Capstone Project 2

Sentiment Analysis of Yelp Restaurant Reviews

Shirley Zhu

Problem Statement

When people go to a new restaurant, they usually don't know what to order. Yelp reviews can usually provide some ideas about what they might like, but there are often too many reviews and it takes a long time to read the reviews and make a decision. This project leverages machine learning to figure out what everyone prefers to eat based on the restaurant"s reviews.

My client will be Yelp and the users of this feature will be the restaurants that list their business on Yelp and the customers who go to eat in these restaurants. Yelp can build this feature because users would love using Yelp to figure out what to order. Yelp can help restaurants to roll out its "popular dishes feature" to help the customers in deciding what to order from the restaurant with whom they aren't familiar. Good rated restaurants can have bad dishes and ok restaurants can have really good dishes. The restaurants can use this information to advertise their popular dishes and improve their not so good dishes, so the customers will be more satisfied after eating there. This will boost the business of the restaurants. Yelp can charge the restaurants for this feature, and Yelp takeout business will grow too.

The first part of this project is the prediction of review stars using Natural Language Processing. The second part is the recommendation of highly rated food items for a restaurant.

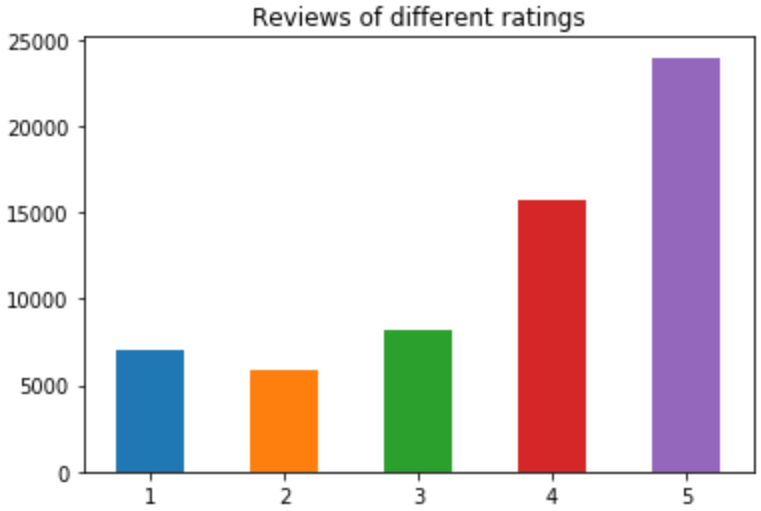
The data source is from the Yelp Open Dataset https://www.yelp.com/dataset. The dataset is in the json format. Business.json and review.json were used in this project.

Data Wrangling

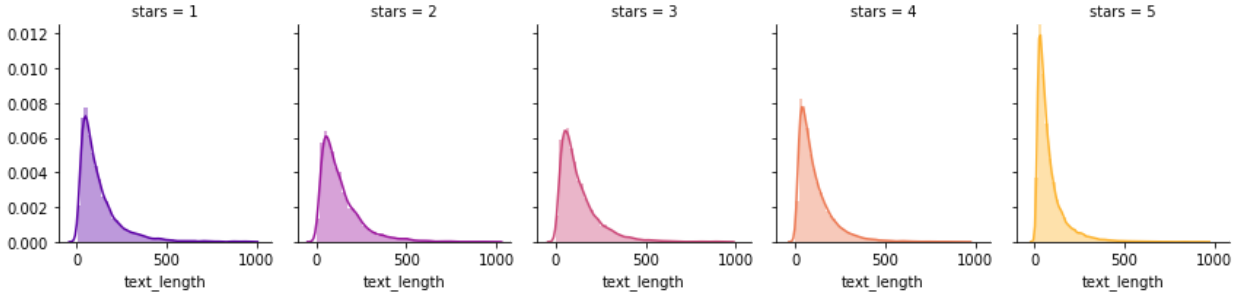
- Read the json file into a pandas dataframe. Because the size of the json is too big, the json was read by data chunks.
- Plot the distribution of review counts for businesses.

- Load the business json. Filter out the businesses that have “Restaurants” in their categories. Sample 2000 restaurants because the original dataset is too big.
- Create a dataframe that contains only restaurants and these restaurants have reviews in the review dataframe.
- There are 188593 businesses in the yelp_academic_dataset_review dataset. There are 57173 businesses that are restaurants. In order to extract the restaurants from the review dataset, merge the restaurants dataframe with the reviews dataframe on business_id.
- A sample size of 30000 was taken to make it faster.

