RESTAURANT RECOMMENDATION SYSTEM

SUBMITTED BY:
RIYA GARG - MT22058
THANMAYEE MATHA - MT22084
KOMALJYOT KAUR - MT22105
ABHINAV GARG - MT22002
SAKSHAM NAUTIYAL - MT22061

Updated Problem Statement:

Recommendation systems are widely used today. Almost every major digital business has used them in one way or another. For example, YouTube uses them to choose which video to play next on autoplay, while Amazon uses them to propose things to customers. Design and implement a restaurant recommendation system by considering the user's past dining history and ratings, as well as the features of restaurants, such as location, cuisine, price range, ambiance, and ratings.
The system uses hybrid filtering which combines the results of collaborative filtering to identify similar users based on their past dining history and ratings, recommend restaurants that the user might like and content based filtering techniques to recommend restaurants that match the user's preferences based on the features of the restaurant such as location, cuisine, price range, ambiance, and ratings to provide personalized restaurant recommendations to users. The system also provides a user-friendly interface that allows users to easily search for and discover new restaurants based on their preferences and recommendations.
The system is developed based on the user reviews in Yelp Dataset which consists of various information about the restaurants including their menus, ratings, and reviews, as well as user preferences, including their cuisine preferences, price range, and other factors.

Literature Review:

By using the users' current geographical position, Md. Ahsan Habib et al. present a novel location, preference, and time based restaurant recommendation system. The method evaluates user check-in accounts to investigate his visiting habits, meal preferences, and popular eateries. Four main factors used to calculate recommendation scores are User preference score, Restaurants' separation, the hour of the day , and the popularity ratings of the eatery[1].

In order to predict the customer satisfaction rating, Nanthaphat Koetphrom et al. present a method based on real data (connected to customer/restaurant features) and similarity in consumer preferences. The three filtering strategies being suggested are collaborative, content-based, and hybrid. The results show that the collaborative filtering approach uses cluster-based methodology to regulate information among its peers.[2]

The purpose of the study, according to Khushbu Jalan et al., is to recommend inn names to the explorer based on their interests and inclinations, using the feedback from other explorers and the rating as an incentive to increase prediction accuracy. The setting-aware cross-breed methodology is employed where CF method aggregates wistful research and provides personalized inn ideas. The use of a setting-based procedure then improves the proposal's results even more.[3]

The adaptive climate is used in the café recommender framework developed by Jun Zeng et al. The framework eventually gives suggestion outcomes based on the model after first building an inclination model for customers

based on client/café area nuances and café visits. The contextual study also showed that the BMCS and BWCS-based café recommender system could effectively use the client's tendency.[4]

Ling Li et al. suggest three changes to the conventional UCF algorithm. The UCF algorithms' accuracy was quite poor because there were numerous factors that could affect a user's preference for a restaurant. Last but not least, actual private information of registered internet users is being used to gauge the similarity connected to user features.The results make it abundantly evident that the ACFmodified algorithm improves the accuracy of similarity computation and provides the user with a highly accurate restaurant suggestion. [5]

The study by Michelle Renee D. Ching et al., helps the restaurants to improve their customer satisfaction through analyzing the text reviews of the customers by over siving the business aspect of the reviews. This was done with the use of Aspect Based Sentiment Analysis(ABSA) endpoint of the AYLEIN Text Analysis API with which we can perform opinion mining.Time series forecasting using linear regression for one year with the use of the Weka machine learning workbench will be performed and conduct a linear regression with the one-year predicted data to understand its pattern to extract valuable information that will help in recommendation of business strategies.[6]

Boya Yu et al., proposed a recommendation system which uses the support vector machine(SVM) model to decipher the sentiment tendency of each review from word frequency.Word scores generated from the SVM models are further processed into a polarity index indicating the significance of each word for special types of restaurant.This method includes collecting keywords from different cuisines, our model can also be used for automatically generating ratings for tips (short reviews that are not accompanied with ratings) on Yelp by assigning weights to tips using the sentiment score of words and hence gives more reasonable overall ratings for restaurants.[7]

