UGCA Project 3

Building a Crowdsourced Recommendation System

Yuejia Feng, Ting-Yi Hung, Abhilasha Kanitkar, and Brian Peterson

10.31.18

Task A

We used Chrome's webscraper to scrape the reviews given by consumers to Mexican restaurants in Austin. This scraping resulted in a data file that contained 3 columns – the name of the restaurant, reviews, and ratings.

Basic Data Cleaning . . .

In [1]:
```
1 import pandas as pd
```

In [2]:
```
1 df = pd.read_excel('yelp-mexican-food.xlsx')
```

In [3]:
```
1 df.tail()
```

Out[3]:

	restaurantname	reviewtext	ratings
3995	Z'Tejas Mexican Restaurant and Grill	Great place and ambience .Server gave very goo...	[{"ratings":"","ratings-title":"5.0 star ratin...
3996	Z'Tejas Mexican Restaurant and Grill	Absolutely the worst wait staff and experience...	[{"ratings":"","ratings-title":"1.0 star ratin...
3997	Z'Tejas Mexican Restaurant and Grill	The OUTSTANDING service is what made this meal...	[{"ratings":"","ratings-title":"4.0 star ratin...
3998	Z'Tejas Mexican Restaurant and Grill	Keep it fresh. Can compete with all of the new...	[{"ratings":"","ratings-title":"5.0 star ratin...
3999	Z'Tejas Mexican Restaurant and Grill	We had large family group of 15, service was o...	[{"ratings":"","ratings-title":"1.0 star ratin...

In [4]:
```
1 print("# of comments in consideration : " ,len(df))
```

```
# of comments in consideration :  4000
```

In [5]:
```
for i in range(0, len(df.iloc[:,2])):
    df.iloc[:,2][i] = float(df.iloc[:,2][i].split('"')[7].split(' ')[0])
#df.iloc[:,2]
```

In [6]:
```
len(df.iloc[:,2])
```

Out[6]:
```
4000
```

In [7]:
```
comments=list(map(lambda x: x.lower(), df.iloc[:,1]))
comments
```

Out[7]:
```
['i need to start off by saying that i am mexican and love mexican food and i
look forward to saturday and sunday breakfast. \xa0this past saturday i made
the mistake of going back to angies for breakfast. \xa0i had been there for l
unch and although it was a lil pricey the puffy taco was good so i thought ho
w bad can breakfast be? \xa0well we sit down and get approached by the grouch
iest woman ever...if she is the owner or front house manager she shouldn\'t b
e waiting tables, she really has no business in the service industry. \xa0she
takes our order and leaves. \xa0never brings us chips but does drop off a dol
lop of salsa...brings me a water because she has to make lemonade i ordered h
alf tea half lemonade. \xa0she brings me a tea with a drop of lemonade. \xa0i
have to add sweetner. \xa0when i bite into my taco, the chorizo is sour. \xa0
i am mexican and eat chorizo all the time, i know what spoiled chorizo smells
like, i spit it out and flag down the other waitress since ours refuses to co
me back to the table. \xa0i tell her what is wrong and ask for something els
e. well our original waitress returns and now suddenly only speaks spanish an
d tells me "why don\'t you like th chorizo. \xa0what\'s wrong with it?" \xa0i
tell her she rolls her eyes and says "well what do you want?" \xa0i tell her
i reordered but would like a refill on my drink. \xa0she then says "we charge
for refills" \xa0you will be charged for another tea and another lemonade. \x
```

In [8]:
```
from nltk.corpus import stopwords
from nltk.corpus import brown
from nltk.tokenize import word_tokenize,sent_tokenize
from string import punctuation, digits
from collections import Counter
stopwords_set = set(stopwords.words('english'))
brown_set = set(brown.words())
characterstoclean = r'?!,:,/\"-+=@#$%^&*()><{}[]|' + r"'"
punc = punctuation
digi = digits
```

In [9]:
```
word_list = []
for comment in comments:
    for char in characterstoclean:
        comment = comment.replace(char, '')
    for word in word_tokenize(comment):
         if word in brown_set and word not in stopwords_set and word not in c
                word_list.append( word )
len(word_list)
```

Out[9]:
```
165818
```

In [10]:
```
cnt =Counter(word_list)
```

In [11]:
```
cnt.most_common(1000)
```

Out[11]:
```
[('food', 2850),
 ('good', 2296),
 ('place', 2016),
 ('great', 1850),
 ('like', 1306),
 ('service', 1265),
 ('get', 1259),
 ('one', 1242),
 ('really', 1101),
 ('back', 1061),
 ('time', 1003),
 ('go', 989),
 ('also', 914),
 ('would', 897),
 ('got', 871),
 ('delicious', 831),
 ('best', 799),
 ('chicken', 780),
 ('ordered', 779),
```

