Restaurant Recommendation and Classification Model

Pragya Shrestha & Salu Khadka, August 2017

Executive Summary

This document presents an analysis of data concerning restaurant reviews and recommending restaurants to the user based on that information. The restaurant reviews are also utilized to classify the restaurant using Text mining. The analysis is based on a scaled down version of Yelp Dataset available from Kaggle which included 11,537 businesses, 43,874 users, 229,907 reviews and 8,282 check-in sets. After exploring the data, only the restaurant related information was extracted and recommendation model was created based on restaurant ratings and classification was done using restaurant reviews.

With the explosive growth in the digital information and the number of internet users, a potential challenge has been created which hinders timely access to item of interest on the Internet. Thus, the demand of the recommendation engine has increased very much. Personalized recommendation is provided to users by the recommendation model which uses user-user based collaborative filtering. The recommendation model is simply a collaborative filtering system which calculated similarity using Pearson's correlation coefficient whereas the classification was done using k-neighbors Classifier.

Initial Data Exploration

The project used a scaled down version of Yelp Dataset available from the website of Kaggle[1]. Since, the yelp dataset consists of reviews and business information of businesses other than restaurants, the dataset was further cleaned up by removing irrelevant data.

Summary Statistics of Data before Data Clean-up

Table 1 Summary Statistics of Data

Businesses	11,537
Check-in Sets	8,282
Users	43,873
Reviews	229,907

[1] [Source: https://www.kaggle.com/c/yelp-recsys-2013/data]

Since, only restaurant information is needed so all the other businesses were deleted. Also, the reviews related to restaurants were only kept as our project is focused towards generating restaurant recommendation. The check-ins were not relevant to our project so they were deleted.

Summary Statistics of Data after Data Clean-up

Table 2 Summary Statistics of Data after Clean up

Restaurants	4503
Users	34789
Reviews	149319

Data Format

All the data (restaurants, users, reviews) are represented as list of dictionaries.

For example,

1. Sample representation of a restaurant:

```
{'city': 'Glendale Az',
 'full_address': '6520 W Happy Valley Rd\nSte 101\nGlendale
Az, AZ 85310',
 'latitude': 33.712797,
 'longitude': -112.200264,
 'new_id': 1,
 'rating': 3.5,
 'restaurant_id': 'PzOqRohWw7F7YEPBz6AubA',
 'restaurant_name': 'Hot Bagels & Deli',
 'review_count': 14,
 'state': 'AZ'}
```

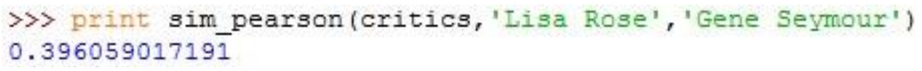
2. Sample representation of a review:

```
{'cool': 2,
 'date': '2011-01-26',
 'funny': 0,
 'rating': 5,
 'restaurant_id': 3010,
 'review': 'My wife took me here on my birthday for breakfast and it was
excellent.  The weather was perfect which made sitting outside
overlooking their grounds an absolute pleasure.  Our waitress was
excellent and our food arrived quickly on the semi-busy Saturday morning.
It looked like the place fills up pretty quickly so the earlier you get
here the better.  Do yourself a favor and get their Bloody Mary.  It
was phenomenal and simply the best I\'ve ever had.  I\'m pretty sure
they only use ingredients from their garden and blend them fresh when
you order it.  It was amazing.  While EVERY THING on the menu looks
excellent, I had the white truffle scrambled eggs vegetable skillet and
it was tasty and delicious.  It came with 2 pieces of their griddled
bread with was amazing and it absolutely made the meal complete.  It
was the best "toast" I\'ve ever had.  Anyway, I can\'t wait to go back!',
  'review_id': 1,
  'useful': 5,
  'user_id': 24538}
```
3. Sample representation of a user:
```
{'new_id': 0,
 'user_id': 'CR2y7yEm4X035ZMzrTtN9Q',
 'user_name': "b'Jim'"}
```

Methods

Similarity Calculation

In this project, similarity between the users were calculated by using Pearson’s correlation score between two users and finding the correlation between them. The correlation coefficient is a measure of how well two sets of data fit on a straight line. It returns a value between -1 and 1. A value of 1 means that the two people have exactly the same ratings for every item and value of -1 means that the two people have exactly opposite rating for every item. Figure (1) shows the calculation of similarity between users by Pearson’s correlations coefficient.


```
>>> print sim_pearson(critics,'Lisa Rose','Gene Seymour')
0.396059017191
```
Figure 1 Similarity between two users using Pearson’s correlations

Building dictionary of critics.

