University of New Haven
TAGLIATELA COLLEGE OF ENGINEERING
Electrical & Computer Engineering and Computer Science

Distributed and Scalable Data Engineering (DSCI-6007-01)

Restaurant Recommendation System

Group #2 :

Venkata Masthan Sai Nishant K
Tulika K
Saicharan A

Abstract:

Recommendation systems provide consumers with individualized, relevant recommendations and have been utilized in a variety of environments, including shopping, movies, and so on. Currently, Yelp, the world's largest publisher of local business reviews, does not offer recommendations. Instead, users must filter, sort, and read reviews to see if a company can give them what they require. A personalized recommendation system would improve the user experience by encouraging users to review and rate more restaurants in exchange for better restaurant recommendations, which will provide Yelp with additional data to improve the recommendation system.

We created a restaurant recommendation system for individuals and groups based on Yelp business and Yelp reviews datasets. We developed a matrix factorization with a collaborative filtering model with features for each of the 10k Yelp users. Food preferences and dietary restrictions, as well as cuisine style, services offered, ambiance, noise level, and average rating, and so on.

As an alternative to conventional group recommendation systems that select the most frequently recommended restaurant among individual users, this method selects the most generally recommended restaurant across a group of users.

Link to Github is given below:
https://github.com/tulkot/Restaurant_Recommendation_System

Theory:

When we consider factorization, we can compare it to the concept of three times nine equaling twenty-seven. Although twenty-seven is a large number, we can represent it as a product of two small numbers, three and nine, allowing us to break down a large number into two small ones. Matrix Factorization is based

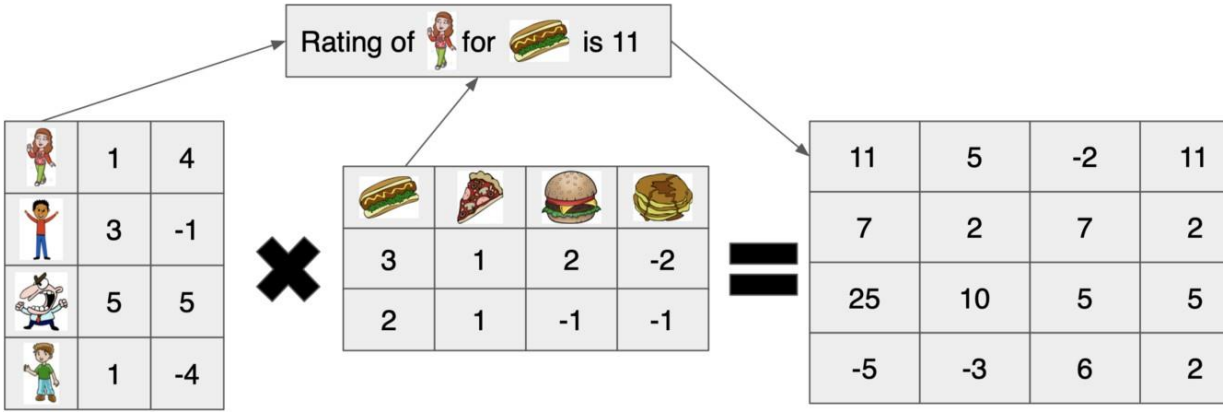
On a similar notion. Recommendation systems are mainly classified into two categories.

