Food Based Recommendation System

Ashish Aggarwal | Syed Haroon | Vikas Singh
{aggarwal.ash | haroon.s | singh.vika@husky.neu.edu

1 Introduction

Recommendation systems are very common now days and are widely used in most of the applications whose intent is to provide relevant options to its users in order to increase its sales/clicks or visits etc. In each scenario the recommendations are usually modelled around the tags allotted to each product/service being sold. These tags, sometimes, can be misleading or irrelevant, thus giving inefficient recommendations.
Also, many a times, businesses often do not understand the dynamics of the reviews they are receiving. Dynamics such as what constitutes towards a 5-star rating is understood only after the ratings have been submitted with the accompanying review.
Through this project we try to solve both these problems by (1) Building a recommendation system and (2) Predicting the rating based on User Review sentiments.
We used Yelp Dataset to perform this, so we have a business data and its corresponding review and user data as well.

2 Data

As mentioned earlier we used Yelp Dataset [1], below is its brief overview:

Contains 4,700,000 reviews from over 156,000 businesses in 12 metropolitan areas in json format. The following is the information of the data our project mainly dealt with:

1. business.json : contains the business id, business name, neighborhood, address, city, postal code, stars, review_count, is_open (0 or 1 for closed or open), attributes (free wi-fi, noise level, ambience, attire, “good for groups)

2. review.json : contains review_id, user_id, business_id, stars, date, text, useful, funny, cool

3. user.json : user_id, name, review_count, friends, compliments

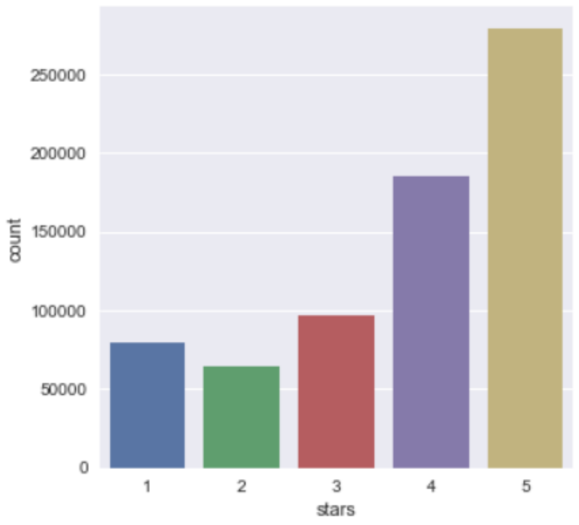
3 Pre-processing

For the project we wanted to build a model that dynamically pulls out a feature from the reviews and assigns it to a business, but soon we figured out that this was not easy given the resources and time limitations. We used apache spark databricks to run our code, and given its cluster limitation of 6GB for any community edition account we had to reduce our dataset accordingly.
But removing the rows from the reviews table could lead to an inefficient model, as we may unknowingly remove the reviews of a specific business that may make our dataset bias to a certain feature or a user may get an irrelevant recommendation due to the loss of data. So, to tackle this problem we sliced the dataset using location attribute of the business, we took the location Las Vegas, and extracted all the businesses located in Las Vegas having minimum review count of 10. Then for each business that we filtered, we extracted all reviews corresponding to that business.

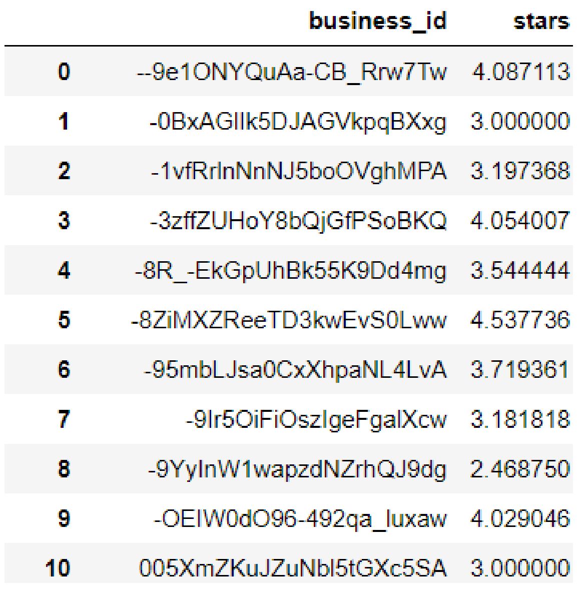
This helped us maintain the data efficiency and reduced the dataset to approximately 700000 reviews which we uploaded to databricks along with other files such as user and business. Data files were converted to csv and json according to the need. The resulting file is *slicedReview2.json*
Now we can proceed to perform EDA on our dataset and look at what more needs to be done.

