Github:
https://github.com/CorneliusOsei2/AiPrac
Video:
https://drive.google.com/file/d/1UMLdAQJT4wXyFrHv9ddgB_1vcGSceEUq/view?usp=sharing

Top-N Restaurant Recommendations

Emily Robinson (err78), Shuyan Nie (sn429), Cornelius Boateng (cob32)

AI keywords: Sentiment Analysis, TensorFlow, PCA, Neural Networks

Application setting: Top-N restaurant recommendations

Plan

Comparison to Status Report

P1 - P4 come directly from the status report.

P1: *We will create an AI that provides good place recommendations. To accomplish this, we need to tackle two problems: getting places that fit the constraints of a user's request and assigning values to place recommendations to determine how "good" they are. Afterward, the AI will provide the top recommendations.*

Throughout the final phase of our project, we successfully implemented the intended functionality. Our project utilizes AI algorithms to rank restaurant recommendations based on data obtained from the Yelp API. The AI-powered ranking system enables us to enhance the quality and relevance of the recommendations provided. Additionally, we incorporated Principal Component Analysis (PCA) to generate weights for the review and rating scores, as described in subsequent sections of this report. This weighting mechanism allows us to effectively combine the two scores and derive a comprehensive recommendation ranking. Ultimately, our project offers an improved user experience by enabling users to specify the desired number of recommendations they wish to receive. This level of customization empowers users to obtain tailored suggestions that meet their individual preferences and needs.

P2: *Problem 1 involves using a user's request, such as "I want spicy food" to query the Google API for a list of appropriate restaurants.*

We have made substantial progress in implementing the planned functionality, with one notable difference: we utilized the Yelp API instead of the Google API, as mentioned in a blue comment in the status report. The decision to switch APIs was primarily due to the ease of obtaining a free API key for Yelp.

Consistent with our status report, our interface allows users to make queries such as "I want spicy food" when prompted. Moreover, users have the option to specify location information for more targeted recommendations. For a visual representation of our interface, please refer to Figure 1.

To address Problem 1, we focused our efforts on the query.py file. The core component of this file is the query_api function, which employs several helper functions to retrieve results by passing the user's query to the Yelp API. Subsequently, the function converts the obtained results into a compatible format for seamless integration with our other functions. Finally, the write_to_json function is invoked to store all the relevant information in the all_restaurants.json file. For an example of the data format used, please consult Figure 3.

P3: *Problem 2 uses our AI. First, the AI randomly samples N (determined by the programmer) of the Google recommendations, then calculates a value for each. The value derives from the average semantic value of reviews (e.g +1 = positive, 0 = neutral, -1 negative It's just +1 and 0) and the normalized rating.*

For Problem 2, we opted to create functions to evaluate recommendations for N restaurants in the all_restaurants.json file. To accomplish this, we designed a Restaurant class in our restaurant.py file, which encompasses these evaluation functions. To transform each recommendation into a Restaurant object, we utilized the init_restaurants function in main.py.

One of the functions within the Restaurant class, __set_rating_score, normalizes the rating score and calculates the average, in line with our initial plan as outlined in the status report. Our intention was to ensure that ratings fall within the range of 0 to 1 and are on a consistent scale.

The score methods, namely __set_review_score and __set_final_score, depend on functions defined in the model.py file, which rely on a trained neural network model. This model is created by invoking the train function in restaurant.py, which, in turn, calls the make_model function responsible for the actual training process in model.py. The trained model takes a review as input and returns the probability of that review

being positive, ranging between 0 and 1, as opposed to just +1 and 0 as mentioned in the previous description.

Within the Restaurant class, the __set_review_score function calls eval_reviews, a function defined in our model.py file. This function calculates the overall semantic value of all the reviews using the trained model.

The __set_final_score function utilizes the eval_weights function, which employs PCA to determine the weights for the review score and rating score. These weights are then used to combine the review and rating scores, resulting in the calculation of the final score for a restaurant. By incorporating weights, we aimed to automatically account for the relative importance of each feature during the final score calculation. Please refer to Figure 5 for a visual representation.

All of the scoring functions mentioned above are invoked within the set_scores function, which resides within the Restaurant class.

P4: *Finally, the AI outputs the M highest valued recommendations. It's not the AI itself that does this but our script. The AI is just used in our score calculation.. M comes from another user query asking how many suggestions they want, which will only be valid if less than N. M will not come from the user, but the programmer.*

We have implemented a Recommendations class in restaurant.py, which includes a function called "set_top_N" that generates the top-N (or rather, top-M based on how we defined N and M) results. It's important to note that the AI is not involved in this particular calculation. However, there has been a change from the status report: the value of M will now be determined by the user instead of being predefined by the programmer. On the other hand, N is still set by the programmer.