- Content-Based System

- Collaborative Filtering (CF) System

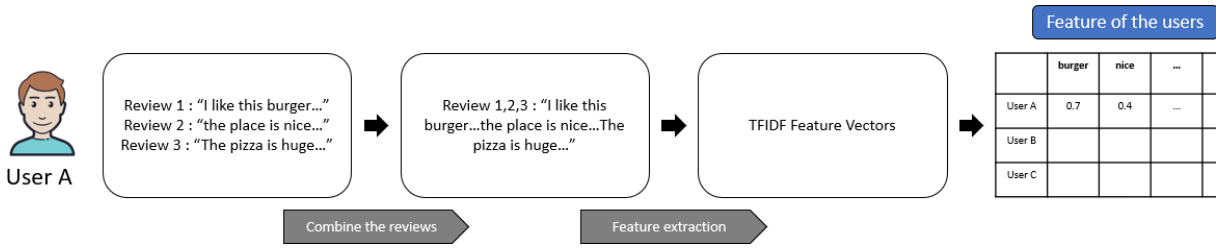
Entirely based on past interactions between the customer and the product. Thus, the primary input for Collaborative filtering (CF) is historical data relating to all transactions between user interactions with targeted products. Matrix is used to store data, where rows are customers and columns, are products. This technique is purely dependent on historical data and nothing more, such as current trends and cultures. At the deepest level, CF can be classified into memory-based and pattern-based methods. The memory-based method is the simplest method that tracks only historical data with a simple distance

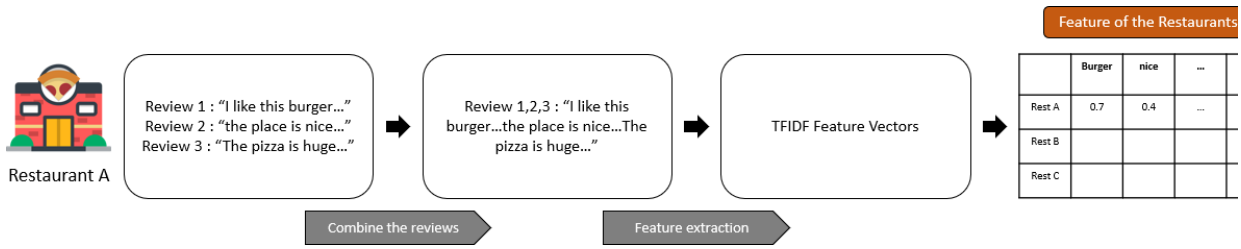
measurement. The model-driven approach used a model to fit possible outcomes.



A recommendation system is a data filtering system that attempts to forecast the user's rating of an item (in this case a restaurant). We can divide the huge matrix of user and item ratings into two smaller user-feature and item-feature matrixes. For instance, if user A enjoys hotdogs but despises pizza, and restaurant P serves excellent hotdogs, we multiply the matrices using the dot product, and the result is the ratings (in the example above will be 11).

In order to use the text for the matrix factorization recommendation system, we will follow the architecture below to extract the features from the review text.





For each user, combine all reviews into one paragraph. After combining everything, apply the TFIDF vectorizer to extract features from the text. Each restaurant should take a similar approach and specify max_feature to match the dimensions of the matrix.

Methodology:

This project is based on the public opinion of restaurants. The datasets was extracted from the Kaggle (provided the link below). This was acquired on July, 2021. The extracted reviews and business datasets contained 10,000 unique comments/compliments from multiple users around the world and 10 columns each.

1. https://www.kaggle.com/datasets/omkarsabnis/yelp-reviews-dataset
2. https://www.kaggle.com/datasets/tanyadixit/yelp-business-dataset

Steps Involved:

- Get data on restaurants and reviews of those restaurants.
- Identify rows by customer's ids and restaurant id's.
- Preprocess the text of the reviews - remove stop words from the stop words library and normalize all text to text without punctuation.
- Splitting datasets into test and train datasets in the 85:15 ratio.
- Process data to train and test datasets by customer's ids and theirreviews, and restaurants ids and their reviews.
- Tokenize the text of the reviews using TFID Vectorizer for easier NLP processing.
- Create a value matrix of ratings, and convert it to NumPy arrays.

- Matrix Factorization.
- Predictions - we input what we desire to eat to the recommendation system, and it gives us the top 5 recommendations based on what we want to eat.
- Visualizations – review count by star ratings, review text length by star ratings, the number of reviews per star rating, reviews word cloud for the 1-star rating, and a 5-star rating.