For this project, different users and their preferences were represented in a dictionary called critics in Python by using nested dictionary. This dictionary uses ranking from 1 to 5 as a way to express how much each of these restaurant critics (and the user) liked a given restaurant. [11] Dictionaries were used because they are convenient for experimenting with the algorithm and for illustrative purposes. It’s also easy to search and modify the dictionary. Sample critics dictionary is given below:

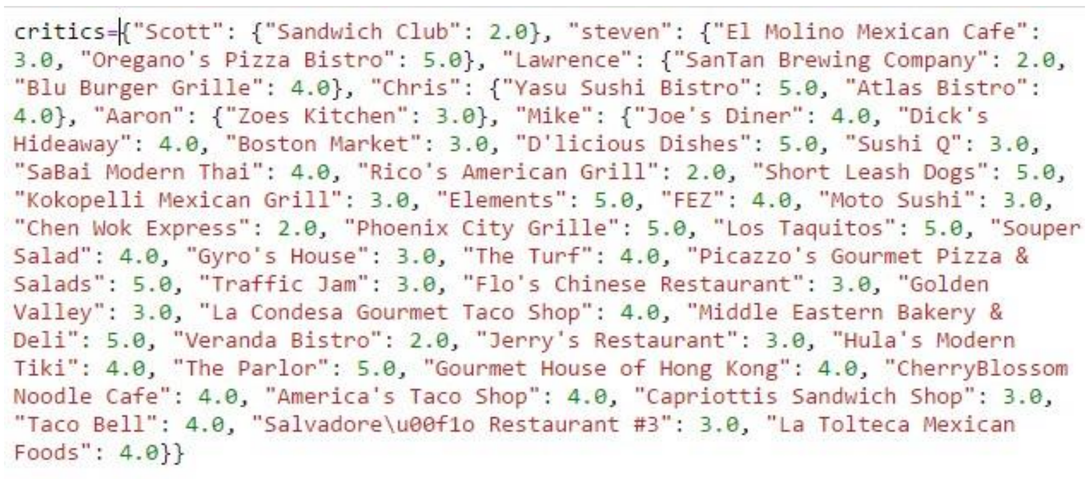

```
critics={"Scott": {"Sandwich Club": 2.0}, "steven": {"El Molino Mexican Cafe":
3.0, "Oregano's Pizza Bistro": 5.0}, "Lawrence": {"SanTan Brewing Company": 2.0,
"Blu Burger Grille": 4.0}, "Chris": {"Yasu Sushi Bistro": 5.0, "Atlas Bistro":
4.0}, "Aaron": {"Zoes Kitchen": 3.0}, "Mike": {"Joe's Diner": 4.0, "Dick's
Hideaway": 4.0, "Boston Market": 3.0, "D'licious Dishes": 5.0, "Sushi Q": 3.0,
"SaBai Modern Thai": 4.0, "Rico's American Grill": 2.0, "Short Leash Dogs": 5.0,
"Kokopelli Mexican Grill": 3.0, "Elements": 5.0, "FEZ": 4.0, "Moto Sushi": 3.0,
"Chen Wok Express": 2.0, "Phoenix City Grille": 5.0, "Los Taquitos": 5.0, "Souper
Salad": 4.0, "Gyro's House": 3.0, "The Turf": 4.0, "Picazzo's Gourmet Pizza &
Salads": 5.0, "Traffic Jam": 3.0, "Flo's Chinese Restaurant": 3.0, "Golden
Valley": 3.0, "La Condesa Gourmet Taco Shop": 4.0, "Middle Eastern Bakery &
Deli": 5.0, "Veranda Bistro": 2.0, "Jerry's Restaurant": 3.0, "Hula's Modern
Tiki": 4.0, "The Parlor": 5.0, "Gourmet House of Hong Kong": 4.0, "CherryBlossom
Noodle Cafe": 4.0, "America's Taco Shop": 4.0, "Capriottis Sandwich Shop": 3.0,
"Taco Bell": 4.0, "Salvadore\u00f1o Restaurant #3": 3.0, "La Tolteca Mexican
Foods": 4.0}}
```