Task B & Task C

Replacements

In [12]:
```
food = ['tasty','quality','delicious','flavorful', 'mouthwatering']
service = ['speed', 'friendliness','friendly','fast', 'quick']
price = ['cheap', 'inexpensive', 'reasonable', 'reasonably', 'affordable']
location = ['interstate','highway','road','corner','accessible','convenient'
```

In [13]:
```
new_comments = []
new = str()
for comment in comments:
    for word in word_tokenize(comment):
        if word in service:
            comment = comment.replace(word,'service')
        elif word in food:
            comment = comment.replace(word,'food')
        elif word in price:
            comment = comment.replace(word, 'price')
        elif word in location:
            comment = comment.replace(word, 'location')
        else:
            comment = comment

    new_comments.append(comment)
print(len(new_comments))
```

```
4000
```

In [14]:
```
new_comments[1]
```

Out[14]:
```
'this seems to be a love-it-or-hate-it type place. i\'ll admit, the service is
not perfect, every dish is not a masterpiece, the margaritas are not great, and
yet i\'ve been going here weekly for over 5 years because i\'m addicted to the
corn tortillas and salsa. i\'ve not noticed any change in food food, and on the
contrary, find it consistently satisfying. \n\nthe unique corn tortillas, which
others have described with disdain as "thick and chewy" are just that, and that
\'s why i love them; especially in the form of tacos, in which case they are li
ghtly fried. if you\'re looking for healthy: no. but if you\'re looking for foo
d: oh yeah! \n\nif you think the food is bland, you\'re not using enough of the
fantastic salsa, which is fresh, super spicy and full of nice veggies - just th
e way i like it. (occasionally, and sadly, the salsa gets slightly less hot, wh
ich angie explains is due to low availability of hot jalapenos at certain times
of the year.) \n\nin defense of some of the nay-saying: \n\nyes, angie and some
of her staff are not dripping with service, but i find them genuine, appreciati
ve and hardworking. a few of them are really nice! \n\nyes, they use american c
heese, which i happen to enjoy. people - it\'s tex-mex after all. \n\nyes, the
weekday lunch hour is super busy. it would not be if this was really a one- or
two-star establishment. tips: there are a few extra parking spaces underneath t
he restaurant, location from the alley behind. once inside, make your way past
the register line into the dining room for faster seating. arriving at 11:00 or
1:00 will usually eliminate any waiting. \n\nyes, angie has a policy of no sepa
rate checks at the table. however, she is more than happy to separate any check
at the register - however you like.'
```

In [15]:
```
word_list_new = []
for comment in new_comments:
    for char in characterstoclean:
        comment = comment.replace(char, '')
    for word in word_tokenize(comment):
         if word in brown_set and word not in stopwords_set and word not in c
                word_list_new.append(word)
len(word_list_new)
```

Out[15]: 165952

In [16]:
```
cnt=Counter(word_list_new)
cnt.most_common(1000)
```

Out[16]:
```
[('food', 4358),
 ('good', 2296),
 ('service', 2176),
 ('place', 2016),
 ('great', 1850),
 ('like', 1306),
 ('get', 1259),
 ('one', 1242),
 ('really', 1101),
 ('back', 1061),
 ('time', 1003),
 ('go', 989),
 ('also', 914),
 ('would', 897),
 ('got', 871),
 ('best', 799),
 ('chicken', 780),
 ('ordered', 779),
 ('order', 755),
```

Task D

Perform a cosine similarity analysis between the attribute set and the reviews

In [17]:
```
#Defining an attributes vector - against which each comment vector will be me
attributes_vector=Counter(['service','price','food','location'])
attributes_vector
```

Out[17]: Counter({'service': 1, 'price': 1, 'food': 1, 'location': 1})

In [18]:
```
#defining a function to convert the text from a comment to a word vector (bas
def text_to_vector(comment):
    comment_word_list = []
    for char in characterstoclean:
        comment = comment.replace(char, '')
    for word in word_tokenize(comment):
         if word in brown_set and word not in stopwords_set and word not in c
                comment_word_list.append( word )

    return Counter(comment_word_list)

```

In [19]:
```
# Defining cosine similarity function
import math
def cosine_similarity(vector1, vector2):
    intersection = set(vector1.keys()) & set(vector2.keys())
    numerator = sum([vector1[x] * vector2[x] for x in intersection])

    sum1 = sum([vector1[x]**2 for x in vector1.keys()])
    sum2 = sum([vector2[x]**2 for x in vector2.keys()])
    denominator = math.sqrt(sum1) * math.sqrt(sum2)

    if not denominator:
        return 0.0
    else:
        return float(numerator) / denominator
```