4 Exploratory Data Analysis

We ran a preliminary data analysis on review and business table to figure out various questions about the dataset, like how diverse are the review in terms of star ratings:

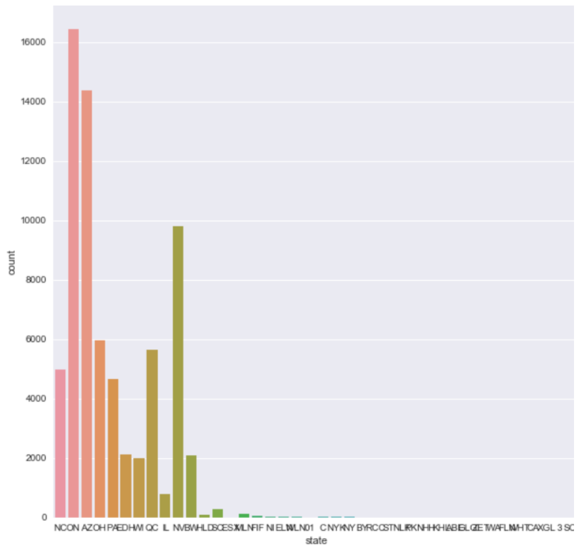


Average star rating for each business:



	business_id	stars
0	--9e1ONYQuAa-CB_Rrw7Tw	4.087113
1	-0BxAGlIk5DJAGVkpqBXxg	3.000000
2	-1vfRrlnNnNJ5boOVghMPA	3.197368
3	-3zffZUHoY8bQjGfPSoBKQ	4.054007
4	-8R_-EkGpUhBk55K9Dd4mg	3.544444
5	-8ZiMXZReeTD3kwEvS0Lww	4.537736
6	-95mbLJsa0CxXhpaNL4LvA	3.719361
7	-9Ir5OiFiOszIgeFgalXcw	3.181818
8	-9YyInW1wapzdNZrhQJ9dg	2.468750
9	-OEIW0dO96-492qa_luxaw	4.029046
10	005XmZKuJZuNbI5tGXc5SA	3.000000

State wise count of reviews:



5 Methodology

Now that we have our data explored and ready we proceeded with NLTK library [2] which provides extensive methods to ease natural language processing techniques.
For implementation we had a review table that acted as a join between the business and user table. It had review_id, business_id and user_id, hence we were able to conclude different inferences based on them. For example: Total no. of users who do not gave any review. Which users reviewed which businesses and how many reviews a user gave. We were also able to calculate average rating of a business given all its reviews and average rating given out by the user.
The first method that we assumed has to be done was to build a model that takes in a review and runs sentiment analysis on it and gives out the binary output of whether it was positive or negative, after that we could do a simple keyword match to see if the review talks about any feature or not. So we used NLTK and cleaned the data using stemmer and removed all the stop-words that seemed unnecessary, this greatly increased our throughput, and is usually practiced for NL processing. Words like ‘the’, ‘that’, ‘a’ etc. usually occur more than the meaningful words in a sentence in English language so it is considered efficient to remove them before processing the data for sentiment analysis.

Moving forward with a cleaned dataframe we used sklearn's CountVectorizer, it produces a sparse representation of the counts using scipy.sparse.coo_matrix [3]. The matrix formed by the CV take all the individual words used in the text as columns and for each review provided to it, it enters a row in the matrix with binary value for each column that denotes if that word was present in the text or not. So, the end result is something like this:

	whatever	good	pizza	like
Review1	0	1	1	0
Review2	1	0	0	1
Review3	0	0	1	1
ReviewN	1	1	1	1