A multilingual recommender system based on sentiment analysis to help Algerian users decide on products, restaurants, movies and other services using online product reviews is proposed by Amel ZIANI et al., combines both recommendation system and sentiment analysis in order to generate the most accurate recommendations for users. This system detects the opinions polarity score using the semi supervised SVM.The results analysis evaluation provides interesting findings on the impact of integrating sentiment analysis into a recommendation technique based on collaborative filtering with very high precision and 100% recall.[8]

Dataset:

The data we used comes from the following link: http://www.yelp.com/dataset_challenge. This data has been made available by Yelp for the purpose of the Yelp Dataset Challenge. In particular, the challenge dataset contains the following data:

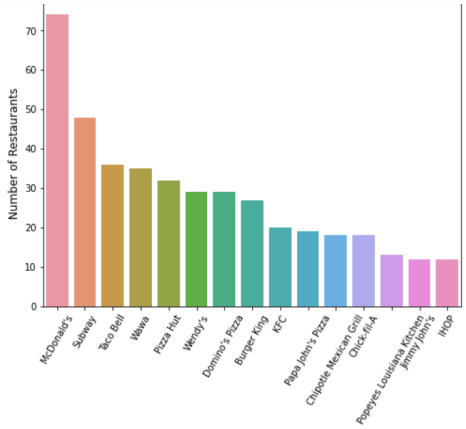
- 1.6M reviews and 500K tips by 366K users for 61K businesses
- 481K business attributes, e.g., hours, parking availability, ambience.
- Social network of 366K users for a total of 2.9M social edges.
- Aggregated check-ins over time for each of the 61K businesses

Data Preprocessing:

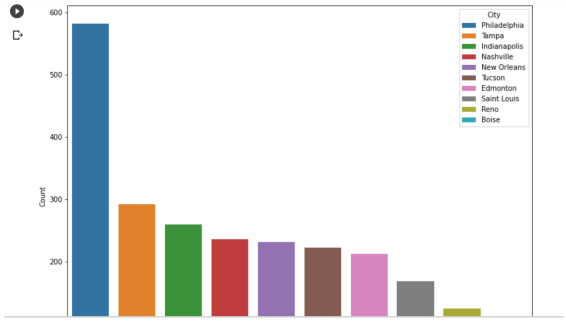
Data originally is in json format . So we first converted all files into the csv format.

Exploratory Data Analysis:

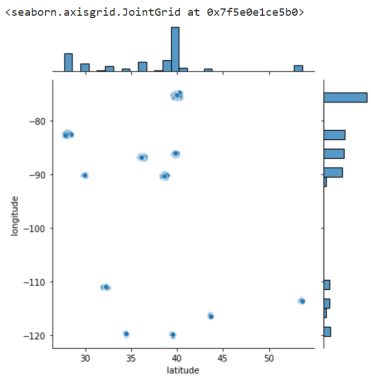
- Firstly we found 10 most popular restaurants in the dataset.



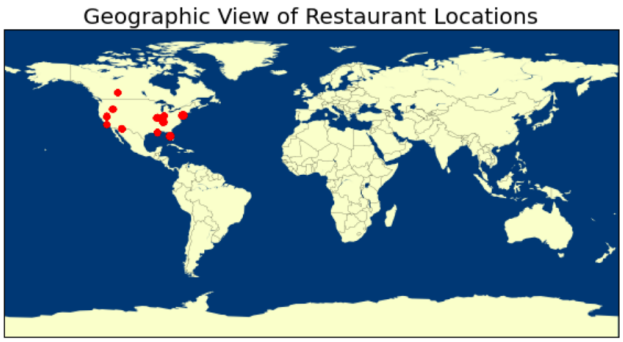
- We explored top 10 cities having maximum restaurants



- We see that locations of businesses are concentrated in clusters. These clusters must be big cities.

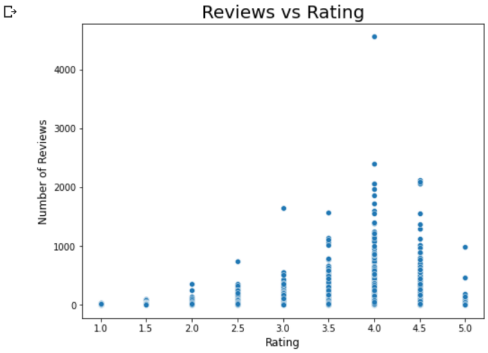


- We analyzed that our data has businesses from certain cities of U.S. and not all over U.S.

