Restaurant Recommendation Engine-Paper

Salma Roohi Khayum  
*Computer Science Department*

University of North Carolina at Charlotte

Charlotte, USA

[skhayum@uncc.edu](mailto:skhayum@uncc.edu), 801168027

Megha Nawale  
Software and Information systemsUniversity of North Carolina at Charlotte

Charlotte, USA

mnawale@uncc.edu,5715998906

Akintonde Abbas  
*Electrical and Computer Engineering Department*

*University of North Carolina at Charlotte*

*Charlotte, USA*

[*aabbas5@uncc.edu*](mailto:aabbas5@uncc.edu)*, 801077300*

*Abstract*

Recommendation predicts the preference or rating that a user would give to an item. Knowledge discovery techniques can be applied to the problem of making personalized recommendations about items. In our case, Restaurants. The nearest neighbor-based method and the latent factor-based model are two widely used collaborative filtering methods. Collaborative Filtering algorithms give recommendations to a user based on the ratings of other users in the system. Traditional collaborative filtering algorithms face issues such as scalability, sparsity, and cold start. In the proposed framework, prediction using item and user based collaborative filtering and prediction using similarity metrics is implemented.

Keywords—Collaborative, recommendation, sparsity, styling,

# Introduction

Our project involves restaurant input dataset which uses latent vector collaborative filtering. This recommendation engine can be used to locate the restaurants if you are new to the city and also if you're not sure about what is best around. Also, this paper includes the types of recommendation engine and a brief overlook on this. Further we have tried two different methodologies in collaborative filtering and compared their outcomes.

Recommendation Systems:

Basically, there are two types of Recommendation Engine, they are:

1) Collaborative filtering

2) Content Based Filtering

Collaborative Filtering:

Collaborative filtering is a technique that can filter out items that a user might like based on reactions by similar users. This works by searching a large group of people and finding the smaller set of users with tastes like a user in consideration. Collaborative filtering is the most common technique used when it comes to building an intelligent recommendation system. Most websites like Amazon, YouTube and Netflix use collaborative filtering.

The next type is content based filtering. This system recommends the items based on a comparison between the content of the item and the user profile. This technique is used in information retrieval from a search engine, paper or book recommendation system in a library.

# Tools aand Liberies used

## Tools:

Python: We have used python 3 as programming language to implement algorithms.

Anaconda: We have used Anaconda environment as editor to write a code.

## Libraries:

Numpy for matrix operations.

Pandas for grooming and analyzing the data.

sklearn.model\_selection is used to split arrays or matrices into random train and test subsets of the data.

sklearn.feature\_extraction.text is used to convert a collection of raw documents to a matrix of TFIDF (Term Frequency Inverse Document Frequency) features. These feature vectors are used as an input to the estimator.

NLTK.corpus is used to remove stopwords from the review. Stopwords are basically the articles and pronouns which is a useless form of data in NLP as they have little lexical content.

NLTK.tokenize is used to split text into small words or tokens. Tokenization is the process of breaking up the given text into units called tokens. Tokens may be words numbers or even punctuation marks.

# Methodology

We have developed restaurant recommendation engine by using two different algorithms, one is based on latent factor and second is Neighborhood-Based method. Both methods are type of collaborative recommendation system. Collaborative recommendation is method which is based on user reviews and rating given for product. So, for restaurant recommendation system we have used Yelp dataset for customer’s reviews and built recommendation engine. In below part of paper, we have discussed working of both methods.

## Latent factor collaborative filtering

This method is a type of collaborative filtering in which simple matrix factorization is used.  In this method we have decomposed one big user rating matrix into two matrices as user-factor matrix and restaurant-factor matrix. This decomposition of one matrix into two is similar to singular value decomposition. Through this method we can find latent factors or hidden features from a rating matrix. In other words through coding we need to find hidden features of user and restaurant from the review text.

Following is matrix factorization, Matrix P rows represent users' features rating for different features and matrix Q is matrix in which columns represent restaurant’s features rating for each feature. These matrices P and Q factorized matrices of user rating matrix R. Mathematical representation of factorization of one matric in two matrices is shown in (1)

P: Rows represent users and columns represent features

Q: Rows represent features and columns represent restaurants

(1)

Matrix decomposition is helpful to find latent factors and to minimize dimensions.

For our restaurant recommendation system, we have used Yelp restaurant review dataset. Which contain review id, user id, restaurant is as business id, review text and rating as1 to 5. To find latent factor i.e to decompose rating matrix into two matrices and build recommendation system following is flow diagram.