In [20]:
```
#Testing the above code for a single comment
vec1 = text_to_vector(new_comments[1])
cs = cosine_similarity(attributes_vector , vec1)
cs
```

Out[20]: 0.2587274448341005

In [21]:
```
df.iloc[1,0]
```

Out[21]: 'Angie’s Mexican Restaurant'

In [22]:
```
#Calculating the cosine similarity for each comment with the attributes vecto
#and creating a list of tuples (restaurantname,comment,cosine_similarity_scor
cosine_similarity_of_comments = []
for i in range(len(new_comments)):
    cosine_similarity_of_comments.append((df.iloc[i,0],new_comments[i],cosine

```

In [23]:
```
#Just checking one element, if it worked fine
cosine_similarity_of_comments[1]
```

Out[23]:
```
('Angie’s Mexican Restaurant',
 'this seems to be a love-it-or-hate-it type place. i\'ll admit, the service is
not perfect, every dish is not a masterpiece, the margaritas are not great, and
yet i\'ve been going here weekly for over 5 years because i\'m addicted to the
corn tortillas and salsa. i\'ve not noticed any change in food food, and on the
contrary, find it consistently satisfying. \n\nthe unique corn tortillas, which
others have described with disdain as "thick and chewy" are just that, and that
\'s why i love them; especially in the form of tacos, in which case they are li
ghtly fried. if you\'re looking for healthy: no. but if you\'re looking for foo
d: oh yeah! \n\nif you think the food is bland, you\'re not using enough of the
fantastic salsa, which is fresh, super spicy and full of nice veggies - just th
e way i like it. (occasionally, and sadly, the salsa gets slightly less hot, wh
ich angie explains is due to low availability of hot jalapenos at certain times
of the year.) \n\nin defense of some of the nay-saying: \n\nyes, angie and some
of her staff are not dripping with service, but i find them genuine, appreciati
ve and hardworking. a few of them are really nice! \n\nyes, they use american c
heese, which i happen to enjoy. people - it\'s tex-mex after all. \n\nyes, the
weekday lunch hour is super busy. it would not be if this was really a one- or
two-star establishment. tips: there are a few extra parking spaces underneath t
he restaurant, location from the alley behind. once inside, make your way past
the register line into the dining room for faster seating. arriving at 11:00 or
1:00 will usually eliminate any waiting. \n\nyes, angie has a policy of no sepa
rate checks at the table. however, she is more than happy to separate any check
at the register - however you like.',
 0.2587274448341005)
```

In [24]:
```
#In place sorting of comments based on cosine similarity score
cosine_similarity_of_comments.sort(key=lambda tup: tup[2],reverse=True)
```

In [25]:
```
cosine_similarity_of_comments[0]
```

Out[25]:
```
('Jefes Street Tacos',
 'plain and simple: \xa0food street tacos. price, food, location. smiling servi
ce and a food plate.',
 0.7276068751089989)
```

In [26]:
```
#taking top 200 comments with highest cosine similarity score
top_200_comments_cs_similarity= cosine_similarity_of_comments[:200]
```

In [27]:
```
df_comment_cosine = pd.DataFrame(top_200_comments_cs_similarity)
#df_comment_cosine.columns['Index','Restaurant','Comment','CosineSimilarity']
df_comment_cosine.head()
```

Out[27]:

	0	1	2
0	Jefes Street Tacos	plain and simple: food street tacos. price, f...	0.727607
1	La Pena	great location, service owner, food food, very...	0.608781
2	Las Trancas	tacos are food and can't beat the price. the g...	0.588348
3	Taquería Chapala	this place is awesome! if you are looking for ...	0.588348
4	Chipotle Mexican Grill	i eat at this location pretty often. service i...	0.577350

Task D

Take average of sentiment score grouped by restaurant

In [28]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()
```

In [29]:
```
def cal_sentiment_scores(sentence):
    snt = analyser.polarity_scores(sentence)
    return snt['compound']
```

In [30]:
```
unique_restaurants = ((pd.DataFrame(top_200_comments_cs_similarity)).iloc[:,0
```

In [31]:
```
len(unique_restaurants)
```

Out[31]: 59

In [32]:
```
restaurant_average_sentiment = []

for res in unique_restaurants: #For every unique restaurant
    avg_sentiment = 0;
    sentiment_arr = []
    for comment in top_200_comments_cs_similarity:
        if(comment[0] == res): #if the comment is for the restaurant in consi
            sentiment_arr.append(cal_sentiment_scores(comment[1]))

    avg_sentiment = sum(sentiment_arr)/len(sentiment_arr)
    restaurant_average_sentiment.append((res,avg_sentiment))
```

In [33]:
```
#Sort the restaurant_average_sentiment in descending order
restaurant_average_sentiment.sort(key=lambda tup: tup[1], reverse=True)
df_restaurant = pd.DataFrame(restaurant_average_sentiment)
df_restaurant.columns = ['Restaurant', 'Sentiment Score']
df_restaurant.head(3)
```

Out[33]:

	Restaurant	Sentiment Score
0	Austin Taco Project	0.96790
1	Licha's Cantina	0.96330
2	Taqueria Los Altos	0.96115

Task E

Based on the above calculation, we recommend the following restaurants:

In [34]:
```
for i in range(3):
    print(str(i+1) + ') ' + df_restaurant.iloc[i,0] + '\n')
```