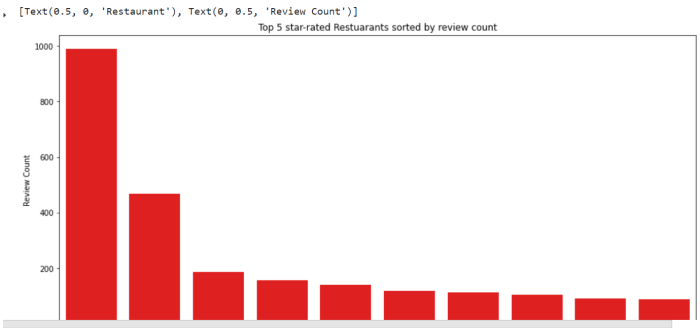


- Then we explored most reviewed food categories in business data

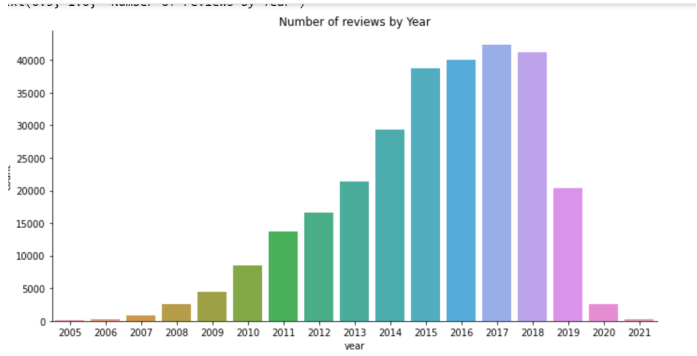
- Checked how rating and reviews are related to each other as these are important factors for restaurant recommendation.We can see that as the rating increases from 1.0 to 4.0, the number of reviews tends to increases as well. However, as rating increases further, especially from 4.5 to 5.0, the number of review shrinks.

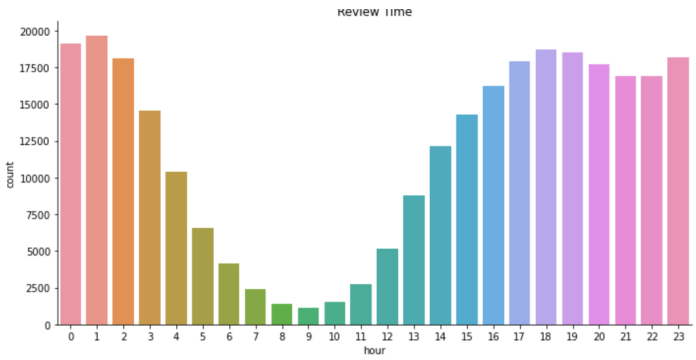


- We also found the top 10 5 star restaurants sorted by review count



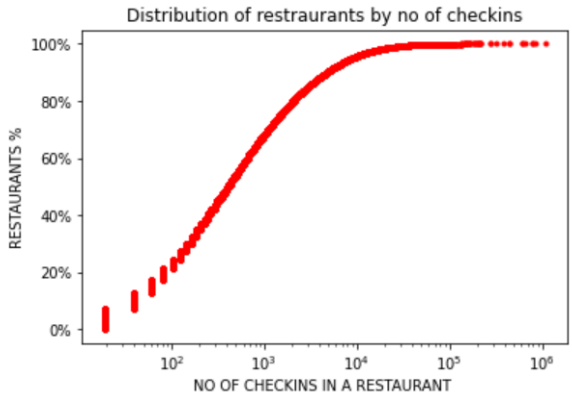
- Restaurants with TakeOut, AcceptCreditCard, GoodForKids, Reservation, GoodForGroups, BusinessParking, HasTV, Alcohol, BikeParking, Delivery,, Attire are more popular
- We saw correlations roughly at the center of heatmap in business data
- We also analyzed the number of reviews in a year and the most used time to give reviews.