Exploratory Data Analysis



The 5 star reviews have the most counts, followed by 4, 3, 1 and 2.



It shows that people who tend to review a business as good (4 or 5 stars) have shorter reviews (86 or 113 words), and the reviews that have poorer ratings tend to be longer words.

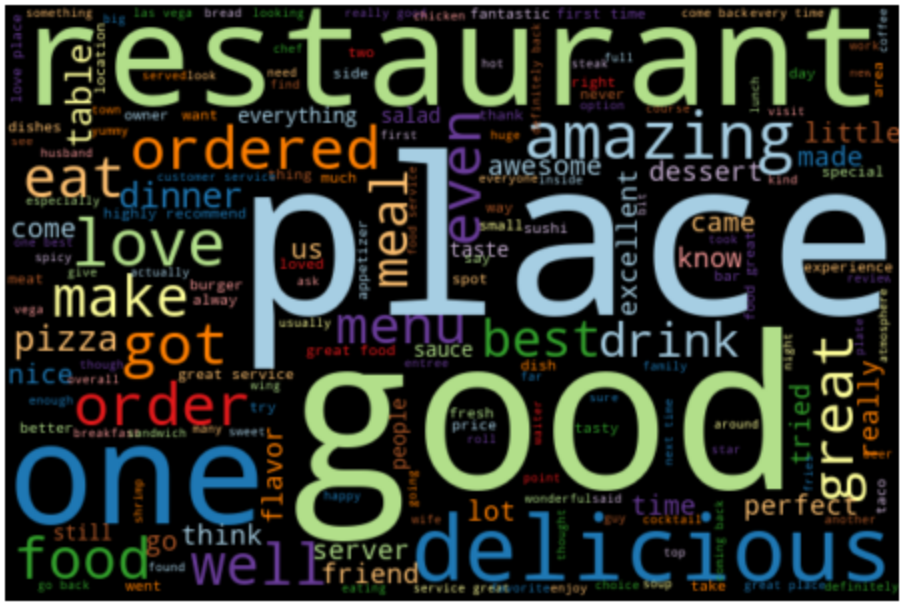
Inferential Statistics

One-way ANOVA was used to tests the null hypothesis that the distribution of the text lengths of all stars have the same mean length. The conclusion is the null hypothesis should be rejected because the p-value is 0, meaning the mean text lengths for different stars are significantly different.

Text Mining

Steps:

- Import the nltk, string, wordcloud libraries to do the text mining.
- Define a “cleaning” function to remove the punctuations, lowercase all case-based characters, and remove common English stopwords.
- Use the function “cleaning” to process the reviews.
- Create a bag of words by joining the words in cleaned text for each star.
- Generate the word clouds.



Word Cloud of Reviews of 5 Stars

Word Cloud of Reviews of 4 Stars

Word Cloud of Reviews of 1 Star

The 5 star reviews use positive words like good, great, love, delicious, amazing. The 4 star reviews have similar words as 5 stars, but not as many "love", "amazing", "best" as in 5 star reviews. The most frequent word for all reviews would be a neutral word "place". The most request words in 1 star reviews are neutral words such as place, food, time, said, total, table, order, service, never.

Prediction of the Review Stars

The code of the in-depth machine learning analysis to predict the review stars was in a jupyter notebook file “Sentiment Analysis of the Yelp Reviews”.

Steps:

- Create another column in the review dataframe called “clean_text” after removing the punctuations, lowercasing all case-based characters, and removing common English stop words.
- Use CountVectorizer to convert the “clean_text” to a matrix of token counts and convert this matrix to Compressed Sparse Column format by using .tocsc(). Use that as the feature variables.
- Specify the values of the stars as the target variables.
- Split the samples into train set (80%) and test set (20%).
- Apply supervised learning algorithms (Multinomial Naive Bayes, Gradient Boosting Machine, and Random Forest) to fit the training set and predict the labels of the test set.
- Apply different metrics to evaluate the models (accuracy scores, confusion matrix, classification report).

Summary of findings:

1) The sample size is 30000, and the number of features depends on the vectorizer. Feature number is 50680 with CountVectorizer, but is reduced to 5462 after specifying the min_df=20, max_df=11000. It increases to 11940 after adding 2-gram, and 12420 after adding 2-gram and 3-gram.

