionality on the React-Native Mobile Application.

Design and Implementation:

Front-end: I am the key developer to implement the front-end part of the application entirely using React-Native Mobile Application. I chose React-Native to implement the application as our target users were mobile phone users and the app can run on variety of devices like iOS, Android, Desktop. I implemented different screens of the User Interface using loosely coupled architecture and parsed the response json coming from the service hosted on Google App Engine to show the results on the UI.

Backend: I also played a key role in the team as backend developer to implement the services hosted on Google App Engine. I researched about the different APIs that are available online to get the customer reviews and found out Yelp serves our purpose which also comes with a proper documentation. I queried the Yelp API with the current location to find out the restaurants that are around 25 miles. We will be storing these restaurant ids to further query the MongoDB review data.

Database: I also played a key role in creating the MongoDB Atlas instance to store the review data of Phoenix Metropolitan Region. I also tested the MongoDB collection on different types of queries on restaurant name, id etc. I also wrote a service on Google App Engine to query MongoDB review collection based on the restaurant ids retrieved from the Yelp API. We will be using the retrieved reviews to rate best dish out of the reviews for each restaurant and to rate best dish out of reviews in a particular location.

Code Integration: I played a key role in integrating the different components of the code. We implemented all the different modules of the application independently and tested their working when they are deployed alone on the Google App Engine. I started the integration by calling the backend service hosted on Google App Engine in the front end React Native Mobile Application to retrieve different results on best dishes in a particular location. Similarly, I integrated the code to call the MongoDB review collection based on restaurant ids retrieved from the Yelp API. I further merged the code of calling the AutoML Model on each review to identify the dishes. I also merged the code for analyzing sentiment of a review and score the dishes in that review.

Testing: After the code Integration Phase, I thoroughly tested each workflow to identify any issues and solved the issues one after the other. I published the mobile application on Expo Live Server so that it is available for testing to all of our teammates. I also tested different scenarios of giving fake location to our application and checking the results are changing for each location. I also tested the application for different batch inputs given to the Apache Workbench for autoscaling.

Team Member: Pavan Kalyan Reddy Thota

My contribution to the projects involves in design and implementation of various components of the backend. I played a major role in designing the architecture of the project. I have worked on developing AutoML NER on food, Ranking and defining the food and Restaurant metrics, Deployment, Testing and Evaluation.

Design: In the design of the project, we decided to implement Nodejs server on Google App Engine of GCP and used DocumentDB on MongoDB Atlas cluster. I designed the flow of request and responses which starts from the user on front end and ends with response to the user. The front-end user sends latitude and longitude to Nodejs server which makes the request for Nearby Restaurants of YELP API. This response is then sent to MongoDB to fetch reviews. These relevant reviews are then sent to AutoML food NER and for sentiment analysis followed by evaluation and ranking the dishes. The result is sent as JSON response to the front-end user.

Implementation:

AutoML Food NER: For the project we need to extract the dishes from a review, and this is called Named Entity Recognition. Upon research I found that there is no existing model to do NER task on food dataset. We decided to create Food NER and found a food dataset on Kaggle. I have preprocessed the dataset and wrote the python code to format the dataset as per the google AutoML NER instructions and then trained it and achieved an accuracy of 94.51% . We then created a REST API which takes text as payload and responses dishes in it. This API is then used for the next components of the project.

Restaurant metrics: I have defined and implemented the metrics for ranking the dishes individually and cumulatively. This component reads a jsonl file from AutoML and Sentiment analysis component. I wrote the python code to rank the dishes based on the frequency and sentiment score. For this we set a threshold of 0.5(positive score) and scored them cumulatively and individually in a restaurant.

Deployment: We have deployed our final application on Google App Engine on Nodejs server runtime. To deploy the application, we configured the run time on App.yaml file and set it to nodejs14. It is deployed every time by running ‘gcloud app deploy’ command. I found out the issues in the deployment and figured out the component responsible for it and redeployed multiple times based on the changes in the code.