Results:

1. Checking the distribution of data

In [5]:
```
yelp.head()
```

Out[5]:

	business_id	date	review_id	stars	text	type	user_id	cool
0	9yKzy9PApeiPPOUJEtnvkg	2011-01-26	fWKvX83p0-ka4JS3dc6E5A	5	My wife took me here on my birthday for breakf...	review	rLtl8ZkDX5vH5nAx9C3q5Q	2
1	ZRJwVLyzEJq1VAihDhYiow	2011-07-27	IjZ33sJrzXqU-0X6U8NwyA	5	I have no idea why some people give bad review...	review	0a2KyEL0d3Yb1V6aivbIuQ	0
2	6oRAC4uyJCsJl1X0WZpVSA	2012-06-14	IESLBzqUCLdSzSqm0eCSxQ	4	love the gyro plate. Rice is so good and I als...	review	0hT2KtfLiobPvh6cDC8JQg	0
3	_1QQZuf4zZOyFCvXc0o6Vg	2010-05-27	G-WvGaISbqqaMHlNnByodA	5	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	review	uZetl9T0NcROGOyFfughhg	1
4	6ozycU1RpktNG2-1BroVtw	2012-01-05	1uJFq2r5QfJG_6ExMRCaGw	5	General Manager Scott Petello is a good egg!!!...	review	vYmM4KTsC8ZfQBg-j5MWkw	0

In [3]:
```
businesses.head()
```

Out[3]:

	business_id	full_address	hours	open	categories	city	review_count	name	neighborhoods	longitude	state	stars	latitude	att
0	O_X3PGhk3Y5JWVi866qlJg	1501 W Bell Rd\nPhoenix, AZ 85023	{'Monday': {'close': '18:00', 'open': '11:00'}...	True	[Active Life, Arts & Entertainment, Stadiums &...	Phoenix	29	Turf Paradise Race Course	[]	-112.092329	AZ	4.0	33.638573	out
1	QbrM7wqtmoNncqjc6GtFaQ	18501 N 83rd Avenue\nGlendale, AZ 85308	{}	True	[Tires, Automotive, Fashion, Shopping, Departm...	Glendale	3	Sam's Club Members Only	[]	-112.234755	AZ	3.5	33.648545	{'P {'g
2	7IbvsGKzhjuX3oJtaXJvOg	5000 S Arizona Mills Cir\nSte 590\nTempe, AZ 8...	{'Monday': {'close': '21:00', 'open': '10:00'}...	True	[Women's Clothing, Men's Clothing, Fashion, Sh...	Tempe	7	Forever 21	[]	-111.964485	AZ	3.5	33.383123	{'P {'g
3	gjxoKVsRJwEoa8zd9XxlAw	912 W Sycamore Pl\nChandler, AZ 85225	{'Monday': {'close': '19:00', 'open':	True	[Pet Services, Pet Boarding/Pet Sitting, Pets]	Chandler	4	Loving Hands Pet Care	[]	-111.857818	AZ	5.0	33.356472	

2. Cleaning the text for processing: Text preprocessing - removing stop words, and removing punctuations as well

In [12]:
```
stop = []
for word in stopwords.words('english'):
    s = [char for char in word if char not in string.punctuation]
    stop.append(''.join(s))
```

In [13]:
```
def text_process(mess):
    nopunc = [char for char in mess if char not in string.punctuation]

    nopunc = ''.join(nopunc)

    return " ".join([word for word in nopunc.split() if word.lower() not in stop])
```

In [14]:
```
reviews_data['text'] = reviews_data['text'].apply(text_process)
```

```
C:\Users\tulik\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  """Entry point for launching an IPython kernel.
```

3. Splitting the datasets into training and testing sets :

Training and Testing datasets: 85:15

In [15]:
```
vld_size=0.15
X_train, X_valid, y_train, y_valid = train_test_split(reviews_data['text'], reviews['business_id'], test_size = vld_size)
```

In [16]:
```
userid = reviews_data[['user_id','text']]
businessid = reviews_data[['business_id', 'text']]
```

