Hybrid Travel Recommender System

Nilufa Yeasmin

1. Introduction and Motivation

One of the first things to do while planning a trip is to book a good place to stay. Booking a hotel online can be an overwhelming task with thousands of hotels to choose from, for every destination. Motivated by the importance of these situations, we decided to work on the task of recommending hotels to users. We used Expedia's hotel recommendation dataset, which has a variety of features that helped us achieve a deep understanding of the process that makes a user choose certain hotels over others. The aim of this Hybrid Travel Recommender System project is to predict and recommend five hotel clusters to a user that he/she is more likely to book given hundred distinct clusters.

2. Problem Definition

Nowadays, online reservations have become very popular and travellers find it much easier to book hotels of their choice online. However, people have a difficult time choosing the hotel that they want to book. Therefore, the recommender system comes into play. For the goals of the project, we have used 4 different recommendation models – 2 Collaborative filtering techniques and 2 Hybrid technique. The motivation behind this was to explore different models in the field of recommender systems and come up with an efficient solution. ***Memory-Based CF by computing cosine similarity*** and ***Model-Based CF by using singular value decomposition (SVD)*** technique were the Collaborative Filtering techniques used and Hybrid technique uses a combination of ***Wide and Deep*** technique and ***DeepFM*** technique.

3. Dataset

We have used the Expedia Hotel Recommendation dataset from Kaggle. The dataset, which had been collected in the 2013-2014 time-frame, consists of a variety of features that could provide us great insights into the process user go through for choosing hotels. The training set consists of 37,670,293 entries and the test set contains 2,528,243 entries. Apart from this, the dataset also provide some latent features for each of the destinations recorded in the train and test sets. The data is anonymized and almost all the fields are in numeric format. The goal is to predict 5 hotel clusters where a user is more likely to stay, out of a total of 100 hotel clusters. The problem has been modeled as a ranked multi-class classification task. Missing data, ranking requirement, and the curse of dimensionality are the main challenges posed by this dataset.

3.1. Data Preprocessing and Exploratory Analysis

Our first step was to clean and pre-process the data and perform exploratory analysis to get some interesting insights into the process of choosing a hotel. The first thing we observed was that there were many users who have only searched for hotels and did not make any reservation. Moreover, the test data had only the users who made a reservation. Thus, we pruned the dataset by removing all the users who did not make any booking as these entries do not provide any indication of which hotel clusters those users prefer and this could possibly have interfered with making predictions.

	Feature	Description
1	date time	Timestamp
2	site name	ID of Expedia point of Sale
3	posa continent	ID of site's continent
4	user location country	ID of customer's Country
5	user location region	ID of customer's region
6	user location city	ID of customer's city
7	orig destination distance	Physical distance between a hotel and a customer
8	user id	ID of user
9	is mobile	1 for mobile device, 0 otherwise
10	is package	1 if booking/click was part of package, 0 otherwise
11	channel	ID of a marketing channel
12	srch ci	Check-in date
13	srch co	Check-out date
14	srch adults cnt	Number of adults
15	srch children cnt	Number of children
16	srch rm cnt	Number of rooms
17	srch destination id	ID of the destination
18	srch destination type id	Type of destination
19	hotel continent	Hotel continent
20	hotel country	Hotel country
21	hotel market	Hotel market
22	is booking	1 if a booking, 0 if a click
23	cnt	Number of similar events in the context of the same user sessiont

24	hotel cluster	ID of hotel cluster

Table 1: Features used for Training

From the remaining entries, we identified the searches by each user belonging to a specific type of destination. This gave us some useful information about which hotel cluster was finally chosen over other hotel clusters explored by the user. One important observation to note is that few users might be travel agents and could explore multiple type of destinations at the same time. This could also be true for few users who are planning multiple vacations at the same time. That is why we considered the preferences of the users separately for each destination type he/she explored. Also, after a booking was made, subsequent searches by the user were treated separately. We describe aforementioned approach with an example below.

In table 2, the searches made by user 1 are shown. There are 2 types of destinations: 11938 and 8821. Based on the type of destination, we identify the hotel clusters that were rejected/selected. For destination type 11938, cluster 52 was selected, and clusters 87 and 65 were rejected. Similarly, for destination type 8821, cluster 20 and cluster 35 were rejected. Now, we keep a track of the rejected clusters.