According to our original concept, N represents the desired number of restaurants to be retrieved from the all_restaurants.json file. This parameter is primarily used in the "make_restaurants" function in main.py to create N Restaurant objects based on the data in the all_restaurants.json file. We incorporated this level of customization mainly for testing purposes, allowing us to work with a manageable number of restaurants and verify the functionality of our methods.

Regarding M, we refer to it as N within the "set_top_N" function. It still denotes the number of recommendations to be output, but its value is determined by the user when prompted by the interface (see Figure 1).

Additions

In addition to providing recommendations, we have extended our functionality beyond what was mentioned in the status report. Our plan now involves prompting the user to choose a recommendation, after which the program will return the corresponding location. Furthermore, we have programmed the interface to ask the user if they would like to select another recommendation. If the user responds affirmatively, it will repeat the process, and if not, it will inquire whether they want to rerun the entire program. Please refer to Figure 2 for a visual representation.

We have also made an addition to our testing approach. As mentioned earlier, we have incorporated parameters for specifying the number of restaurants to create, as well as a display parameter to showcase the results of calculating each score in the "make_restaurants" function. This level of customization and the ability to display detailed information for each restaurant has proven invaluable during the testing phase of our scoring methods.

It allows us to select a subset of restaurants and observe the output without having to iterate through the entire dataset. For instance, in Figure 4, we called the "make_restaurants" function with the parameters "n=1" and "display=True," resulting in the creation of a single restaurant and the printing of each of its scores. To thoroughly test our "set_top_N" function, each team member acted as a user, ensuring that the recommendations were generated correctly.

Plan Conclusions

In summary, we have accomplished our planned objectives successfully. We have successfully implemented all the score functions, developed the neural network model, and created the PCA weight function, as outlined in our initial plan stated in the status report.

Moreover, although we initially stated in the status report that we would not provide the location of recommended restaurants to users, we were able to allocate sufficient time to address this feature as well. This enhancement allows users to access not only the recommendations but also the corresponding locations of the recommended restaurants.

Furthermore, we conducted comprehensive testing of all our functions, with team members actively participating as users. This testing approach ensured that our functions were thoroughly validated and yielded reliable results.