In [17]:
```
userid.head()
```

Out[17]:

	user_id	text
0	rLtl8ZkDX5vH5nAx9C3q5Q	wife took birthday breakfast excellent weather...
1	0a2KyEL0d3Yb1V6aivbIuQ	idea people give bad reviews place goes show p...
2	0hT2KtfLiobPvh6cDC8JQg	love gyro plate Rice good also dig candy selec...
3	uZetl9T0NcROGOyFfughhg	Rosie Dakota LOVE Chaparral Dog Park convenien...
4	vYmM4KTsC8ZfQBg-j5MWkw	General Manager Scott Petello good egg go deta...

Sample reviews in the userid dataframe

4. Feature Extraction on the reviews:

In [23]:
```
userid_vec = TfidfVectorizer(tokenizer = WordPunctTokenizer().tokenize, max_features=5000)
userid_vector = userid_vec.fit_transform(userid['text'])
userid_vector.shape
```

Out[23]:
```
(6403, 5000)
```

In [24]:
```
userid_vector
```

Out[24]:
```
<6403x5000 sparse matrix of type '<class 'numpy.float64'>'
        with 457919 stored elements in Compressed Sparse Row format>
```

In [25]:
```
#Business id vectorizer
businessid_vec = TfidfVectorizer(tokenizer = WordPunctTokenizer().tokenize, max_features=5000)
businessid_vector = businessid_vec.fit_transform(businessid['text'])
businessid_vector.shape
```

Out[25]:
```
(4174, 5000)
```

In [26]:
```
userid_rating_matrix = pd.pivot_table(reviews_data, values='stars', index=['user_id'], columns=['business_id'])
userid_rating_matrix.shape
```

Out[26]:
```
(6403, 4174)
```

5. Matrix Vectorization Algorithm:

Matrix Vectorization Algorithm

In [29]:
```
def matrix_factorization(R, P, Q, steps=3, gamma=0.001,lamda=0.02):
    for step in range(steps):
        for i in R.index:
            for j in R.columns:
                if R.loc[i,j]>0:
                    eij=R.loc[i,j]-np.dot(P.loc[i],Q.loc[j])
                    P.loc[i]=P.loc[i]+gamma*(eij*Q.loc[j]-lamda*P.loc[i])
                    Q.loc[j]=Q.loc[j]+gamma*(eij*P.loc[i]-lamda*Q.loc[j])
        e=0
        for i in R.index:
            for j in R.columns:
                if R.loc[i,j]>0:
                    e= e + pow(R.loc[i,j]-np.dot(P.loc[i],Q.loc[j]),2)+lamda*(pow(np.linalg.norm(P.loc[i]),2)+pow(np.linalg.norm(Q.loc
        if e<0.001:
            break

    return P,Q
```

In [30]:
```
%%time
P, Q = matrix_factorization(userid_rating_matrix, P, Q, steps=3, gamma=0.001,lamda=0.02)
```

```
Wall time: 58min 58s
```

6. Test Input to get top restaurant recommendations - we are giving the following input to the system:

“I want to have eggs for breakfast “

In [37]:

```
test_input = "i want to have eggs for breakfast"
test_df= pd.DataFrame([test_input], columns=['text'])
test_df['text'] = test_df['text'].apply(text_process)
test_vectors = userid_vec.transform(test_df['text'])
test_v_df = pd.DataFrame(test_vectors.toarray(), index=test_df.index, columns=userid_vec.get_feature_names())

predictItemRating=pd.DataFrame(np.dot(test_v_df.loc[0],Q.T),index=Q.index,columns=['Rating'])
topRecommendations=pd.DataFrame.sort_values(predictItemRating,['Rating'],ascending=[0])[:7]

for i in topRecommendations.index:
    print(businesses[businesses['business_id']==i]['name'].iloc[0])
    print(businesses[businesses['business_id']==i]['categories'].iloc[0])
    print('Star rating: ' + str(businesses[businesses['business_id']==i]['stars'].iloc[0])+ '; Number of reviews: '+str(businesses[businesse
    print('')
```