2) CountVectorizer() and MultinomialNB(): The accuracy scores for the train set and test set are 0.760 and 0.581 respectively. In multi-label classification, classifier.score is the subset accuracy which is a harsh metric since we require for each sample that each label set be correctly predicted. There are five categories for the classifier, so an accuracy score of 0.58 is not too bad.

   The reason accuracy score is used as the main evaluation metric is that this is a multicategorical classification problem, there would be 5 precisions and 5 recalls. Accuracy provides a single value that measures the correctness of the classification conservatively. A star 5 review being misclassified as star 4 or star

1 is equally wrong. So accuracy is a conservative metric. If we treat the target variable, the review stars, as a continuous variable, this can be a regression problem too, and other metrics like RMSE can be used too.

Confusion Matrix:

```
array([[ 476,   93,   54,   31,   38],
       [ 184,   91,  161,   99,   43],
       [  63,   40,  240,  369,   92],
       [  20,    9,   76,  705,  729],
       [  21,   10,   22,  360, 1974]])
```

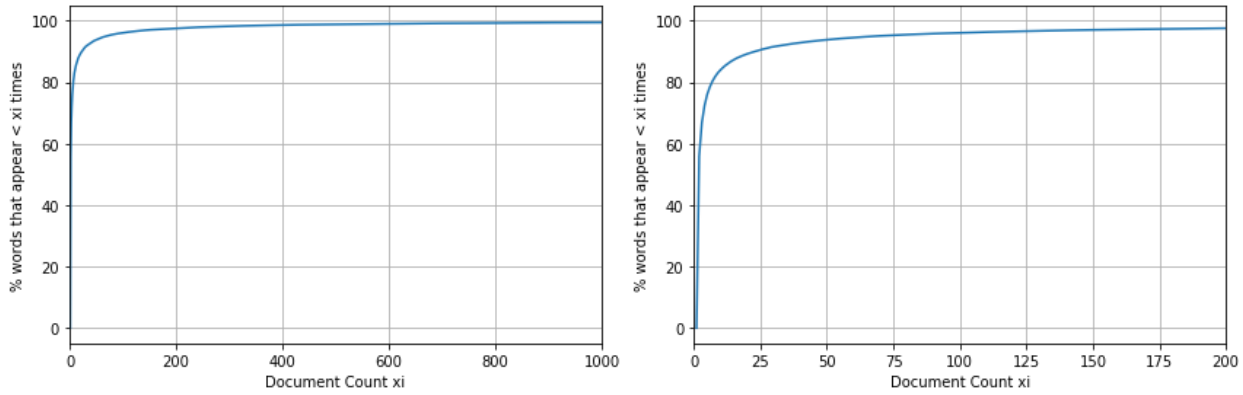
Classification Report:

	precision	recall	f1-score	support
1	0.62	0.69	0.65	692
2	0.37	0.16	0.22	578
3	0.43	0.30	0.35	804
4	0.45	0.46	0.45	1539
5	0.69	0.83	0.75	2387
micro avg	0.58	0.58	0.58	6000
macro avg	0.51	0.49	0.49	6000
weighted avg	0.55	0.58	0.56	6000

All metrics, precision, recall and f1-score are highest for 5 star reviews, followed by 1 star reviews. It shows that the naive bayes model works better for the extremely good or bad ratings. The recalls for class 2 and 3 are very low, because neutral customer experiences might not have distinct features to be labeled as neutral and also because the sample size are much smaller than 4 or 5 star reviews.

Hyperparameters tuning for CountVectorizer:

An elbow method was used to decide the min_df (min_df=20), and the max_df was set at 11000, because the frequency for the most common non-stopword “place” is 11767. The accuracy scores for the train set and test set are 0.677 and 0.596 respectively. The accuracy score for the test set after using min_df is slightly better than without min_df.



Hyperparameters tuning for Multinomial Naive Bayes:

The best hyperparameter alpha for the Multinomial Naive Bayes is 1.2.

When using the best alpha and specifying min_df and max_df, the accuracy score for the test set can improve slightly to 0.597.

2) TfidfVectorizer() and MultinomialNB()
The accuracy score for the train set is 0.611 and the accuracy score for the test set is 0.549, lower than using CountVectorizer.