- We found that no reviews recorded are minimum in the morning.
- Overtime users become less harsh reviewers.
- Half of the restaurants have less than 20 check ins, even less than the reviews, indicating that check in is not a widely used feature on Yelp.

```
ax.set_yticklabels(['{:.0%}'.format(x) for x in ax.get_yticks()])
```



- The popularity of reviews shows a steady upward trend since the beginning in 2004 with seasonal fluctuations, whereas the popularity of tips (# of tips) increases in the first four years after its introduction (2009-2013) and slowly dives down thereafter. Therefore tips are not as popular as reviews.

Baseline Results:

- ☐ We build a recommender system on item similarity based collaborative filtering. It recommends restaurants like the given restaurant id. For this we have used a KNN based model.
  Model to recommend the rating of the user.
- ☐ In this the dataset contains the user id, buisness_id ,and stars from the final merged dataset .Then we split the data into train ,test and validation dataset.
  The baseline model has the average mean ratings of all the users.

```
Baseline MSE using mean rating:

        Train Data: 1.8027,
        Val   Data: 1.8055,
        Test  Data: 1.8005
```

Updated baseline results:

1. Recommendation based on restaurant location and feature based keyword filtering:(non- personalization):It is designed to filter using relevant keywords. The restaurant location-based keywords (zip code, longitude, latitude) and restaurant feature-based keywords should both be supported by the keyword-search module (cuisine, style, price). Prior to rating the restaurant directory using the user-selected ranking metrics, the restaurant directory will first be filtered by keywords.The user-selected ranking criteria are used to order

the restaurant catalogue after it has been initially filtered by all the keywords showing the user's preferences. Finally, the user selects n, and the top-n restaurants from the rated list are returned. Both the restaurant's original average star rating and the recently added weighted rating have been applied for the ranking criteria and are available to users via the module interface.

```
        """
        # re-initiate the following variables every time the module is called so that the recommendation starts fresh
        self.recomm = df # start with the desired restaurant catalog
        self.recomm['distance_to_interest'] = np.nan # reset the distance between each restaurant and the location of interest
        self.column_to_display = ['state','city','name','address','attributes.RestaurantsPriceRange2','cuisine','style','review_count','stars','adj

        # assign variables based on user's keyword inputs
        self.zipcode = zipcode
        self.city = city
        self.state = state
        self.max_distance = max_distance
        self.cuisine = cuisine
        self.style = style
        self.price = price

        # filter by restaurant location
        if (self.zipcode != None) or (self.city != None) or (self.state != None):
            if (self.zipcode != None) or (self.city != None): # use zipcode and/or city whenever available
                self._filter_by_location()
            else: # filter by state if state is the only location information available
                self._filter_by_state()
            if len(self.recomm) == 0:
                print("no restaurant found for the matching location of interest.")
                return None

          # filter by restaurant 'cuisine'
        if self.cuisine != None:
            self._filter_by_cuisine()
            if len(self.recomm) == 0:
                print("no restaurant found for the matching cuisine of {}".format(self.cuisine))
                return None

        # filter by restaurant 'style'
        if self.style != None:
            self._filter_by_style()
```

```
# test 11: use the original average rating and return top 10 recommendations
print("------\nresult from test11 (top 10 recommendations ranked by original average rating): ")
kw2 = Recommender(n=10, original_score=True)
kw2.keyword(city='Phoenix', zipcode='85023',cuisine='barbeque', style='restaurants');
```

```
result from test9 (a combination of price range, cuisine and style):
Below is a list of the top 3 recommended restaurants for you:
       state         city                          name                 address  \
33182     FL        Tampa             BJ's Alabama BBQ  3423 S Dale Mabry Hwy
110999    TN  Nolensville  Outlanders Southern Chicken    7215 Nolensville Rd
39116     FL  Port Richey         Robert's Smokin' BBQ     7810 US 19, Bldg B

       attributes.RestaurantsPriceRange2  \
33182                                  1
110999                                 1
39116                                  1

                                        cuisine  \
33182                                  barbeque
110999  barbeque, chicken wings, american (traditional)
39116                        barbeque, southern

                      style  review_count  stars  adjusted_score
33182            restaurants           200    4.5        4.426094
110999           restaurants           188    4.5        4.421871
39116   restaurants, caterers           187    4.5        4.421497
------
```

2.For collaborative filtering based recommendation:It supports personalized restaurant recommendation given the unique user_id.The distinct user id provides customized restaurant recommendations. The user-unrated restaurants from the catalog are ranked by the ratings anticipated by the model and returned as personalized suggestions. The customization is computed based on the user's and all other users' rating histories of all Yelp establishments.