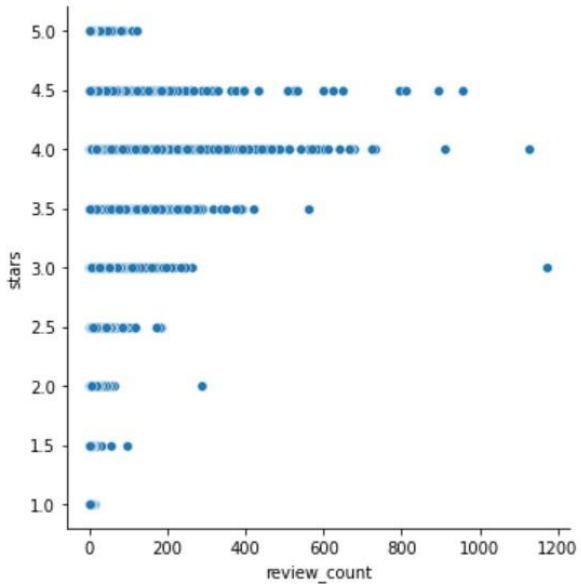
```
The Good Egg
['Breakfast & Brunch', 'Restaurants']
Star rating: 4.0; Number of reviews: 45

Breakfast Club
['Breakfast & Brunch', 'American (Traditional)', 'Restaurants']
Star rating: 3.5; Number of reviews: 382

Crackers & Co Cafe
['Breakfast & Brunch', 'American (Traditional)', 'Restaurants']
Star rating: 4.0; Number of reviews: 155

Butterfield's Pancake House
['Breakfast & Brunch', 'Restaurants']
Star rating: 4.0; Number of reviews: 236
```

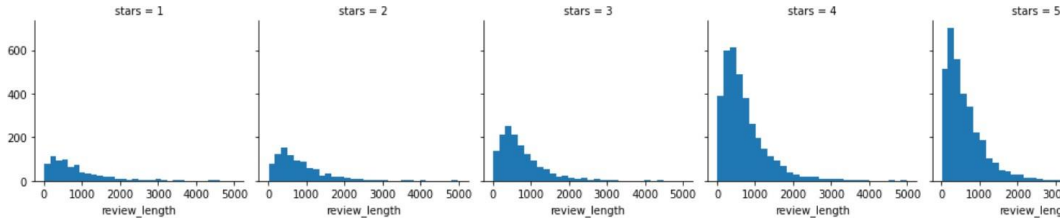
```
Verde
['Mexican', 'Restaurants']
Star rating: 4.0; Number of reviews: 59

The Good Egg
['Breakfast & Brunch', 'Restaurants']
Star rating: 3.5; Number of reviews: 26
```

7. Visualizing the datasets:

In [38]:
```
sns.relplot(x="review_count", y="stars", data=businesses)
```

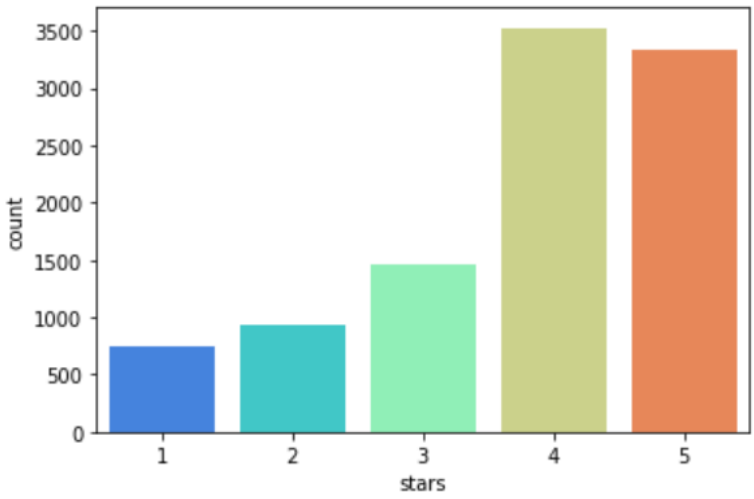
Out[38]:
```
<seaborn.axisgrid.FacetGrid at 0x2114f1f0b88>
```