User ID	**Hotel Cluster**	**Destination Type**	**Booking**
54261	65	11938	0
54261	87	11938	0
54261	52	11938	1
54261	20	8821	0
54261	30	8821	0

Table 2: Subset of training set

Also, we use the check-in and check-out dates to find the duration of the stay for each of the entries in the training set. From the Figure 1, we see that all the clusters seem equally likely when the duration is short, but as the duration increases, certain hotel clusters are preferred over the others. This seems to be a good feature to identify the hotel cluster user would choose.

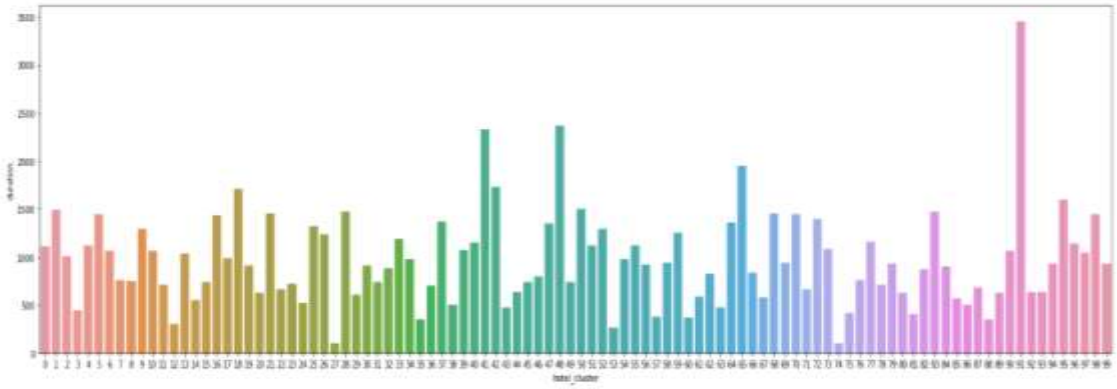


Figure 1: Duration of stay vs hotel cluster

Furthermore, as per the details provided by Expedia on the competition page, hotels tend to change their cluster seasonally. To capture this variation, we also included one-hot representation of the month from which user is seeking to start his/her stay. Finally, we make use of the latent features of the destinations provided in the dataset.

Finally, we make use of the latent features of the destinations provided in the dataset. However, since we have 149 latent features for each destination, we decided to apply PCA to extract the most relevant dimensions.

Next, we visualize the correlation matrix between the features of the training set in Figure 2 and observe and observe following things:

- ***hotel_cluster*** does not seem to have a strong (positive or negative) correlation with any other feature. Thus, methods which model linear relationship between features might not be very successful.