Matrix Factorization algorithms implemented till now:

- SVD (singular value decomposition): Using the singular value decomposition technique, the user x business matrix is factored into the user latent matrix and the business latent matrix.

```
# simple SVD model
svd = SVD(n_factors=20, n_epochs = 30, biased=False) # initiate a SVD algorithm object
svd.fit(training) # training on the trainset
pred_svd = svd.test(testing) # predict ratings for the testset
accuracy.rmse(pred_svd) # compute RMSE score

# user and item matrix with latent features
mean = svd.trainset.global_mean # global mean rating of the trainset
user_latent, item_latent = svd.pu, svd.qi
print(user_latent.shape, item_latent.shape)
```

```
RMSE: 2.0039
(177878, 20) (4921, 20)
CPU times: user 7.78 s, sys: 60 ms, total: 7.84 s
Wall time: 14.7 s
```

- SVD with bias:

```
[ ] %%time
    # SVD bias model with defaults
    svd_bias = SVD(n_factors=20, n_epochs = 30, biased=True) # initiate a SVD algorithm object with the bias terms
    svd_bias.fit(training) # training on the trainset
    pred_svd_bias = svd_bias.test(testing) # predict ratings for the testset
    accuracy.rmse(pred_svd_bias) # compute RMSE score

    # extract useful information from the fitted model
    mean = svd_bias.trainset.global_mean # global mean rating of the trainset
    # user and item matrix with latent features
    user_latent, item_latent = svd_bias.pu, svd_bias.qi
    print(user_latent.shape, item_latent.shape)
    # user and item bias vector
    user_bias, item_bias = svd_bias.bu, svd_bias.bi
    print(user_bias.shape, item_bias.shape)
```

```
RMSE: 1.2305
(177878, 20) (4921, 20)
(177878,) (4921,)
CPU times: user 4.16 s, sys: 33 ms, total: 4.2 s
Wall time: 4.2 s
```

GRID Search optimization:

```
# cross validation to optimize parameters of SVD with bias
param_grid = {'n_factors': [10,20,30,50], 'n_epochs': [50,100,200], 'lr_all': [0.005],'reg_all': [0.05], 'biased': [True]}
svd_gs = GridSearchCV(SVD, param_grid, measures=['rmse'], cv=KFold(3, random_state=42), joblib_verbose=2)
svd_gs.fit(data_train) # gridsesarch optimization on the trainset

# best RMSE score
print(svd_gs.best_score)
# combination of parameters that gave the best RMSE score
print(svd_gs.best_params)

# update SVD bias model with optimized parameters
svd_gs_best = svd_gs.best_estimator['rmse']  # algorithm with the optimized parameters
svd_gs_best.fit(training)
pred_svd_gs_best = svd_gs_best.test(testing)
accuracy.rmse(pred_svd_gs_best)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    7.8s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:  9.7min finished
{'rmse': 1.2357453835714054}
{'rmse': {'n_factors': 10, 'n_epochs': 50, 'lr_all': 0.005, 'reg_all': 0.05, 'biased': True}}
RMSE: 1.2307
```

- NMF (non-negative matrix factorization): Using a non-negative matrix factorization technique, the user x business matrix is factorised into a non-negative user latent matrix and a non-negative business latent matrix. In contrast to SVD, the user and business latent matrices' fitted values are all non-negative.

```
from surprise import NMF

n_factors = 20
model = NMF(n_factors=n_factors, biased=False)
```

```
model.fit(training) # training on the trainset
pred_nmf = model.test(testing) # predict ratings for the testset
accuracy.rmse(pred_nmf)

# user and item matrix with latent features
mean = model.trainset.global_mean # global mean rating of the trainset
user_latent, item_latent = model.pu, model.qi
print(user_latent.shape, item_latent.shape)
```

```
RMSE: 1.4150
(177878, 20) (4921, 20)
```