3) n-grams CountVectorizer()
N-grams are phrases containing n words next to each other. This is useful because "not good" and "so good" mean very different things.
The accuracy score for the test set is 0.609 after adding 2-gram and 0.606 after further adding 3-gram.

The accuracy score after adding bigram to the unigram countvectorizer model is better than without it. The reason might be that many meaningful expressions are in 2 word phrases, and unigram does not consider these expressions that could make a difference in the meaning of a sentence. And the results showed that adding trigram does not add further value to the prediction. This is because as n increases, the model does not scale well since the feature set becomes more sparse.

4) CountVectorizer and Random Forest Classification
The accuracy score for the train set is 1.000 and the accuracy score for the test set is 0.560. The accuracy score for the test set shows there is overfitting for the train set. It

took very long time to tune the hyperparameters for the random forest because the feature number is too large and my computer died every time if it was trying to tune the hyperparameters grid.

5) CountVectorizer and Random Forest Classification
The best parameters after using RandomizedSearchCV gave a best accuracy score for the test set of 0.550.

Among the algorithms that were tried, using the CountVectorizer of 2-gram and Multinomial Naive Bayes has got the best predictive power, the test accuracy score is 0.609. And the best alpha is 0.4.

As a baseline for the machine learning models, if I predict the most common class (5 stars) for every review, the accuracy would be 0.398, much lower than the accuracy of using the above algorithms.

Strongly Predictive Features:

By using sklearn feature ranking with recursive feature elimination, rfe(), we can get the then most important features of a model.

The ten most important features for the countvectorizer using unigram vectorizer and Naive Bayes are:
“Disrespectful”, “flawless”, “insulting”, “locked”, “magnificent”, “manny”, “prakash”, “redeeming”, “remembers”, and “sublime”.

The ten most important features for the countvectorizer using 1 and 2-gram vectorizer and Naive Bayes are:
“best italian”, “give zero”, “instead 5”, “manager told", "notch service", "place rocks", "recommend everyone", "took great", "worst customer", and "worst place".

The ten most important features for the countvectorizer using 1, 2 and 3-gram vectorizer and Naive Bayes are:
"give zero", "give zero stars", "great care us", "instead 5", "notch service", "reason didn"t give", "took great", "top notch service", "worst customer service", and "worst place".

Good Words and Bad Words:

Here we define the good words to be the words that have the highest probability in reviews with 5 stars and define the bad words to be words that have the highest probability in reviews with 1 stars, and get the probability by using the predict_log_proba() method of the MultinomialNB model.

Best words: "delish", "gem", "polenta", "unique", "beautifully", "perfect", "perfection", "delightful", "hearty", and "pumpkin".

Worst words: "flavorless", "poisoning", "unprofessional", "worst", "tasteless", "aok", "terrible", "unacceptable", "luke", and "lacked".

Popular Food Recommendation

Steps:

- Get a list of common food items from nltk.corpus.wordnet
- Define a function that creates a list of food items that appear in the reviews for a restaurant given the business ID.
- Define a function to list the top 10 most recommended food items by calculating the estimated score using shrinkage estimator of reviews in which the food item appears for a restaurant with known business_id. A shrinkage estimator takes account of number of votes and the mean of the score.
- A similar function is created to list the top 10 food items to avoid if a restaurant's business ID is known.
- Find what are the top 20 most common food items.
- Find what are the average ratings of the most common food items.
- Find what are the most popular food generally (common and highly rated).
- Find what are the food to avoid generally (common and poorly rated).

Summary of findings:

- The functions can successfully generate a list of the best and worst food if a restaurant's business ID is known. The list could be slightly different when specifying different minimum mentions (m) required to be listed in the most popular dishes. This is the result for a sample restaurant for m = 2, 3, or 4:

['macaroni', 'pork', 'mango', 'avocado', 'plate', 'noodle', 'tuna', 'orange', 'guava', 'fish']
['macaroni', 'pork', 'mango', 'avocado', 'plate', 'noodle', 'tuna', 'orange', 'guava', 'fish']
['macaroni', 'pork', 'mango', 'avocado', 'plate', 'date', 'noodle', 'tuna', 'orange', 'guava']

- Test the function to list food items to avoid for five random restaurants:

[['eggplant', 'chocolate', 'strawberry', 'green', 'coconut', 'beef', 'bread', 'mango', 'prawn', 'spinach'],
['beef', 'raspberry', 'veau', 'bread', 'spaghetti', 'truffle', 'plate', 'tart', 'pumpkin', 'bacon'],
['onion', 'mushroom', 'pepper', 'meat', 'buffalo', 'pineapple', 'ham', 'leftovers', 'shoulder', 'wiener'],
['chicken', 'bacon', 'fish', 'cheese', 'pie', 'buffalo', 'lettuce', 'heart', 'pepperoni'],
['loaf', 'collards', 'confit', 'beef', 'shoulder', 'pepper', 'butter', 'meatloaf', 'scone', 'gem']]

- The top 20 most common food items in this data set are:

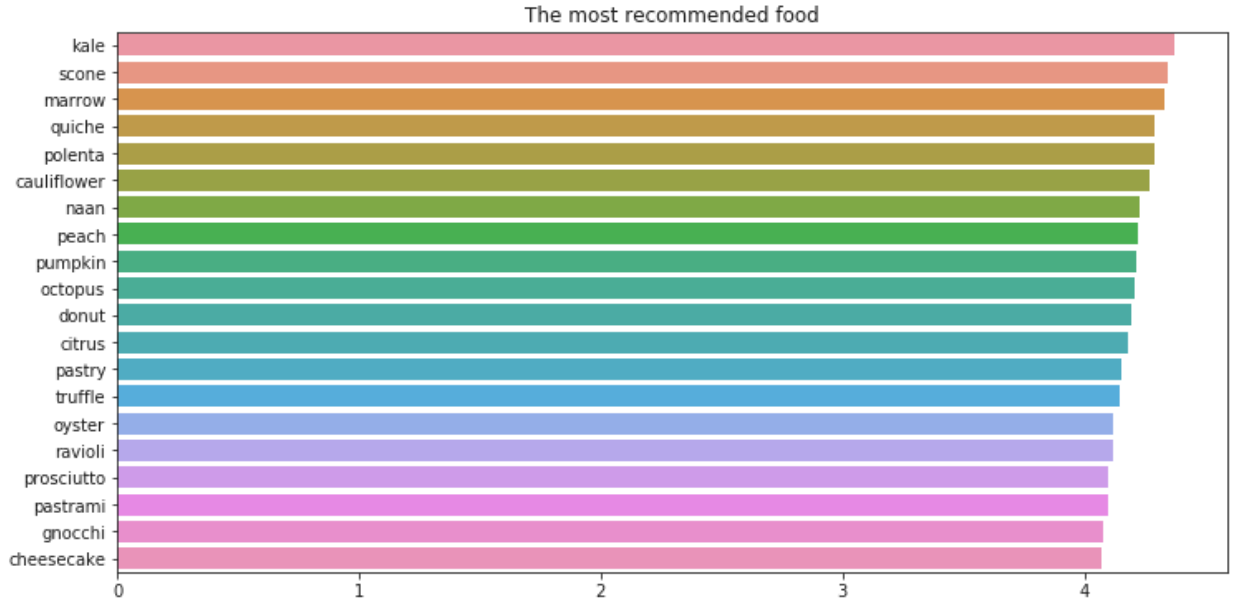
['chicken', 'cheese', 'side', 'fries', 'meat', 'beef', 'bread', 'steak', 'shrimp', 'fish', 'pork', 'roll', 'chips', 'plate', 'pasta', 'bacon', 'salmon', 'green', 'crab', 'cake']