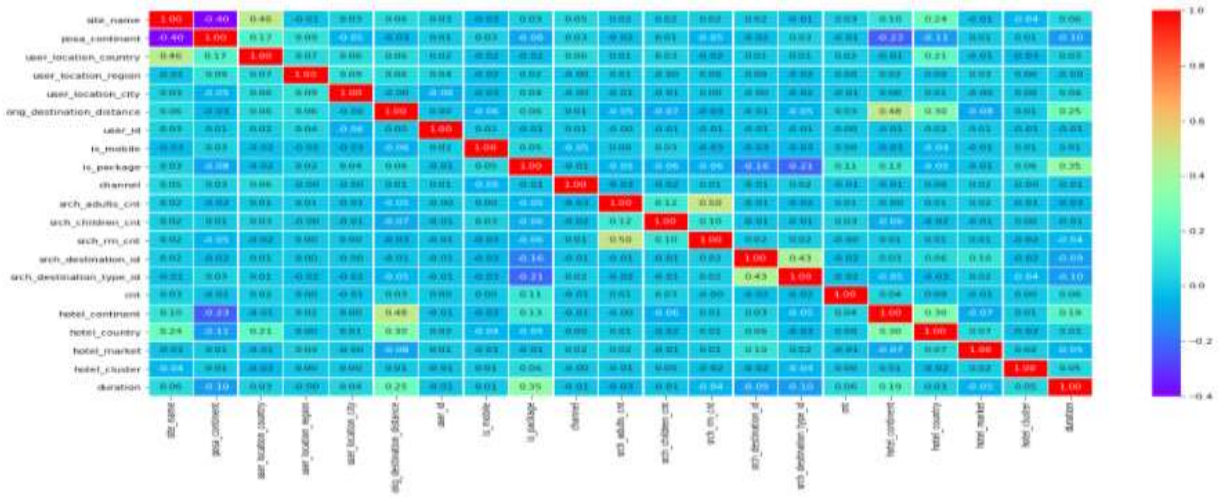


Figure 2: Correlation matrix of features

- ***orig_destination_distance*** has a positive correlation with duration (constructed using srch_ci and srch _co), which means people who are planning for a long trip tend to go far away from the place of origin.

- ***hotel_continent*** and ***posa_continent*** (which is from where the booking is done) are negatively correlated. This means that people tend to go to continents different from theirs for vacations.

- ***duration*** seems to have a strong positive correlation with ***is_package***. This means that people who tend to book hotel for longer duration usually choose hotels as a part of a package.

- ***srch_destination_id*** has a strong correlation with ***srch_destination_type_id***. This is expected as each destination would have an associated type; for example, vacation spot, city, etc.

- ***duration*** is also positively correlated with ***hotel_continent*** which means certain continents are preferred for longer duration of stay.

- ***srch_rm_cnt*** has a very strong correlation with ***srch_adults_cnt***, and to an extent, with ***srch_children_cnt*** also. This is expected as people tend to take rooms based on how many families/ couples are there.

Feature Engineering

- We applied some "Feature Engineering Techniques" to prepare input data to produce results. The date time, check-in date and check-out date columns can't be used directly. Therefore, some features such as month, year, number of searches, trip duration and number of bookings, solo trip or family trip, were extracted from data.

- We have 149 latent features for each destination; we decided to apply PCA to extract the most relevant dimensions.

- Expedia on the competition page, hotels tend to change their cluster seasonally. To capture this variation, we also included one-hot representation of the month from which user is seeking to start his/her stay.

4. Model Training Methodology and Evaluation:

As mentioned earlier, 4 different models were used to obtain the recommendations. In this section, we will discuss about the models used in detail, the input for the model, training and evaluation of the model.

i. Collaborative-Filtering (Baseline):

We have used Collaborative-Filtering model as the baseline model. Collaborative Filtering can be divided into Memory-Based Collaborative Filtering and Model-Based Collaborative filtering. First of all, we have performed some data analysis for removing duplicates data and understand better user item rating distribution. We have created user-item matrices for train and test set. Then performed some data analysis on the rating matrix and transformed rating matrix into average rating matrix (per user/row). Then, two collaborative filtering models were implemented from scratch.

- **Memory-Based CF by computing cosine similarity:** Memory-Based Collaborative Filtering approaches can be divided into two main sections: user-item filtering and item-item filtering. A user-item filtering will take a particular user; find users that are similar to that user based on similarity of ratings, and recommend items that those similar users liked. In contrast, item-item filtering will take an item, find users who liked that item, and find other items that those users or similar users also liked. It takes items and outputs other items as recommendations.

  In both cases, we created a user-item matrix that builds from the entire dataset in order to make recommendations. Since we have splited the data into testing and training, we have created two matrices. The training matrix contains 75% of the ratings and the testing matrix contains 25% of the ratings. After, we have built the user-item matrix; we calculated the similarity and created a similarity matrix. The similarity values between items in Item-Item Collaborative Filtering are measured by observing all the users who have rated both items. For User-Item Collaborative Filtering the similarity values between users are measured by observing all the items that are rated by both users.

  A distance metric commonly used in recommender systems is cosine similarity, where the ratings are seen as vectors in n-dimensional space and the similarity is calculated

based on the angle between these vectors. I've used sklearn (pairwise_distances function from) to calculate the cosine similarity. The output ranges from 0 to 1 since the ratings are all positive. Next step was to make predictions from similarity matrices(user_similarity and item_similarity).

