Yelp Restaurant Recommendation System using Ensemble Learning

Amit Garg – 014541072 – amit.garg@sjsu.edu
Maaz Sirkhot – 014279616 – maaz.sirkhot@sjsu.edu
Shriya Vanvari - 014561417 - shriya.vanvari@sjsu.edu

Repository Link: https://github.com/shriyavanvari/Yelp-Recommendation-System

Introduction

- **Motivation**

  Watching Netflix always has been our amazing experience and we realized this was because how personalized experience it provides in terms of suggesting movies and TV Shows. Similarly, we wonder if similar strategy could be applied to restaurant search as when we especially travel to a different city or even in the same city, we are looking for eating joints, it becomes little difficult to find the most suitable restaurant. Therefore, it is important that instead of simply listing all the restaurants around the area, we get listings of restaurants we are really interested in. More personalized the suggestions, better would be the experience as the hassle of not making up mind to find a restaurant is avoided.

- **Objective**

  We propose to build a model that would suggest restaurants to the user based on there previous experiences. The idea is to leverage the past activities and preferences to provide a personalized experience such that hunting for restaurants become a productive task. We plan to consider the ratings given by all the users on the system to each restaurant to train a model which when simulated with the preferences mentioned by the user, would fetch top recommendations for the user. While the data collected by restaurants is varied and ambiguous, our goal is to standardize the data and process the same to a standard structure using multiple Data Preprocessing methods. Further, we will train multiple models on this standardized data and collect independent results from all models. Lastly, these results would be combined and processed to find top 10 results which will be shown to the users.

- **Literature/Market Review**

  Restaurant Recommendation using Latent Factor Collaborative Filtering:
  This recommendation system was built using Latent Factor Collaborative filtering algorithm by considering stars given, reviews text and other business data. In this system, data was cleaned using NLTQ libraries to remove stop words and punctuations. The system extracts feature from the review text and combines all the reviews to form a single paragraph. This text data undergoes TFIDF vectorization. Similar approach is followed for restaurant data as well. Additionally, the matrix is created with users and business ratings. With the two matrices for user features and business features, the system predicts the ratings for all the user-business pairs according to the error. Finally, the latent factor collaborative filtering algorithm is applied on the data to suggest restaurants with a test data that provides a review text.

  Restaurant Recommendation based on location using Cosine Similarity:
  This system is very similar to the previous system discussed in terms of data preprocessing and applying TF-IDF. However, this system considers location of the restaurants to provide suggestions using cosine similarity after features are extracted from the data. This model focuses heavily on the statistical analysis of the data for recognizing patterns.

System Design and Implementation Details

- **Algorithms:**
  We implemented three algorithms to suggest restaurants. The results of these three algorithms are combined to provide top 10 restaurants to the user. The three algorithms are,

  Collaborative filtering using neural networks: We used FastAI library's pretrained models to which data was fed. A dense neural network predicts the ratings for the missing restaurants and collectively analyses these ratings to provide recommendations.

  Collaborative filtering using SVD:
  Since dataset was huge, we performed down sampling of data by 20% for active users. We used out of time approach for training and validating the model. The key step included matrix factorization for collaborative filtering which would give us similar users. Using the similar users, we compiled list of restaurants for the given test data.

  Content based filtering: This approach implements a popular strategy of standardizing the data and finding similarity between the features. In this method, sparse csr matrix undergoes cosine similarity to find correlation between the data. Obtained correlation matrix provides the standard which is used to find similar scores of other restaurants and in turn achieve closest possible contents.

- **Technologies and Tools:**
  We used Python libraries for implementing all the algorithms. Various libraries such as NumPy, Pandas, Matplotlib, Sklearn, Scipy, Sparse, NLTQ etc. are utilized. All the computations were completed in Jupyter Notebook.
  To implement the neural network architecture, we used FastAI library which is a high-level-library based on PyTorch framework. It provides pre-trained models and achieves state-of-the-art results.
  Surprise package allowed us to use various combinations of hyperparameters which as a result helped us to avoid overfitting on modeling sparse rating matrix.