```
1) Austin Taco Project

2) Licha's Cantina

3) Taqueria Los Altos
```

Task F: Recommendation based only on average of ratings of the customer

In [35]:
```
restaurants = (df.iloc[:,0]).unique()
```

In [36]:
```
print("# of restaurants in total comments : ",len(restaurants))
```

```
# of restaurants in total comments :  93
```

In [37]:
```
#Create a deep copy of the original data frame and drop the comments column
df_ratings = df.copy(deep=True)
df_ratings=df_ratings.drop(['reviewtext'],axis=1)
df_ratings.head()
```

Out[37]:

	restaurantname	ratings
0	Angie’s Mexican Restaurant	1
1	Angie’s Mexican Restaurant	3
2	Angie’s Mexican Restaurant	1
3	Angie’s Mexican Restaurant	4
4	Angie’s Mexican Restaurant	5

In [38]:
```
restaurant_ratings=list(df_ratings.values.tolist())
restaurant_ratings[:5]
```

Out[38]:
```
[['Angie’s Mexican Restaurant', 1.0],
 ['Angie’s Mexican Restaurant', 3.0],
 ['Angie’s Mexican Restaurant', 1.0],
 ['Angie’s Mexican Restaurant', 4.0],
 ['Angie’s Mexican Restaurant', 5.0]]
```

In [39]:
```
#Now calculate the average rating of each restaurant
restaurant_avg_rating = []
for r in restaurants: #for each restaurant
    avg_rating = 0
    per_rest_arr = []
    for i in range(len(restaurant_ratings)):
        if(restaurant_ratings[i][0] == r): #if the restaurant is same as rest
            per_rest_arr.append(restaurant_ratings[i][1])

    avg_rating = sum(per_rest_arr)/len(per_rest_arr)
    restaurant_avg_rating.append((r,avg_rating))


```

In [40]:
```
#Sort the restaurants by decreasing average ratings
restaurant_avg_rating.sort(key=lambda tup: tup[1], reverse=True)
restaurant_avg_rating[:5]
```

Out[40]:
```
[('Antojitos Guatemala', 5.0),
 ('Asador', 5.0),
 ('Dave’s Tacos', 5.0),
 ('Discada', 5.0),
 ('Nissi Vegan Mexican Cuisine VegMex', 5.0)]
```

Based on the average ratings calculation, the following restaurants will be recommended:

In [41]:
```
for i in range(3):
    print( str(i+1) + ') ' + restaurant_avg_rating[i][0] + '\n')
```

```
1) Antojitos Guatemala

2) Asador

3) Dave’s Tacos
```

In [42]:
```
pd.concat([ df[df['restaurantname'] == 'Antojitos Guatemala'], df[df['restaur
```

Out[42]:

	restaurantname	reviewtext	ratings
92	Antojitos Guatemala	Skip the taco chains, this place is amazing! T...	5
93	Antojitos Guatemala	I had the pastor tacos and an agua fresca. The...	5
94	Antojitos Guatemala	This is a great taco stand... Amazing breakfas...	5
173	Asador	This place is located in the courtyard behind ...	5
174	Asador	Amazing tacos. Get the guac if you get chips. ...	5
175	Asador	THESE ARE THE BEST TACOS IN TOWN!!\nGet the br...	5
893	Dave’s Tacos	Fantastic tacos! Fresh ingredients and great p...	5
894	Dave’s Tacos	Deliciously simple and authentic tacos without...	5
895	Dave’s Tacos	Absolutely amazing tacos. Carnitas and shrimp ...	5
896	Dave’s Tacos	Ordered a tasty Barbacoa egg and cheese burrit...	5
897	Dave’s Tacos	Wow! This place is one you must check out . I ...	5
898	Dave’s Tacos	My husband's barber recommended we have lunch ...	5

As you can see above, the recommendations in Task F are very different from Task E. Notice the commonalities of our recommendations in Task F. The majority of reviews are exclaiming the excellent tacos at each venue. However, system isn't considering the service, location, or the price attributes that the end user valued.

Using average rating as the recommendation algorithm will not be a good measure for generating meaningful recommendations. Cosine similarities and sentiments allow us to make more personalized recommendations.