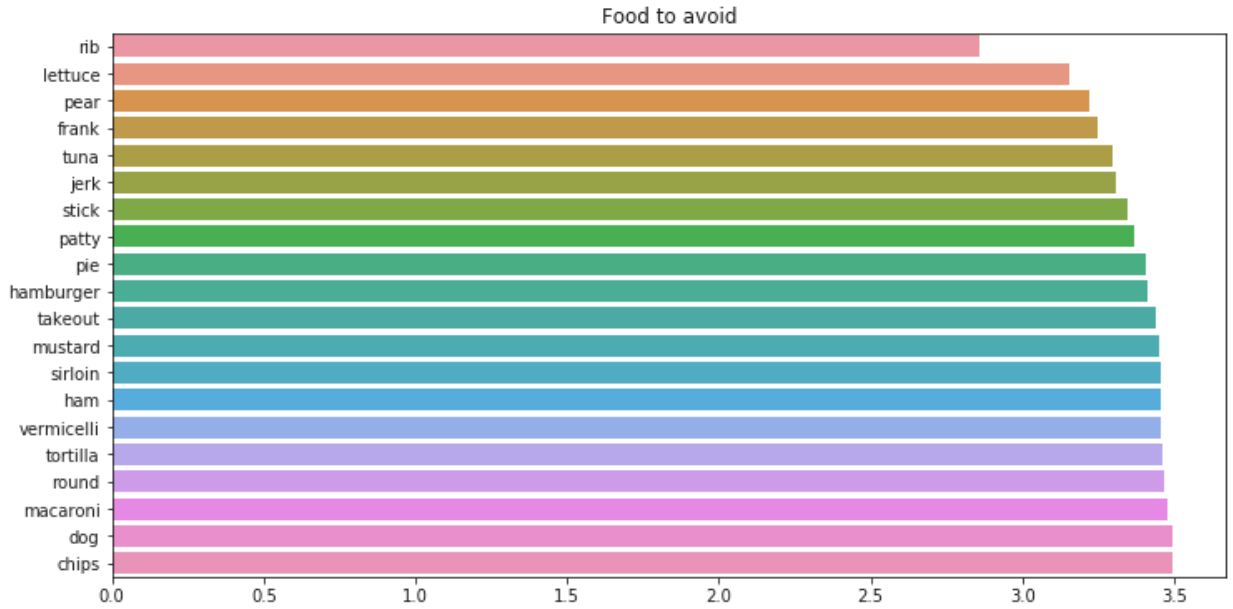
And their ratings:
{'chicken': 3.6747676912080056, 'cheese': 3.7409847434119277, 'side': 3.630952380952381, 'fries': 3.6484069312465066, 'meat': 3.5606865549652635, 'beef': 3.7943056362579894, 'bread': 3.748766041461007, 'steak': 3.7081949058693247, 'shrimp': 3.8401065956029314, 'fish': 3.684078036500944, 'pork': 3.8251695553880936, 'roll': 3.7698492462311557, 'chips': 3.4972375690607733, 'plate': 3.5302850356294537, 'pasta': 3.829181494661922, 'bacon': 3.80998613037448, 'salmon': 3.8181818181818183, 'green': 3.8101391650099403, 'crab': 3.838757396449704, 'cake': 3.918649270913277}

- The 20 most common food items are mostly meat names. "Chicken" is the most common food. Beside meat, "cheese", "side", "fries", "roll", "chips", "plate", "pasta", "green", "cake" are also in the list.

  The average stars for the reviews containing the 20 most common food items are not too high, between 3.5-3.9. "Cake" has the highest ratings, and "chips" has the lowest among these 20 food items.

- The most recommended food and food you should avoid:





The most recommended food in this list is kale, followed by scone, marrow, quiche, and polenta. And the top of the list of food to avoid, there are rib, lettuce, pear, and frank. How can we use this list when we go to a restaurant? If you like seafood, octopus and oysters are good choices, but try to avoid tuna. If you like dessert, please order scones, donuts, truffle, cheesecake and avoid pie. Do not eat fast food, they are not healthy, and are rated poorly at yelp (hamburger, (hot) dog, chips, etc).

However, the rating here does not necessarily mean the quality of a dish of kale is on average higher than a rib dish. It might mean that a kale or octopus lover is happier about the dining generally than a rib lover or takeout lover.

Recommendation

This food recommendation function can be utilized by Yelp or any third party mobile APP producer that can access the yelp review data. For example, Yelp can help restaurants to roll out its "popular dishes feature" to help the customers in deciding what to order from the restaurant with whom they aren't familiar. The restaurants can use this information to advertise their popular dishes at their Yelp webpage and improve their customer dining satisfaction. Restaurant owners can see what food their customers don't like, so they can improve their not so good dishes. This will boost the business of the restaurants.

Limitations

The limitations of this function of the food recommendation are that,

1) I don't have the menu item names to begin with, so I used the a food list from nltk.corpus wordnet.

2) Sometimes, there are several food items in one review, for example, "the fish is good, but the beef is awful". This model treats them as the same rating using the star rating of that review. More work could be done to analyze the reviews by sentence, and by the predicted score of that sentence in which the food item appears.

Future work

Future analysis would include:
1) labeling the food items by phrases to create a list that is more similar to the real menu, for example, if "pork" and "rib" are together, the menu item might be "pork rib".

2) the sentiment analysis of reviews by sentence, so a more accurate food recommendation model can be built.