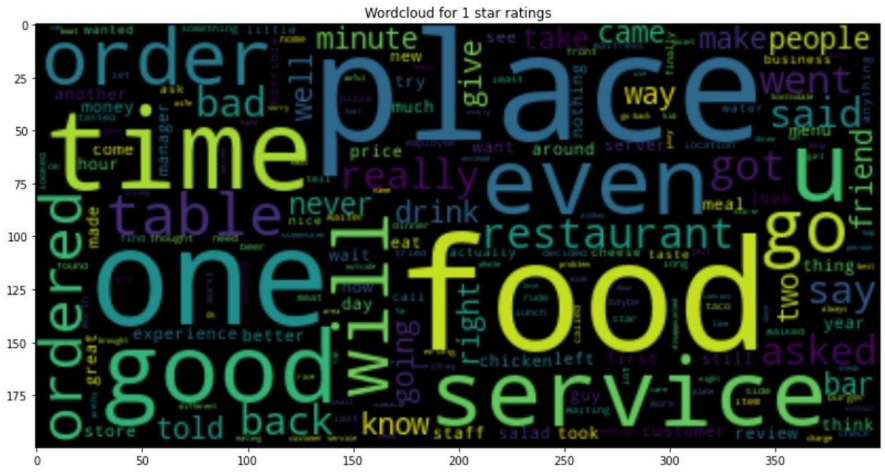
```
In [40]: sns.countplot(x='stars',data= reviews, palette='rainbow')
```

```
Out[40]: <AxesSubplot:xlabel='stars', ylabel='count'>
```



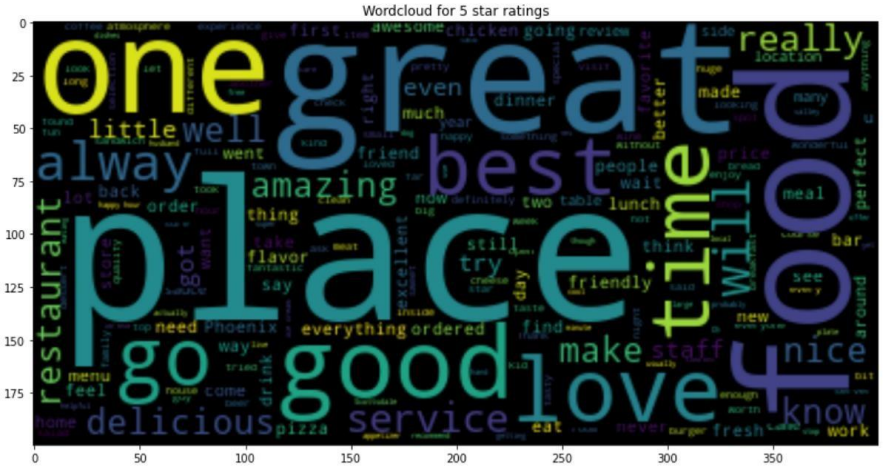
Below was the word cloud obtained. A word cloud (also known as a tag cloud) is a visual representation of words. Here, the frequency of words is determined by their size in the cloud. The bigger ones are the more frequently used words and vice versa. As we can see, food was the most talked about topic when these tweets were collected.

```
Out[42]: Text(0.5, 1.0, 'Wordcloud for 1 star ratings')
```



Word cloud for 5-star ratings:

Out[43]: Text(0.5, 1.0, 'Wordcloud for 5 star ratings')



Conclusion:

In this project, we implemented matrix factorization using collaborative filtering on user reviews for datasets that are collected from yelp. So we tried tfidf as matrix factorization technique and we successfully retrieved restaurant recommendations for a test user phrase.