Figure 2 Sample dictionary of critics in Python

Recommendation Model

The recommendation model is based on the similarity score which is calculated using Pearson's correlation coefficient. Here, Suppose A, and B be two users and R_i are the restaurants, where i = 1, 2, 3…. n. The value of n equals the total no. of restaurants. Assuming, following data from users:

Given,

Table 3 Ratings of user A

	R_1	**R_2**	**R_3**	**R_4**
User A	3.0	4.0	3.5	4.5

Table 4 Ratings of user B

	R_1	**R_2**	**R_3**	**R_5**
User B	4.0	5.0	4.5	5.0

Now, at first a dictionary of critics is created, which is a nested dictionary of user's rating. Assuming, there are only two users now,

critics= {'A':{'R_1':3.0, 'R_2':4.0, 'R_3':3.5, 'R_4':4.5}, 'B':{'R_1':4.0, 'R_2':5.0, 'R_3':3.5, 'R_5':4.5}}

Now if the user B wants recommendations, then the recommendation model compares it with every other users' in the critics based on the similarity score calculated using Pearson's correlation coefficient.

Algorithm for generating recommendation for user B,

```
a. Create a dictionary of critics of every user and their preference.
b. Compare user B with every other user in critics based on the similarity score.
    If  similarity_score > 0:
          i) Score restaurants that user B has not visited but similar user has visited,
             score[i]= similarity_score * rating of restaurant from similar user
          ii) Calculate average scores for scores calculated for each restaurant,
              avg_score = sum(score)/ len(score)
          iii) Rank the restaurants according to their avg_scores
```

iv) The recommendations for user B is the restaurants within the top20 rank.

recommendation = rankings[:20]

c. If len(recommendations) == 0:

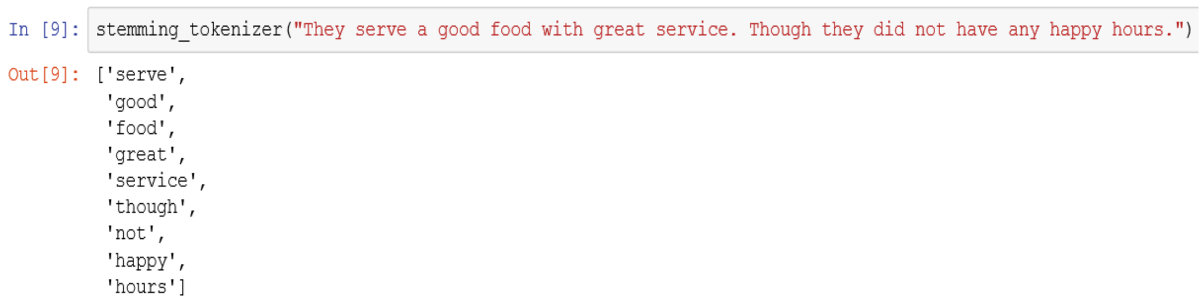
i)return the top 20 most popular restaurants from the data

recommendation = topRestaurants[:20]

d. Return the recommendations for user B

Tokening a review

In the process of text mining, the most preliminary step involves tokenization. Tokenization breaks a sentence to words called tokens. These token, further serves the system as a feature for the process classification. To tokenize a text, TextBlob library in Python was used to correct the sentence and also to lemmatize words to their root. This is useful for the next process of text mining which is a count vectorizer. In the project, only positive stop words were included as the negative could be a useful information to predict whether the text contains a positive or negative information about any fields.



```
In [9]: stemming_tokenizer("They serve a good food with great service. Though they did not have any happy hours.")

Out[9]: ['serve',
         'good',
         'food',
         'great',
         'service',
         'though',
         'not',
         'happy',
         'hours']
```