- **Architecture Related Decisions:**
  The beginning phase of analysis included retrieving the data and preprocessing it. As part of preprocessing activity, we extracted only required features from the dataset instead of all features. Therefore, we reduced the dimensions in our data as well as got rid of noise. This was key decision as every model of ours depended upon the data and we really wanted to have single dataset that could be used everywhere instead of customizing the data every time leading to inaccurate results.
  We had planned to use clustering techniques to find the similar restaurants and predict ratings. However, we soon realized that the type of data is not best suitable for clustering, instead we need to do filtering as there were large number of attributes to account for.
  For each of our models, we had to decide the factor of down sampling the data as matrix columns could not be in millions. For SVD model, we down sampled the data on the user ids such that only the active users were considered even after down sampling. Moreover, we implemented matrix

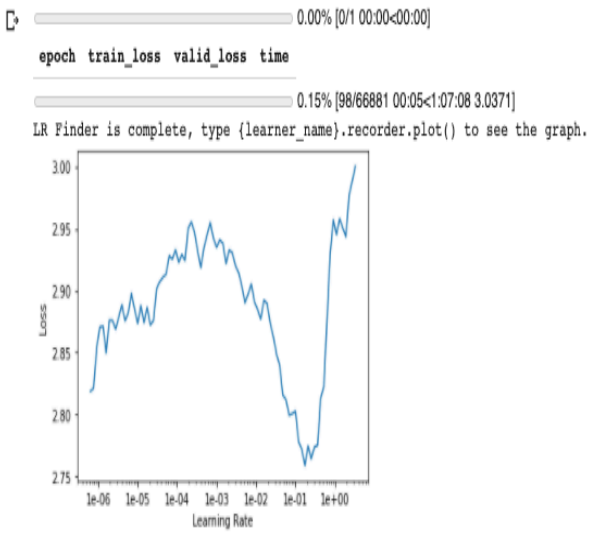
factorization over K-nearest neighbors since our matrix was sparse and it would be difficult to find similar users accurately.
The goal of all our experiments was to find the best hyperparameters for our models. Hence, we used RMSE (Root Mean Squared Error) to determine the best hyperparameters. Parameters with lowers RMSE were picked.

- **Component Details**
  Data preprocessing component takes the JSON data, creates dataframes with required columns and prepares the data for feeding to the model.
  The restaurant recommendation system consists of three major components where each component runs on its respective algorithm. All the three components take user, restaurant and ratings data to train the models and provide ten recommendations each.
  Apart from these, we have another component that merges the results from each algorithms and finds best restaurant recommendations.

- **Use cases/screenshots**
  FastAI model snippet:

To find the appropriate learning rate for our architecture, we used lr_find function provided by FastAI.

```
[ ] learn.lr_find() # find learning rate
    learn.recorder.plot() # plot learning rate graph
```




```
[ ] from fastai.callbacks.tracker import *
    learn.fit_one_cycle(1, 1e-2, callbacks=[SaveModelCallback(learn, every='epoch', monitor='valid_loss', name='collab_model'), EarlyStoppingCallback(learn, monitor='v
```