NMF WITH BIAS

```
[ ]  n_factors = 20
     nmf_b = NMF(n_factors=n_factors, biased=True)
```

```
[ ]  nmf_b.fit(training) # training on the trainset
     pred_nmf_bias = nmf_b.test(testing) # predict ratings for the testset
     accuracy.rmse(pred_nmf_bias) # compute RMSE score

     # extract useful information from the fitted model
     mean = nmf_b.trainset.global_mean # global mean rating of the trainset
     # user and item matrix with latent features
     user_latent, item_latent = nmf_b.pu, nmf_b.qi
     print(user_latent.shape, item_latent.shape)
     # user and item bias vector
     user_bias, item_bias = nmf_b.bu, nmf_b.bi
     print(user_bias.shape, item_bias.shape)
```

```
RMSE: 1.5593
(177878, 20) (4921, 20)
(177878,) (4921,)
```

```
[ ]  # initiate a NMF algorithm and fit to the trainset
     model = NMF(n_components=20, tol=5e-5)
     W = model.fit_transform(trainset_m) # W is the user x latent feature matrix
     H = model.components_ # H is the latent feature x item matrix
     print(W.shape, H.shape)

     # test on the testset
     rmse, pred = compute_performance(testset, W, H, train_map_user, train_map_business)
     print(rmse) # RMSE on the testset
```

```
(177878, 20) (20, 4921)
2.795163711923066
```

Grid search optimization of NMF model:

```
[ ]
     from surprise import NMF
     # cross validation to optimize parameters of NMF with no bias
     param_grid = {'n_factors': [10,20,30,50], 'n_epochs': [50],'biased': [False]}
     nmf_gs = GridSearchCV(NMF, param_grid, measures=['rmse'], cv=KFold(3, random_state=42), joblib_verbose=2)
     nmf_gs.fit(data_train) # gridsesarch optimization on the trainset, need to feed in a Dataset object not a trainset object

     # best RMSE score
     print(nmf_gs.best_score)
     # combination of parameters that gave the best RMSE score
     print(nmf_gs.best_params)

     # update simple NMF model with optimized parameters
     nmf_gs_best = nmf_gs.best_estimator['rmse']  # algorithm with the optimized parameters
     nmf_gs_best.fit(training)
     pred_nmf_gs_best = nmf_gs_best.test(testing)
     accuracy.rmse(pred_nmf_gs_best)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   14.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:  4.9min finished
{'rmse': 1.3957781145688253}
{'rmse': {'n_factors': 30, 'n_epochs': 50, 'biased': False}}
RMSE: 1.4057
1.4056699544742501
```

Proposed Method:

We are developing a hybrid restaurant recommendation system using yelp dataset. We plan to develop a hybrid recommendation system formed by non- personalization recommendation based on restaurant location and feature based keyword filtering, collaborative filtering based recommendation and personalized restaurant content based filtering recommendation module which is implemented using a user-friendly interface created to integrate the submodules, gather user interests and navigate users through the hybrid recommendation engine.

The non- personalization recommendation based on restaurant location and feature based keyword filtering facilitates combining restaurant feature-based (cuisine, style, pricing) and location-based (zip code, city, state) keyword filtering of the restaurant catalog. The filtered catalog is ranked according to the user's preferred ranking criteria to produce the tailored suggestions.

In the collaborative filtering based recommendation module ,personalized restaurant recommendations are supported given the distinct user id. The user-unrated restaurants from the catalog are ranked by the ratings anticipated by the model and returned as personalized suggestions. The personalization is computed based on the user's and all other users' rating histories of all Yelp establishments.

In the personalized restaurant content-based recommender module, personalized restaurant recommendations are supported given the distinct user id. User-unrated restaurants from the catalog are ranked by similarity score and returned as personalized recommendations. The personalization is computed based on the similarity between the user's preference indicated by historical ratings and all restaurants' features extracted from a rich set of Yelp restaurant review texts.

We also have a user-friendly interface that supports flexible navigation among the available recommender modules at user's choice and options to display the recommendation results by keywords and/or display the desired number of recommendations.