Figure 3 Tokenization of Review

Building a classifier

Any user who visits a restaurant expresses their experience through reviews and rating. In this project, a classifier that automatically classifies restaurant business reviews into the dimensions a restaurant could be related to was built. About a hundred reviews were manually inspected for restaurant businesses and 5 important dimensions were found which includes “Food”, “Service”, “Ambience”, “Deals/Discounts”, and “Worthiness”. This project experimented with popular multi-label classification approaches using “unigrams”, “bigrams”, “trigrams”, and “review ratings” as features.

The classifier can be addressed as a learning problem, where the task is to build a learner. The learner can classify a given review into respective categories. However, a review can be associated with multiple categories at the same time, it is not a binary but a multilabel classification problem.

This process of building classifier in our application is carried with the help of a python library Pipeline where three modules: CountVectorizer, TFIDF Transformer and KNeighborsClassifier works concurrently to form a stronger classifier. Later, this classifier is fitted with training data.

Formal Definition

Let H be the hypothesis of multi-label classification and C are the set of categories, X is the review text and Y is output, then:

C = {Food, Service, Ambience, Deals and Worthiness}

H: X -> Y, where Y $\subseteq$ C

Table 5 Expected outcome of text mining model

Review	**Categories**			
	Food	Service	Ambience	Deals
They have the best happy hours around, the food is good and their service is even better. When its winter, we become regulars.	1	1	N/A	1
Didn't like the food but the place has a great environment and friendly staffs.	0	1	1	N/A

Algorithm for building classifier

1. Create a tokenizer model that generates tokens by correcting sentences, eliminating stopwords and lemmatizing tokens.
2. Create a pipeline to perform a sequence of transformation, using countvectorizer, tfidf transformer and KNeighbor Classifier. The tokens from step 1 is also utilized.
3. Generate training and testing data using cross validation in the ratio of 3:1.
4. Train the classifier from step 2 with the data from step 3.
5. Return the classifier.

Algorithm for Text mining

1. Input restaurant_id
2. Fetch all the reviews from review table with restaurant id = restaurant_id
3. Form food classifier, service classifier, ambience classifier and deal classifier with the algorithm for building classifier.
4. Predict each review with classifier of step 3.
5. Compute percentage of positive predictions.
6. Return result.

Model Evaluation and Evaluation Results

In this system, Pearson's correlation coefficient was used to calculate the similarity between users and generate recommendation based on the similarity. The recommendation model worked with decent accuracy of 80.71%. The following results were obtained after testing the data against the recommendation model.

Table 6 Performance of Recommendation Model

	Precision	**Recall**	**f-score**	**Support**
Bad	0.31	0.21	0.25	87
Good	0.86	0.92	0.89	478
Avg / total	0.78	0.81	0.79	565

As, for the restaurant classifier, there were 4 separate classifiers for each label i.e. one for food, service, ambience and deals each. For food classification model, it distinguishes whether the reviewer states the food in the restaurant as good or bad and for service classification model, it distinguishes whether the reviewer states the service in the restaurant as good or bad. Similarly, the distinction for other labels are done accordingly. Each classifier model was tested separately against the respective trained classification model. The following results were obtained:

For Food Classification model,

Table 7 Performance of Food Classifier

	Precision	**Recall**	**f-score**	**Support**

Bad_Food	0.75	0.35	0.48	17
Good_Food	0.69	0.92	0.79	26
Avg / total	0.71	0.70	0.67	43

For Service Classification model,

Table 8 Performance of Service Classifier

	Precision	**Recall**	**f-score**	**Support**
Bad_Service	0.76	0.83	0.79	30
Good_Service	0.50	0.38	0.43	13
Avg / total	0.68	0.70	0.69	43