Important Figures

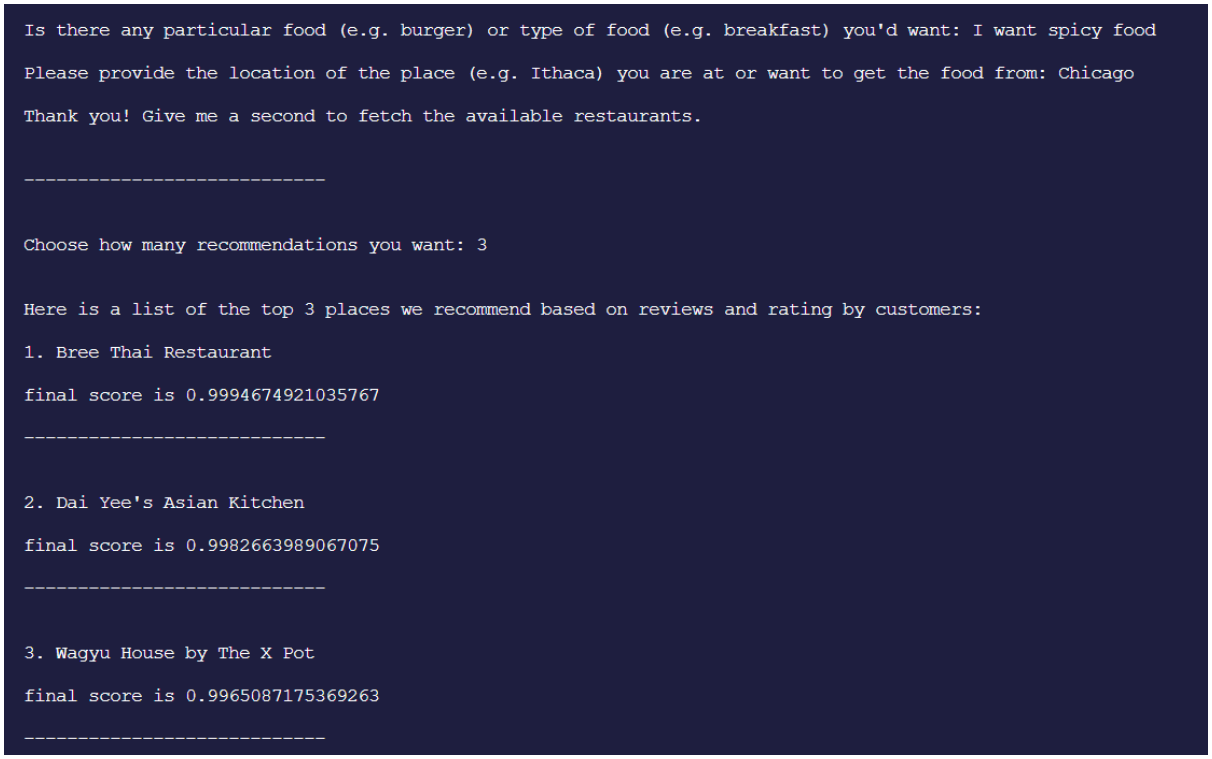

Figure 1: *Users can input their restaurant or food preferences, followed by their desired location. The program then prompts users to specify the number of recommendations they want. Using this information, the program generates the top-N recommendations from Yelp, sorted in descending order based on their final scores.*

```
Give the list number of the recommendation you want (e.g. 1, 2, 3, etc.): 1

Here's the location of Bree Thai Restaurant:
5306 S Central Ave, Chicago, IL 60638

Do you want another restaurant's location? say (y/n): y

Give the list number of the recommendation you want (e.g. 1, 2, 3, etc.): 2

Here's the location of Dai Yee's Asian Kitchen:
4131 N Rockwell St, Chicago, IL 60618

Do you want another restaurant's location? say (y/n): n

Do you want to do this again? say (y/n): y

----------------------------



Is there any particular food (e.g. burger) or type of food (e.g. breakfast) you'd want:
```

Figure 2: *After receiving recommendations, users can choose one and get its location. Then users can choose another restaurant. If they decline, they have the option to re-run the entire program.*

all_restaurants.json

```
You, yesterday | 1 author (You)
{
    "data": {
        "Wildberry Pancakes and Cafe": {
            "location": "130 E Randolph St, Chicago, IL 60601",
            "reviews_and_ratings": [
                {
                    "Rating": 5,
                    "Review": "Food was wonderful like always anytime I can I will continue
                    coming to get an awesome breakfast"      You, yesterday • Finished making
                },
                {
                    "Review": "Food was wonderful like always anytime I can I will continue
                    coming to get an awesome breakfast",
                    "Rating": 5
                },
```

Figure 3: *This shows the format of all_restaurants.json. The file gets updated each time new data is fetched from Yelp.*

```
SCORES FOR Wake 'n Bacon


review score is 0.5379826426506042
 rating score is 0.25
 final score is 0.28254491878129256
```

Figure 4: *This figure prints out the result of calling* ***set_scores*** *for a particular restaurant.*

```
PCA weights
 rating weight: 0.9983998980388897, review weight: 0.056547710793054
```

Figure 5: *This displays weights that were calculated for a particular restaurant.*

Core of Project

Model Creation and Training

We firstly trained and tested our Tensorflow model on reviews from Kaggle to output the probability that a review/comment was positive/good.

Data Querying

We then query real-world data from Yelp based on a user's input. The input has parameters "term" which could be a specific food or type of food (like breakfast) and "location" which specifies where the user is or wants the food.

The data obtained from the API is then parsed and output to a JSON for processing for the ML model.

Testing and Output

The restaurant data are fed into the model, whose prediction together with other computations are used to develop a recommendation system. The system is based on weighted reviews and ratings scores.

The top N (given by the user) recommendations/restaurants are then output to the user.

By extension, the user can restart the whole process or request more information such as location, etc.

Evaluation

We assessed the precision of our AI model by comparing its recommended list of restaurants to a customer's actual favorite restaurants. In this case we were the users testing the program and asked two other students not in this course to participate.

On average when a customer's input of "Chinese" and a request for 10 recommendations in Ithaca, our AI model produces a list of recommendations with corresponding final scores. The actual favorite list of the customer included some overlapping restaurants with the recommendations. Specifically, there was a 70%

overlap on average between the actual set of restaurants the customer likes and the ones we recommended.

In another test, when the customer input was "Steak" in Rochester and a request for 5 recommendations, our AI model provided recommendations with corresponding scores. There was an 80% overlap on average between the actual favorite list of the customer and our recommendations.

Lastly, when the customer input was "Fried Chicken" in Syracuse, our AI model generated a list of recommendations. There was a 60% overlap on average between the actual set of favorite restaurants and our recommendations.

Based on these test results, our AI model successfully recommended the correct number of restaurants in the specified locations. Additionally, it consistently achieved a relatively high precision rate over 50%. While it is challenging to satisfy everyone's unique restaurant preferences, our AI model performed exceptionally well in providing accurate recommendations.

References

Supplemental Materials

- TensorFlow Youtube video
- Twitter analysis using TensorFlow
- Twitter sentiment analysis with TensorFlow
- TensorFlow Sentiment Analysis Notebook

Data resources

- YelpAPI
- Dataset of Reviews