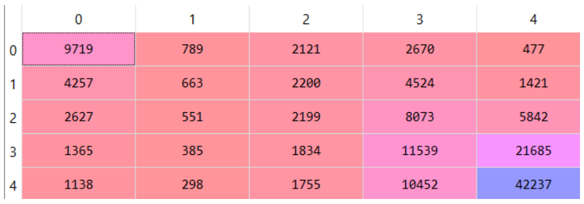
But instead of words in the column name it is mentioned with integers, so in order to view the word frequency we have to use the 'vocabulary' method of the count vectorizer. Then we got a list of all the words with its corresponding frequency in the dataset. This could help us with our feature selection.
Now after the data is processed and feature is selected, we want to run our assumed model and append the sentiment analysis score to the review, after that we will aggregate and group the score for each business and for each user, so that we can tell what score does each business get related to a specific feature, and similarly what feature does user like the most if any.
But after some preliminary analysis we found it to be less accurate and after some research we looked into NLTK Vader package. We found it to be pretty accurate given our review text and the pretrained model, therefore we rely heavily on the Vader analysis.
Vader Sentiment Analyzer provides 4 types of sentiment scores namely positive probability, i.e chances of the text being positive, negative Probability, i.e chances of text being negative, and neutral probability, i.e chances the text is neutral, it also gives a compound score to each text review, which range from -1 to 1. -1 being negative, 0 being neutral and 1 being positive.
The main idea for implementing both use cases involves sentiment analysis of reviews segregated by business and users, with this data.

Use Case 1: In order to speed up processing we limited our recommendation systems to work on only restaurants which offer pizza, burger and salad.
We then calculated 4 sentiment scores for every review of businesses pertaining to pizza, salad, and burger and stored as business sentiment file. The same was followed for user's as well and saved separately. The feature for which the average compound score of the review by user is selected as a prominent feature. The business sentiment file is then sorted for this feature against its compound score and the top 10 results are presented to the user.

Use Case 2: Similar to use case 1, the 4 sentiment scores are tracked for all reviews written by users and not just for those 3 features. Along with this the resulting rating that the business received for that review is also tracked.
This data is then used to build a Naïve Bayes classifier model to predict the rating a user would give based on this review.

6 Implementation and Experiments

After running the analysis with Vader we got 46% accuracy which was still not bad given the pre-trained Vader classifier was not trained on the same subset.
Here is the confusion matrix depicting its accuracy:



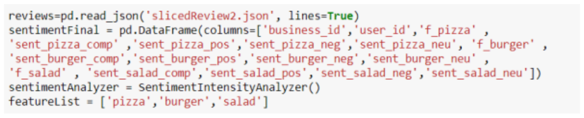
	0	1	2	3	4
0	9719	789	2121	2670	477
1	4257	663	2200	4524	1421
2	2627	551	2199	8073	5842
3	1365	385	1834	11539	21685
4	1138	298	1755	10452	42237

Now for the first use case we have to create a new dataframe, so we took a subset of the reviews dataframe and append 15 new columns to it, 5 for each feature we considered. We took 3 features namely: pizza, pasta, salad. We searched through all the reviews and run Vader sentiment analysis on it. Then filled the output in the appended columns related to that specific feature. So, the resulting dataframe had review data as well as 5 columns for each feature

consisting positive, negative, neutral, compound score of the review text along with the feature review count frequency.
Then we grouped each unique business and aggregated its sentiment score in a single row for each business to classify the business in one of the considered feature.
Similarly, we repeated the above steps for the user. Grouped and aggregated each unique user to classify him/her into one of the 3 features.

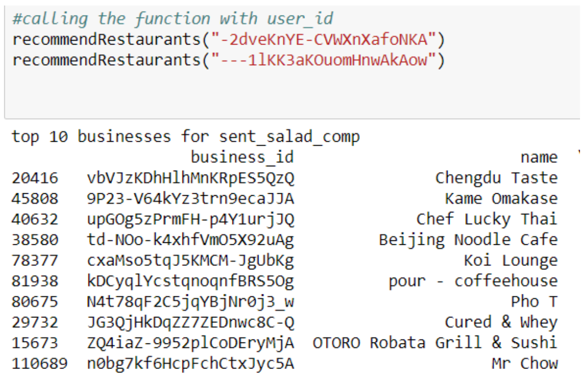
Use Case 1: Recommendation System

To track the sentiments of every business review for pizza, burger and salad. Similarly, we track the sentiments of every user written reviews having word pizza, burger and salad. Then depending on which feature has the most positive sentiment for a user, top 10 businesses with those sentiments would be presented to the user.



```
reviews=pd.read_json('slicedReview2.json', lines=True)
sentimentFinal = pd.DataFrame(columns=['business_id','user_id','f_pizza' ,
'sent_pizza_comp' ,'sent_pizza_pos','sent_pizza_neg','sent_pizza_neu', 'f_burger' ,
'sent_burger_comp','sent_burger_pos','sent_burger_neg','sent_burger_neu' ,
'f_salad' , 'sent_salad_comp','sent_salad_pos','sent_salad_neg','sent_salad_neu'])
sentimentAnalyzer = SentimentIntensityAnalyzer()
featureList = ['pizza','burger','salad']
```