For Ambience Classification model,

Table 9 Performance of Ambience Classifier

	Precision	**Recall**	**f-score**	**Support**
Bad_Ambience	0.70	1.00	0.82	28
Good_Ambience	1.00	0.20	0.33	15
Avg / total	0.80	0.72	0.65	43

For Deals Classification model,

Table 10 Performance of Deals Classifier

	Precision	**Recall**	**f-score**	**Support**
Bad_Deals	0.85	1.00	0.92	35
Good_Deals	1.00	0.25	0.40	8
Avg / total	0.88	0.86	0.82	43

After the classification of reviews of restaurant, it was evident that the primary concern for restaurant customers was the quality of food while service, ambience and deals provided by the restaurant were secondary. This can be shown from the graph below:

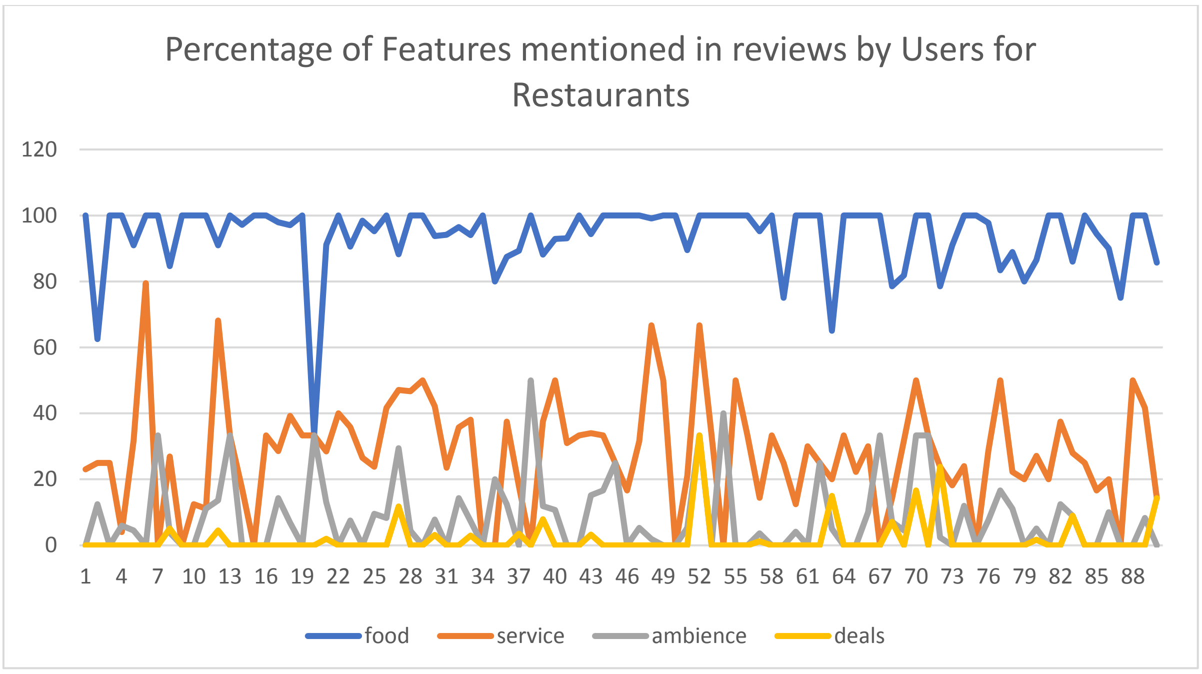


Figure 4 Percentage of Features mentioned in reviews

Conclusion

Historically, we all rely on recommendations given by our friends or relatives but this project focusses on expanding the set of people from whom we users can obtain recommendations. Also, this project aims on effectively narrowing down the choices of the users. Good recommendation creates more user-friendly interfaces. As a result, it also increases and create a number of users for business which can consequently increase the popularity of that restaurant. Significantly, it also helps the application to stand out amongst hundreds of other websites.

Similarly, the opinion mining feature can make an application smart enough to extract the information itself and strengthens the system to produce better and efficient outcomes.