University of California
Berkeley
Data-X Fall 2017

Eat Easy Final Report

Dhruv Madaan
Jennifer Chen
Jessica Kam
Pouriya Bagheri

Contents

Overview of Project

Eat Easy is an application which provides customized menu suggestions to a user. It takes in user input, such as the restaurant name and personal food preferences, and analyzes Yelp reviews to match the user with the best dishes at the selected restaurant. This is extremely useful for anyone who is deciding what to order from a new restaurant or is looking to try a new dish at a favorite spot. A common problem that diners face is when the restaurant servers provide unhelpful suggestions by recommending the entire menu. We aim to aid and streamline the decision-making process by automatically generating a Eat-Easy ranking so users no longer have to read through pages of reviews on Yelp to collect recommendations. We plan to use reviews from the Yelp Data Challenge 2017 dataset, along with menu items from Allmenus.com, to perform sentiment analysis and develop rankings for restaurant dishes in the city of Phoenix, AZ.

Problem Definition

Currently there is not an efficient way to determine what is good to order at a restaurant without reading through numerous reviews. Furthermore, not all restaurants upload their menus on Yelp, so Yelp can not provide comprehensive suggestions on what to get. We aim to solve these two problems.

Approach to solution

We utilized the following tools, some of which we learnt in Data-X and others which were self taught:

1. **MySQL:**
   The Yelp Dataset in raw form was 6 GB. We had trouble loading that dataset into a pandas dataframe and such had to explore other options. Upon research, we discovered SQL was optimized for large datasets and would be useful in performing initial preprocessing and cleaning.
2. **Python:**
   Python was used to develop our code and conduct the various Natural Language Processing algorithms. The wide availability of resources for libraries and algorithms made this preferred platform for development.
3. **Flask:**
   Flask is used to search our database for restaurant menu items, ratings from our recommender, and other relevant information scraped from AllMenus. Users are also able to register and login to add menu items to their dashboard. They can additionally add their own ratings and comments, which are also recorded in our database.

Data Cleaning

Most of our time was spent cleaning the dataset. We had to first understand how the data was structured and then select the appropriate features that would be useful for our project, which are highlighted in the green boxes in Figure 1.

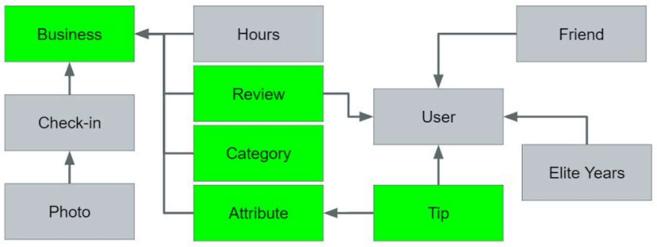

Figure 1: Yelp dataset schema. The green boxes highlight which database tables were useful for our project.

Some of our learnings/challenges from cleaning the data:

1. Contrary to our belief, the Yelp dataset included many businesses that were non-food related.
2. Not all cities had an equal amount of food establishments. In fact, many major cities like San Francisco, New York, and Los Angeles only had a handful of restaurants. This meant that the dataset from Yelp was not a comprehensive database. As a result, we focused on the most populated region from the dataset (Phoenix, AZ) and used the Phoenix reviews to train our model and develop the rankings.

Web Scraping

In order to rank a menu item,, we had to first populate the menus for the specific restaurants. Since this information was not given in the Yelp dataset, we had to scrape the web for menus. We decided to use Allmenus.com to scrape menu information for each of the food establishments in Phoenix, AZ that were mentioned in the Yelp database.

Combining the menus and reviews

This step was the most challenging part of our project. How do we relate a menu item to a review, especially if the item is not explicitly mentioned in the reviews? We decided to run a part of our code with a set of string comparisons. For our matching strategy, we checked if there was a mention of the menu item or a partial mention of the item in the review.

```
def review_contains_item(item_name, review, weight):
    if item_name == None:
        return -1
    if type(item_name) != type("hello"):
        return -1
    if item_name.lower() in review.lower():
        return weight
    for word in item_name.split("
        if word.lower() in review.lower():
            return 1
    return -1
```

An example is the Pulled Pork Sandwich. The majority of reviews refer to other Pulled Pork dishes and only one reviewer specifically mentioned the Pulled Pork Sandwich.

Score with exact match weighting:

Menu item: Pulled Pork Sandwich , Eat Easy Score: 89.3324739564522

Score without exact match weighting:

Menu item: Pulled Pork Sandwich , Eat Easy Score: 85.11491066082213

Developing the Ranking System

In order to rank the the various dishes in each restaurant, we developed the following ranking score, which would amplify the score if the dish appeared exactly in the review as shown in the previous section.

$$\textbf{\textit{Ranking Score of Dish}} = \sum((5 \times \textbf{\textit{Sentiment Score}} \times \boldsymbol{W_1}) + (\textbf{\textit{Star Rating of Review}} \times \boldsymbol{W_2})) \times \boldsymbol{M}$$

Where :

Sentiment Score is output from TextBlob between -1(negative) and 1(positive).

W_1 is the weight of the sentiment score

W_2 is the weight of the star rating given by the reviewer

M equals 1.25 if there is an exact match for dish in the review. Equals 1 if there isn't an exact match but part of the word exists. Equals 0 if there is no match.

Future Steps

We want to be able to incorporate the user feedback to optimize the weights in our ranking score along with the multiplication factor. This could be implemented through iterative feedback tuning of parameters.

We would also like to run additional tests to improve the accuracy of our results and hopefully be able to publish our results for the Yelp Data Challenge.

Link to github repo: https://github.com/jenchen/eat-easy

References

[1] https://github.com/jenchen/eat-easy