We will try to increase the accuracy of the recommendation system using sentimental analysis to give a better user personalized recommendation according to their previous choice of preferences. For this sentimental Analysis we use algorithms like BERT to get the best accuracy of the restaurant recommendation system.

Evaluation Metric

- Recommendation based on restaurant location and feature based keyword filtering:(non-personalization):Test results demonstrate that the recommended restaurants are the only ones that match the user's keyword combination and are ranked according to the appropriate interest scores.
- For collaborative filtering based recommendation:

1. RMSE:We used the RMSE as an evaluation metric to evaluate the performance of our recommendation system. RMSE is a measure of the difference between the predicted ratings and the actual ratings of the restaurants. It is a widely used metric in recommendation systems as it gives a measure of the accuracy of the predicted ratings. The formula for RMSE is given as:
   sqrt(sum((predicted_rating - actual_rating)^2) / n)

2. NDCG(Normalized Discounted Cumulative gain):nDCG is a widely used metric in recommendation systems that takes into account the relevance and ranking of the recommended items. It measures the quality of the ranking of recommended items, where a higher score indicates better performance. The formula for nDCG is given as:nDCG@k = DCG@k / IDCG@k
   Where k: The number of recommended items to consider
   DCG@k: Discounted Cumulative Gain at k, which is the sum of the relevance scores of the top k recommended items
   IDCG@k: Ideal Discounted Cumulative Gain at k, which is the DCG@k when the recommended items are in perfect order based on relevance
   The model used to predict ratings for test sets with new users and businesses as well as testsets without new users or businesses ('testset 2').
   Following the computation of NDCG@top10 and NDCG@5, the recommendation ranking is constructed for each user in the testset based on the projected ratings in decreasing order.

3. Rmse achieved by models after baseline results:
   Simple SVD: -2.0039
   SVD with bias -1.2305
   Grid search (SVD):1.23
   NMF without bias -1.41
   NMF with bias-1.55
   NMF using scikit learn-2.79
   NMF model(grid search optimization):1.4037
4. Average nDCG achieved by our best model at nDCG@5 AND nDCG@10 is 0.96 and 0.94 respectively.

References:

[1] Ahsan Habib, Abdur Rakib, and Muhammad Abul Hasan, “Location, Time, and Preference Aware Restaurant Recommendation Method”, IEEE, 19th International Conference on Computer and Information Technology, pp. 315-319, 2016.

[2] Nanthaphat Koetphrom, Panachai Charusangvittaya, Daricha Sutivong, “Comparing Filtering Techniques in Restaurant Recommendation System”, IEEE, pp. 46-51, 2018

[3] Khushbu Jalan, Kiran Gawande, “Context-Aware Hotel Recommendation System based on Hybrid Approach to Mitigate ColdStart-Problem”, IEEE, Communication, Data Analytics and Soft Computing, pp. 2364-2369, 2017.

[4] Jun Zeng, Feng Li, Haiyang Liu, Junhao Wen, Sachio Hirokawa, “A Restaurant Recommender System Based on User Preference and Location in Mobile Environment”, 5th IIAI International Congress on Advanced Applied Informatics, IEEE, pp. 55-60, 2016

[5] Ling Li, Ya, Zhou, Han Xiong, Cailin Hu, Xiafei Wei, “Collaborative Filtering based on User Attributes and User Ratings for Restaurant Recommendation”, IEEE, pp. 2592-2596, 2017.

[6Ching, M.R.D., De Dios Bulos, R.: Improving restaurants' business performance using yelp data sets through sentiment analysis. In: ACM International Conference Proceeding Series, no. 2013, pp. 62–67 (2019
[7]Yu, B., Zhou, J., Zhang, Y., & Cao, Y. (2017). Identifying restaurant features via sentiment analysis on yelp reviews. *arXiv preprint arXiv:1709.08698*.
[8] Amel ZIANI1, Nabiha AZIZI2, Didier SCHWAB3, Monther ALDWAIRI4, Nassira CHEKKAI5, Djamel ZENAKHRA2, Soraya CHERIGUENE, "Recommender System Through Sentiment Analysis"