Evaluation: There are many evaluation metrics but for this project we used, most popular metric to evaluate accuracy of predicted ratings are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Model with lowest RMSE and MAE were used for recommendations. RMSE is calculated by finding the difference between predicted and existing rating, squaring them and then finding the square root of that value. The MAE mean of the absolute value of the errors is instead of sum square error, it takes sum of absolute value of error. The model with least RMSE, MAE denotes that the user-item interaction embedding has been learnt better and the model is able to predict user ratings to unseen items more accurately.

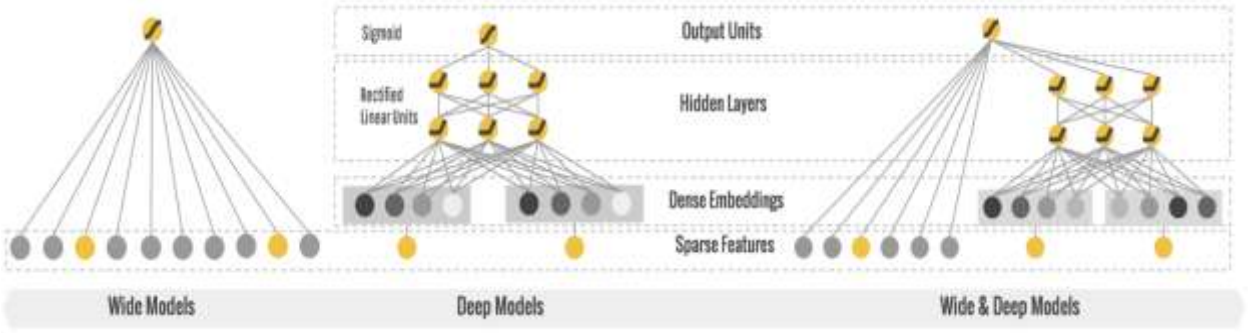
- **Model-Based CF by using singular value decomposition (SVD):** Based CF models are developed using machine learning algorithms to predict a user's rating of unrated items. As per my understanding, the algorithms in this approach can further be broken down into 3 sub-types such as Clustering based algorithm, Matrix Factorization, Deep Learning. For example, we can mention few of these algorithms SVD, NMF, KNN etc.

  We have used matrix factorization method with Singular value decomposition (SVD). Collaborative Filtering can be formulated by approximating a matrix X by using singular value decomposition.

ii. Wide & Deep:

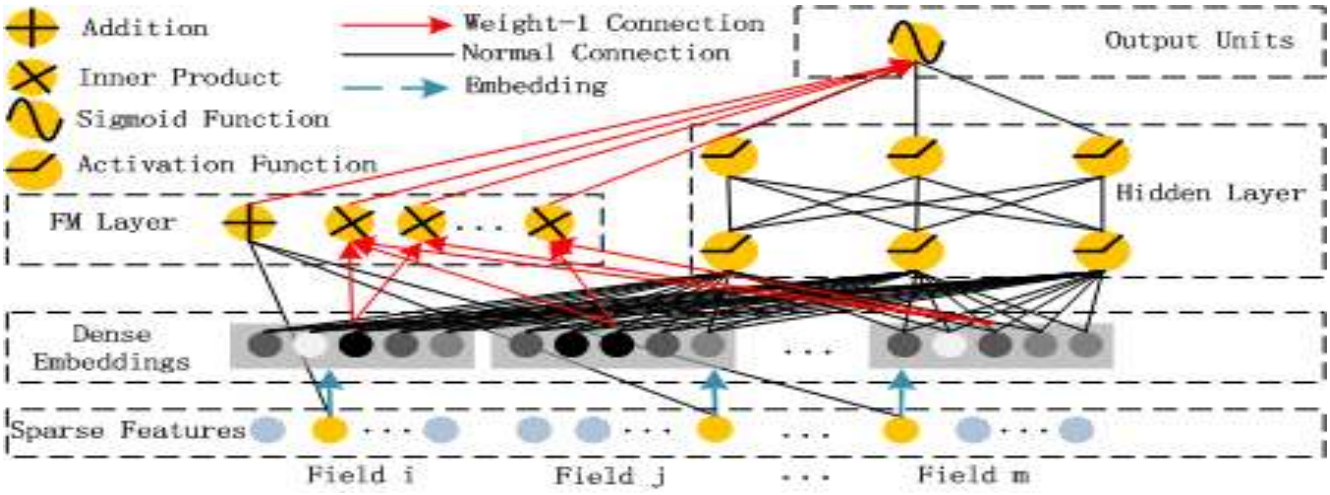
The wide and deep learning has two individual components. The wide network is a linear estimator or a single layer feed-forward network which assigns weights to each features and adds bias to them to model the matrix factorization method, as illustrated in above figure (left). These feature set includes raw input features and transformed. The deep component is a feed-forward neural network, as shown in above figure (right). The high dimensional categorical features are