Final Output:

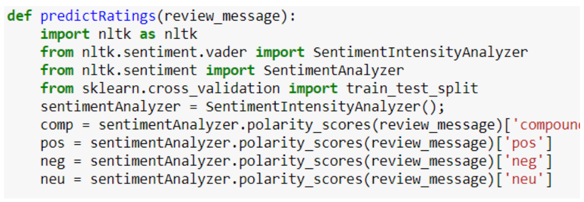


```
#calling the function with user_id
recommendRestaurants("-2dveKnYE-CVWXnXafoNKA")
recommendRestaurants("---1lKK3aKOuomHnwAkAow")
```

```
top 10 businesses for sent_salad_comp
                    business_id                    name
20416   vbVJzKDhHlhMnKRpES5QzQ           Chengdu Taste
45808   9P23-V64kYz3trn9ecaJJA           Kame Omakase
40632   upGOg5zPrmFH-p4Y1urjJQ          Chef Lucky Thai
38580   td-NOo-k4xhfVmO5X92uAg       Beijing Noodle Cafe
78377   cxaMso5tqJ5KMCM-JgUbKg              Koi Lounge
81938   kDCyqlYcstqnoqnfBRS5Og       pour - coffeehouse
80675   N4t78qF2C5jqYBjNr0j3_w                   Pho T
29732   JG3QjHkDqZZ7ZEDnwc8C-Q            Cured & Whey
15673   ZQ4iaZ-9952plCoDEryMjA  OTORO Robata Grill & Sushi
110689  n0bg7kf6HcpFchCtxJyc5A                 Mr Chow
```

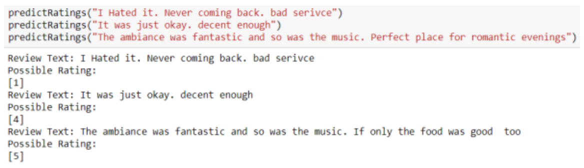
Use Case 2: Ratings Prediction

For each new review provided in the dataset the model with perform sentiment analysis and assign 4 sentiment scores to it and then predict its rating based on the model.



```
def predictRatings(review_message):
    import nltk as nltk
    from nltk.sentiment.vader import SentimentIntensityAnalyzer
    from nltk.sentiment import SentimentAnalyzer
    from sklearn.cross_validation import train_test_split
    sentimentAnalyzer = SentimentIntensityAnalyzer();
    comp = sentimentAnalyzer.polarity_scores(review_message)['compoun
    pos = sentimentAnalyzer.polarity_scores(review_message)['pos']
    neg = sentimentAnalyzer.polarity_scores(review_message)['neg']
    neu = sentimentAnalyzer.polarity_scores(review_message)['neu']
```

Final Output:



```
predictRatings("I Hated it. Never coming back. bad serivce")
predictRatings("It was just okay. decent enough")
predictRatings("The ambiance was fantastic and so was the music. Perfect place for romantic evenings")
```

```
Review Text: I Hated it. Never coming back. bad serivce
Possible Rating:
[1]
Review Text: It was just okay. decent enough
Possible Rating:
[4]
Review Text: The ambiance was fantastic and so was the music. If only the food was good  too
Possible Rating:
[5]
```

7 Conclusion

We have built the model that takes in a user and gives out top 10 businesses that the user may like based on the most prominent feature mentioned in most of user's reviews. Accuracy with Vader sentiment analyzer comes out to be 46%. We build a second use case where the model predicts the user rating based on the review text.

Please refer to the Jupyter notebook for full code and implementation.

8 References

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

[2] NLTK: http://www.nltk.org/

[3] CountVectorizer: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[4] Vader: http://www.nltk.org/_modules/nltk/sentiment/vader.html