User	Recommended Restaurants
4401	16971,33485
28355	33485,3807
10212	49702
8994	48809
9519	43076,49702
8583	43076,54783,3807
1179	43706
4711	43076,54783,32418,3807
4544	49702,43076
8057	49702,50791,43076
8624	43076
13169	43076
5942	48809
219	33485,49702,43076
20734	43076,48809,54783
37969	43076,48809,49702
7258	43076,54783,49702,32418
55272	43076,33485,48809
192533	43076,49702,33485

```
28355 : [3807, 912, 4473, 6251, 620, 1464, 2137, 4684, 3012, 1274]
8994 : [1274, 2137, 4684, 4641, 5668, 4465, 4236, 2650, 2459, 5050]
4711 : [3985, 2289, 4540, 3136, 4522, 3244, 2516, 6209, 1866, 1339]
2120 : [2256, 5352, 1557, 1328, 5994, 4323, 4585, 4866, 2705, 5279]
5597 : [5681, 2846, 2939, 6056, 2178, 5683, 2532, 3924, 5094, 471]
20734 : [4982, 1826, 2375, 529, 6030, 3288, 2046, 4543, 1225, 4973]
62535 : [2870, 370, 1724, 43, 5167, 5059, 4090, 4374, 653, 477]
4416 : [922, 2774, 3965, 2955, 4529, 4714, 5783, 1596, 3929, 2749]
7258 : [3965, 2790, 4680, 3621, 1321, 4106, 5228, 4170, 3594, 5289]
55107 : [3661, 297, 323, 464, 4086, 4468, 4314, 5079, 6086, 5273]
9100 : [5945, 2368, 1300, 1754, 3715, 2269, 5096, 2465, 254, 2843]
9124 : [2323, 5351, 4465, 1303, 2137, 3382, 6212, 1887, 1114, 1051]
8583 : [1274, 3575, 5065, 5686, 3865, 1269, 1835, 4641, 4845, 1127]
2934 : [2658, 3696, 2716, 3389, 2303, 2211, 5211, 223, 4012, 2293]
10212 : [1853, 4364, 4798, 4989, 2530, 2982, 1158, 2860, 1313, 1956]
9519 : [2843, 5261, 3029, 5826, 5094, 5945, 1685, 2487, 2368, 648]
5495 : [100, 3235, 3103, 6172, 3635, 1426, 3887, 4624, 1802, 4147]
33 : [5859, 5653, 4878, 5359, 2542, 1278, 938, 597, 3899, 4973]
725 : [189, 6026, 3596, 2269, 5683, 537, 6234, 2514, 1826, 1318]
4401 : [2295, 3998, 679, 1570, 826, 5279, 4799, 3913, 891, 1228]
2220 : [1031, 1549, 4131, 650, 4691, 5493, 4777, 5414, 426, 5606]
57377 : [6076, 2268, 5339, 3683, 2335, 2658, 6119, 6092, 2310, 6251]
609 : [3828, 2945, 5265, 415, 3126, 5933, 2787, 4259, 653, 6006]
219 : [531, 3684, 5672, 5706, 614, 6026, 419, 324, 597, 33]
89784 : [463, 2799, 1831, 1298, 2918, 3913, 2295, 5837, 1570, 679]
59544 : [3558, 864, 5453, 3103, 3574, 2577, 1257, 4246, 1376, 5367]
1522 : [2287, 2315, 3288, 4076, 2764, 4767, 1382, 101, 1123, 1791]
37993 : [2071, 3574, 172, 5527, 5480, 1888, 5385, 676, 2413, 3399]
2284 : [1127, 3652, 1844, 2508, 3575, 4358, 5065, 1449, 1490, 3630]
31039 : [2203, 2547, 4832, 2035, 2737, 4992, 1367, 3032, 1874, 1645]
5942 : [6212, 3871, 5688, 3592, 1091, 6197, 4304, 4192, 5025, 1486]
6075 : [6152, 6030, 5804, 4938, 5017, 2441, 2874, 2941, 504, 2431]
80035 : [47, 4218, 6165, 178, 2816, 4624, 2369, 5364, 382, 1503]
4948 : [2130, 578, 3376, 4965, 3419, 6131, 1344, 657, 5911, 5686]
37969 : [542, 1289, 3661, 5132, 6178, 5691, 3081, 154, 2921, 2097]
8624 : [6016, 2089, 3412, 369, 2934, 585, 1894, 3432, 4380, 2597]
4647 : [3704, 2427, 2830, 2009, 3235, 5364, 2844, 2763, 2721, 4862]
7953 : [522, 5224, 1400, 3701, 2669, 4521, 5020, 3600, 6238, 5405]
8057 : [4684, 2712, 2137, 5241, 2007, 6119, 4845, 3218, 4930, 4025]
55272 : [3091, 1635, 1931, 650, 3934, 6100, 4976, 57, 3784, 4216]
13169 : [2920, 3895, 2403, 4580, 3658, 547, 992, 3887, 4932, 3499]
44488 : [4297, 5228, 2447, 3818, 2075, 1589, 2314, 1749, 1596, 6242]
670 : [4173, 1332, 4290, 4824, 6233, 1566, 2257, 43, 6167, 3820]
11442 : [5003, 2712, 3807, 3596, 2335, 2883, 3678, 1091, 1464, 4109]
90825 : [2378, 990, 2118, 2695, 6192, 1515, 5643, 1641, 6003, 3624]
4544 : [4073, 4176, 893, 4348, 2275, 3383, 4051, 1862, 5702, 2265]
29426 : [4951, 3622, 5818, 670, 263, 419, 5641, 2431, 1880, 1949]
1179 : [220, 6149, 1838, 2431, 1612, 4600, 1254, 5571, 2375, 5096]
1637 : [5335, 927, 1274, 358, 3807, 4473, 1138, 5148, 4477, 1049]
192533 : [4649, 5825, 4862, 2840, 4550, 333, 4562, 2014, 5087, 1739]
```

Experiments/Proof of Concept Evaluation

- **Datasets used**

  We used Yelp restaurant data set provided publicly on Kaggle. The dataset contains JSON files which has total 5,200,000 user reviews, 174,000 restaurants spanning across 11 metropolitan areas.

  The files are listed as follows:

  1. Yelp_academic_dataset_business.json
  2. Yelp_academic_dataset_checkin.json
  3. Yelp_academic_dataset_review.json
  4. Yelp_academic_dataset_tip.json
  5. Yelp_academic_dataset_user.json

  Each of the above listed file holds respective data which is of over 10 GB in total size.