Testing and Evaluation: The application we created gives the results based on the location. So, I tested the application on various location by using Mock GPS which fakes the current location of the user. We noticed that the response time of the application is different in various location. This is because the reviews of restaurants for few locations are very less or none on our MongoDB database. This eventually decreases the processing time of reviews by Cloud NLP API and the response time is very short.

Autoscaling Testing: We tested the application based on number of requests. To test the Auto Scaling features, I have setup Apache Bench for Load testing and tested with 50000 requests. The app engine scaled to 20 instances and took about 10 minutes to scaled down.

Team Member: Sai Teja Vishal Jangala

Contribution to the Project: My contribution to the project involves in Design, Implementation and Testing of the application. In design, my part involved in designing the Database on MongoDB, using MongoDB Atlas Cluster. In the implementation, my work involves in setting up Yelp Nearby REST API, configuring the database MongoDB Atlas Cluster, performing sentiment analysis, and Deploying the final application on Google App Engine. In testing and evaluation of the entire application, making the application resilient by testing it on different conditions. Each of them is explained clearly below:

<u>Design:</u>
In this project, I have designed the structure of the Database and configured its flow. This is a major component as it is a source of reviews. Many cloud service databases have been tested in the process of choosing a correct database, few of them are Google Firebase, Google Cloud SQL etc., They have failed in few conditions making me to choose MongoDB Atlas Cluster, to store 2,29,000 review entries.

<u>Implementation:</u>
Yelp Nearby REST API: I have contributed to utilizing this Yelp REST API. The front-end module (React Native) will read the Lat-Long values of the user and sends it to the back-end App Engine (server.js), where the Yelp API gets the restaurants within 25 miles. I have made sure that this critical segment of the application works relentlessly.

MongoDB Reviews Hosting: I have played a major role in setting up the Database of this application to maintain a perfect continuity with the application. I have loaded 2,29,000 reviews into the MongoDB Atlas Cluster by converting it from CSV to JSON format to create a Document Database. Created a flow, to store and retrieve the reviews required for model training on Cloud AutoML for NER to figure out food entities and then for sentiment analysis on those food entities.

API for Sentiment Analysis on the reviews: This is one of the critical sections of the application, as the entire idea on the implementation of this module. The reviews from database are given to payload of the request of the Cloud Natural Language API to perform the sentiment analysis. This module implementation will give out the magnitude of the score associated with each review and telling if the review is positive or negative. Following to these results (scores), I summed up to find the cumulative score, which will be later displayed on our front-end module.

Google App Engine (GAE) Deployment: After the application is completed, I have contributed to deployment of the application on the Google App Engine which is on a NodeJS runtime by setting up the configurations in app.yaml. The deployment is carried using "gcloud app deploy" command on google cloud shell. Several issues occurred during the deployment, which were solved through extensive debugging. I have learnt more about the Google App Engine in this process.

<u>Testing and Evaluation:</u> I have contributed to the testing of the application designed and implemented here to check all the functionalities, and by evaluating and monitoring the auto-scaling feature by giving various batch sizes to the Apache Workbench. I contributed to test the application on various coordinates, by setting up the Mock GPS locations and noticed the change in response times to make the application resilient to various conditions. I used this Mock GPS to test in the locations where there are many restaurants and in locations where there are less restaurants to evaluate its performance.

8. References

[1] https://www.yelp.com/dataset

[2] https://www.kaggle.com/swapnilpote/restaurant-chatbot-dataset-intent-entity

[3] https://www.yelp.com/developers/documentation/v3/get_started

[4] https://cloud.google.com/natural-language/automl/docs/models

[5] https://cloud.google.com/natural-language/docs/analyzing-sentiment

[6] https://medium.com/rakuten-rapidapi/top-10-best-food-and-recipe-apis-yelp-zomato-untappd-and-more-2bf712a032c2

[7] https://cloud.google.com/natural-language/automl/docs

[8] https://academic.oup.com/database/article/doi/10.1093/database/baz121/5611291

[9] https://www.scitepress.org/Papers/2019/76863/76863.pdf

[10] https://arxiv.org/ftp/arxiv/papers/1609/1609.01933.pdf

[11] https://reactnative.dev/docs/navigation