first convert into a low dimensional and dense real-valued vector, often referred as embeddings. The embedding vectors are initialized randomly and then the values are trained to minimize the final loss function. Then they are fed into the hidden layers with feed forward step. By jointly training the wide and deep network, it takes the weighted sum of the outputs from both wide model and deep model as the prediction value.



Evaluation: For evaluating wide and deep model, we have calculated the value of Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Squared Error (MSE) and Area under the ROC Curve. We have done hyper-parameter tuning to get best model.

iii. **DeepFM:** DeepFM is a mixed approach between FM and a deep neural network, that both share the same input embedding layer.



The Addition part of the FM Layer gets the raw input vector x directly (Sparse Features Layer) and multiplies each element with its weight ("Normal Connection") before summing them up. The Inner Product part of the FM Layer also gets the raw inputs x, but only after they have been passed through the embedding layer and simply takes the dot product without any weight ("Weight-1 Connection") between the embedding vectors.

The Deep Component is deep neural net architecture (regular feed-forward MLP neural net). The MLP network with embedding layer between the raw data (highly sparse due to one-hot-encoded categorical input) with σ the activation function, W the weight matrix, a the activation from the previous layer, and b the bias.

Evaluation: For evaluating wide and deep model, we have calculated the value of Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Squared Error (MSE) and Area under the ROC Curve. We have done hyper-parameter tuning to get best model. . The model with least RMSE, MAE, and MSE considered as model prediction.

5. Results and Discussion:

Table 3 summarizes the results that we have achieved for each of the method. We can see from the Table 3, that collaborative filtering model (baseline) provides vary high value of RMSE and MAE. We tried to find out reason behind this. We have done data analysis on the rating matrix, found average rating per user, and average rating per movie. The problem with our baseline

model (collaborative-filtering recommender system) was that we couldn't use more features of our large dataset. Therefore, we decided to build hybrid recommender systems to be able to use not only user_id, item_id, and rating features, but also 20 more features that would give us more insights and better recommendations. However, DeepFM provides least RMSE and MAE value compare to Wide and Deep model.

Model		**RMSE**	**MAE**
Memory-Based Collaborative Filtering	**User-based CF RMSE**	0.959	
	Item-based CF RMSE	0.964	
	User-based CF MAE		0.978
	Item-based CF MAE		0.981
Model-Based Collaborative Filtering	**SVD**	0.957	0.916
Wide and Deep		0.260	0.140
DeepFM		0.241	0.131

Table 3: Evaluation Metrics of each Model

6. Conclusion and Future Work:

We have successfully implemented a hotel recommendation system using Expedia's dataset even though most of the data was anonymized which restricted the amount of feature engineering we could do. The most important and challenging part of implementing the solutions was to create and extract meaningful features out of the 38 million data points provided to us. After applying multiple models and techniques, we arrived at the conclusion that Hybrid Model performs best. The scope of the project is really wide and hence there is a lot of room for improvement and future work. Few suggestions are as follows:

i. The dataset from the Kaggle Expedia Hotel recommendations competition. We can scrape data from other website such as tripadvisor. There are several other websites that can contain more relevant information on the items and hence can be used.

ii. The project can be extended by creating an interactive web application to provide recommendations to get information from user. Using the web application, interactions of users can be collected to improve the user experience and provide implicit recommendations.

iii. In our project, three recommendation algorithms were used. Various other recommendation algorithms can be used and fine-tuned to improve the accuracy of the recommendations provided. For instance, xDeepFM or Deep & Cross Network can be used to improve the recommendations.