  Dataset Link: https://www.kaggle.com/yelp-dataset/yelp-dataset

- **Methodology followed**

  Collaborative filtering using Neural Networks:

  Our model architecture consists of 2 fully connected layers with 256, 128 neurons resp. Data was fed into a network in the (“User”, “Restaurant” and “Rating”) format. We used embedding layers of 100 neurons for the user as well as the restaurant. Dropout layers were also added to avoid overfitting. We used the ADAM optimizer for smooth convergence of loss. During the training phase, various callbacks were added for early stopping and saving the best model on each epoch so that it can be used later to predict the ratings.

Collaborative filtering using SVD:
We split the data such that our 50 common users were used in test, remaining 90% training-10% validation for evaluating the proper hyper parameters for the SVD model. The matrix factorization is done on the user-item ratings matrix. From a high level, matrix factorization can be thought of as finding 2 matrices whose product is the original matrix. SVD decomposes a matrix into constituent arrays of feature vectors corresponding to each row and each column.

Content Based filtering:
This method relies on the user ratings and past reviews for training the model and find similar restaurants. In this approach, we tried to find similarity between the users in terms of their reviews to respective restaurants. These similarities are extrapolated to find whether a restaurant will be preferred by the user or not. Finally, we compile the list of restaurants in a descending order of the similarity score thereby the restaurant with highest similarity score would be most recommended.

- **Implementation Details:**

Neural network model:

Index	Architecture and Parameters	Validation data	Validation loss	RMSE
1	• User embedding size: 100 • Restaurant embedding size: 100 • Embedding Dropout: 0.25 • 2 Hidden layers having 256 and 128 neurons resp. • Hidden Layer Dropout: 0.25, 0.25 resp. • Epochs:1	10%	1.5924	1.26
2	• User embedding size: 100 • Restaurant embedding size: 100 • Embedding Dropout: 0.25 • 2 Hidden layers having 256 and 128 neurons resp. • Hidden Layer Dropout: 0.25, 0.25 resp. • Epochs:3	10%	1.599	1.264
3	• User embedding size: 100 • Restaurant embedding size: 100 • Embedding Dropout: 0.25 • 2 Hidden layers having 256, 128, and 64 neurons resp. • Hidden Layer Dropout: 0.25, 0.25, 0.25 resp. • Epochs:3	10%	1.609	1.269
4	• User embedding size: 50 • Restaurant embedding size: 50 • Embedding Dropout: 0.25	10%	1.58	1.267

	• 2 Hidden layers having 256, 128 neurons resp. • Hidden Layer Dropout: 0.25, 0.25 resp. • Epochs:3			

SVD:
No of epochs = [10, 20, 30] # the number of iterations of the SGD procedure
Learning rate = [0.001, 0.003, 0.005] # the learning rate for all parameters
Regularization = [0.02, 0.05, 0.1, 0.4, 0.5] # the regularization term for all parameters

Some of the combinations and their RMSE are given below:

No of Epochs	Learning Rate	Regularization Parameter	RMSE on validation data
10	0.01	0.05	1.4153
10	0.01	0.5	1.4174
10	0.03	0.05	1.3690
10	0.03	0.5	1.3737
10	0.05	0.05	1.3454
10	0.05	0.5	1.3508
20	0.01	0.05	1.3871
20	0.01	0.5	1.3904
20	0.03	0.05	1.3379
20	0.03	0.5	1.3436
20	0.05	0.05	1.3199
20	0.05	0.5	1.3262
30	0.01	0.05	1.3691
30	0.01	0.5	1.3734
30	0.03	0.05	1.324
30	0.03	0.5	1.3292
30	0.05	0.05	1.3141
30	0.05	0.5	1.3166

- **Analysis of results**

  So while evaluating the recommendations provided for the common 50 users in all 5 approaches, we realized that since all the models were built using ratings as the only metric to evaluate similarities between users or restaurants, in all three approaches the most recommended restaurants were the one which has the highest rating.

  In collaborative filtering, since it was item based, there was a popularity bias and all the users were mostly recommended only the popular subset of the restaurants.

  In the SVD, and Neural Network approach, there were a few restaurants that were not amongst the top rated but the high rating recommendations were the ones that were common in both approaches. Hence, our recommendation system, if selected only top N recommendations, it will mostly recommend the popular/highly rated restaurants.

  In SVD, the lower 5 predictions of the top 10 recommendations were unique. The reasoning behind this could be the matrix factorization which derives relationship in rows(users) as well as columns(restaurants) of our ratings matrix.