A screenshot of a cell phone

Description automatically generated

Fig. 1. Flow diagram of Latent factor collaborative filtering

We have built our project step by step approach, Fig.2 shows pictorial representation for steps 1 to 3.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Fig. 2. Feature extraction of user and restaurant from review text

1.Removal of stop words from review text:

By using a natural language processing toolkit we have removed stop words from review text. And now our data frame contains only featured words in review text. Example: As shown in diagram, in this step review text of user 1 : “I like this burger “converted to “Like Pizza” after removing stop words.

2.Aggregate review of each user and each restaurant:

After removing stop words from review, we have separated our data frame into two data frames, one has user id and their reviews and second has restaurant id and review got for that restaurant. After separation we have aggregated review in one based on user id and restaurant id. For example, for user 1 we have collected reviews given by user1 in one cell. And for restaurant 1, we have collected all reviews for restaurant 1 in one cell.

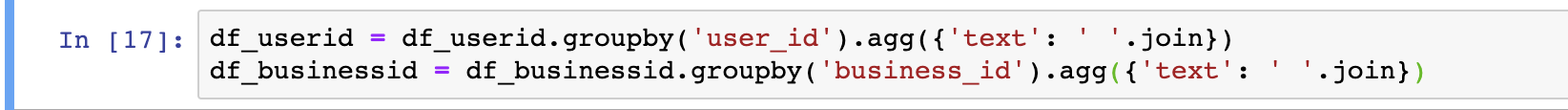


Fig. 3. Python code for aggregating review for user and restaurant based on user id and business id respectively.

After aggregating reviews for each user and for each restaurant, result is as follows for one user.

A screenshot of a cell phone

Description automatically generated

Fig. 4. Aggregated string for one user.

3. Feature extraction from aggregate review for user and restaurant and convert into matrices.

Now we have an aggregate review for each user and each restaurant based on used id and restaurant id. It is time to find latent factors and their weightage and convert into two matrices (Matrix 1: User-feature matrix, Matrix2: Restaurant-feature matrix).

We have achieved it through TFIDFvectorizer. In this step based on tokens (words) in review it assigns some numerical value into user-feature matrix and restaurant feature matrix.

A screenshot of a cell phone

Description automatically generated

Fig. 5. Feature extraction from aggregated review for user.

In this we have limited feature count to 3000. So now we have 10937 different users and 3000 features.

A screenshot of a cell phone

Description automatically generated

Fig. 6. Feature extraction from aggregated review for restaurant.

In this we have limited feature count to 3000. So now we have 1411 different restaurants and 3000 features.

4.Latent factor filtering Optimization:

One problem in the recommendation system is that, matrix is sparse as each user has not visited each restaurant so the number of cells are empty in the rating matrix. So using optimization we solved this issue. To fill empty cells we have used gradient descent formula as shown in (2)

A close up of a logo

Description automatically generated

(2)

κ is the set of the (u,i) pairs for which rui is known.

Here we have initialized λ as 0.02 after some try. With 20 steps we have implemented a regression module to optimize ratings. This optimization is to fit actual ratings to user and restaurant matrices.

A screenshot of a social media post

Description automatically generated

Fig. 7. User and restaurant latent factor’s rating optimization

5.Resturant Recommendation for new user

After optimization of latent factor metrices, we got 2 matrices and rating is related to latent factors of user and restaurant. When new user search for restaurant with some keywords, system will provide possible best list of restaurants from business dataset with basic information of restaurant. Fig.8 shows python code for illustration recommendation engine work.

A screenshot of a social media post

Description automatically generated

Fig. 8. Recommendation of restaurant based on search criteria

## Neighborhood-Based Method

Asides latent factor models, neighborhood-based methods are also another commonly used type of collaborative filtering. As such, these methods could also be applied to building a recommendation system. According to [1], neighborhood-based, also referred to as memory-based methods, are hinged on the premise that similar users display similar rating behavior and similar items get similar ratings. This premise results in the broad classification of neighborhood-based methods into item-based and user-based approaches.

For this project, an item-based approach was applied. Specifically, the restaurants available are considered as items. This selection was based on the premise that the number of restaurants (i.e. number of items) is still quite tractable and as such, it would be computationally efficient to base our recommendations on item-item similarities. The user interacting with the recommendation engine is expected to provide a text-based description of the kind of restaurant he/she is looking for. This text-based description can then be compared with existing reviews for the available restaurants. Fig. 9 provides an overall flow of the approach.

A screenshot of a cell phone

Description automatically generated

Fig. 9. Process flow of neighborhood-based approach for recommendation engine

As shown in Fig. 1, the first step involves the aggregation and preprocessing of text-based reviews for each restaurant. The aggregation is particularly important because each restaurant has several text-based reviews and as such, it becomes important to put all reviews for each restaurant into a single body of text. The processing part of the initial step involves the removal of stop words from the aggregated text and the tokenization of the aggregated text. Afterward, the aggregated and processed texts are converted into vectors, with each restaurant having its vector. This is to put prepare the text-based reviews for further mathematical computations that will be executed own the line. The TF-IDF vectorizer, which has been highlighted in previous portions of this report, was used for this text vectorization.