- **Scalability**
  In all the models and algorithms that we used for recommending restaurants to the user, the primary feature we leveraged was user ratings for restaurants. However, there is a huge scope with better resources to utilize more features such as cuisines, location, cost, popularity, critics reviews, ambience, amenities and many more. We can not only utilize these features but also take more feedback from the results that we are providing to the users to improve our models. It is utmost important to keep improving and refining the existing models regularly with changing parameters and lifestyles. Furthermore, not only the models but also the dataset must be updated regularly whether by using batch updates or other ETL processes. Current algorithms each of them possesses the potential to single handedly perform better than right now and with more training and understanding of the models, we strive to keep innovating and impacting the recommendation system.

Discussion and Conclusion

- **Decisions made:**
  There were several methods of preprocessing the data and reduce dimensions. The user and restaurant details were stored as long alphanumeric strings which would only delay processing time while running the model. Therefore, we created an independent mapping of restaurants and allocated indexes which replaced the alphanumeric identifiers in the actual data. This saved space as well as made it easy to manage and process the data.
  Additionally, we also decided to estimate the missing values by using the mean values of the data. This would help us fill missing ratings and dimensions. Out of 5,200,000 user reviews we only needed the reviews given by users to the restaurant businesses. And therefore, any restaurant or user which was not associated with the reviews were discarded as outliers. This reduced the dimensionality of the data by a huge amount.
  Another important decision to make was to carefully decide on the subsets of the data. As the matrices in the model would not hold millions of columns, we had to decide how to carefully take subsets of the data that would be best simulate the trends in the data. We created chunks of data such that each user and restaurant pairing spanned in a single chunk thereby not affecting the overall accuracy.

- **Difficulties faced:**
  The most difficult part of the whole system was to process the data that in such a way that all three models would be able to use without having to modify it separately for each model. We created the user, restaurant and ratings columns in a fresh dataset that would provide the training set and test data set both. This was difficult to gauge and hence it took a long time for us to just prepare the data for our models.

- **Conclusion**
  In content-based filtering technique, we observed a high popularity bias. Only those movies that had very high ratings been recommended to almost all users. The recommender was not able to recommend any unique or new restaurants to a user because it suffers from the cold start problem since most of the restaurants are not provided with ratings.

In SVD based collaborative filtering, since we had filtered out active users, there was no cold start problem and due to matrix factorization, the model was able to recommend different businesses. However, even for SVD it was observed that the high rated restaurants were recommended to each user. So, in our top 10 recommendation, 4 restaurants were common amongst all the recommendations to different users. The recommendations were picked from unique restaurants too (i.e. the ones with lower ratings or with very few reviews) but there was no way to analyze if the recommendations might be too unique for a user.

In Neural Network Approach, most of the recommendations were the high rated restaurants but like SVD approach, the model did pick unique restaurants as well which had ratings close to user's average ratings. This model too did suffer from popularity bias while making recommendations.

From the above results we can conclude that all 3 approaches were impacted with the popularity bias as ratings were the only criteria used to build all the above-mentioned models. In our future scope, we would like to concentrate on building user and item profiles based on the metadata provided about the businesses and the users in Yelp dataset. This would help us in deriving more personalized recommendations and a higher coverage as fewer known businesses are bound to get picked if their profile attributes match the user's preferred restaurant as review count and average rating would no longer be the only deciding factor while making recommendations. To resolve the cold start problem, we plan to build a model that leverages reinforcement learning while making unique recommendations to a user.

Project Plan/Task Distribution

- **Tasks distribution/performed:**

  Data Preprocessing:

  1. Remove extra features – Maaz
  2. Extract users' details – Amit
  3. Extract businesses details – Shriya
  4. Create a set of users who reviewed the restaurants – Amit
  5. Create a set of restaurants that have reviews – Shriya
  6. Create indexes for users – Amit
  7. Create indexes for restaurants – Maaz
  8. Create final dataset with user, restaurant, ratings mapping – Amit

  Model Implementation and evaluation:

  1. Content Based Filtering – Maaz
  2. Collaborative Filtering using Neural Networks – Amit
  3. Collaborative Filtering using SVD – Shriya

  Results compilation – Shriya

  Documentation:

  1. Project Report – Maaz
  2. Project Presentation deck – Amit
  3. Presentation – Maaz, Amit, Shriya

Appendix

References:

- https://towardsdatascience.com/fast-ai-season-1-episode-5-3-collaborative-filtering-using-neural-network-48e49d7f9b36
- https://towardsdatascience.com/beginners-guide-to-creating-an-svd-recommender-system-1fd7326d1f65
- https://medium.com/@sumith.gannarapu/restaurant-recommendation-system-b52911d1ed0b
- https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26
- https://realpython.com/build-recommendation-engine-collaborative-filtering/
- https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0