In addition to the text-based reviews, numeric ratings for each restaurant, given by different previous users, are also available. As such, it is necessary to obtain a single representative rating for each restaurant. For this representative rating, the average of the available ratings was each restaurant was then calculated. While it is well known that the mean metric is highly sensitive to outliers, such outlier effects are almost non-existent in this application. This is because each rating is restricted to integers between 0 and 5. As such, the mean metric is deemed sufficient for this application.

After the previously described preliminary processes have been executed, the next set of steps deal with the specific text inputs provided by the users of the recommendation engine at each point in time. The first of these steps is the conversion of the user’s text input into vectors. This step is similar to the conversion of the text-based reviews for each restaurant into vectors. After this, similarities between the user’s input vector and review vectors for each restaurant are calculated. Typical metrics used to measure these similarities include the Jaccard’s similarity index, Cosine similarity index,  Euclidean distance etc.

1.Jaccard similarity index: this metric defines the similarity between two vectors (or sets) by dividing the total number of common elements between the two vectors (or sets) by the total number of observations in both vectors (or sets). Mathematically, the Jaccard similarity score is expressed as shown in (3).

(3)

A and B represent the two vectors or sets being considered. Also, the Jaccard similarity index ranges between 0 and 1. However, the Jaccard similarity score is highly sensitive to small sample sizes.

2.Euclidean distance: The Euclidean distance is also referred to as the L2-norm. It computes the similarity between two text vectors based on the magnitude of the Euclidean distance between those two vectors. However, the Euclidean distance metric may give erroneous results when the texts being considered have very different sizes i.e. magnitude of on text vector is far greater than the magnitude of the other text vector.

3. Cosine similarity index: this expresses the similarity between two vectors as the angle between the vectors. Mathematically, this is expressed as shown in (4).

(4)

The cosine similarity score ranges between -1 and 1 with 1 representing perfectly similar items and -1 representing perfectly dissimilar items. A major strength of the cosine similarity index is that it is robust to the size of each text body (i.e. text vector magnitude) as it only focuses on the angle between the vectors representing the text. As such, the cosine similarity index was used as the similarity metric in this project.

After calculating the similarities between the user input text vector and the text vectors for the restaurant reviews, the similarities are sorted using the similarity score and the average rating for each restaurant. The actual implementations of the described process and recommendations for an input text “awesome breakfast and calm atmosphere” in a Jupyter notebook environment are shown in Fig. 10 and Fig. 11 respectively.

A screenshot of a social media post

Description automatically generated

Fig. 10. Implementation of neighborhood-based method

A screenshot of a social media post

Description automatically generated

Fig. 11. Generated Recommendations

# Comparison between two methods

As following the discussions on both the models (i.e.) latent based and the neighborhood based we have observed that the latent based approach works more efficiently why? for this use case than the other approach of the neighborhood model as that would add the salt of expansion.

Latent factor model is more scalable in terms of memory as in this method rating matrix is decomposed into two matrices with less numbers of columns and rows. In other words when we decompose one matrix into two matrices . two matrices represent features only and features are limited compared to ratings, so this method require small space of memory. In Rating matrix multiple rows for single user but in decomposed matrix has single row for one user. This is how this latent factor model is scalable.

Latent based model comprises non-linear optimization so the initial start of the process may be time consuming and tedious but once the right data set is given as the input and is run through some algorithms the linearity could be increased. The scalability aspect was a careful watch during the experiment and comparison of both the approaches finally landed up on determining that latent based approach is suitable due to the above mentioned reasons.

# Future scope

The scope is very vast for this business as restaurants range as the second largest private sectors in this boomed economy. We have used just one data set from yelp for this specific comparison but having a deep dive we see that there would be a zillion number of combinations and the data set also would increase at exponential pace depending on the factors considered and assumptions made. One future scope of that would be adding pricing related data with the choices of restaurants visited or pricing depending on the time of the year or frequency of people visiting and etc for which we can scale this latent based approach or use another approach of K-Nearest neighbor method.

# Conclusion

Depending on the user reviews the recommended systems can be deployed based on both the approaches but in this paper we conclude by saying which approach would better serve our test case. Natural language processing is processed for translating the reviews into machine understood languages. There would be matrices formed based on the user and restaurant features and depending on the combination of their usage. Thus we conclude by saying that the latent based approach would be a better go for this use case rather than the neighborhood based approach.

##### References

1. C.Aggarwal, Neighborhood-Based Collaborative Filtering. In: Recommender Systems. Springer, Cham, 2016
2. Y. Koren, R. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems. Computer, 42(8):30–37, 2009.