Northeastern University

Data Mining Project Report

Recommendation Systems for Yelp Dataset

Aditya Priyadarshi, Abhay Kasturia, Xingxing Liu, Varun Nandu and Gautam Vashisht

Supervised by
Dr. Nate Derbinsky

December 12, 2017

1 Introduction and Related Work

A recommender system or a recommendation system is a subclass of information filtering systems that seeks to predict the rating or preference that a user would give to an item [2]. Recommendations systems have become very relevant today given the presence of e-commerce website like Amazon and Netflix as well as other platforms like Facebook and Youtube. These are utilized in a variety of areas such as movies, music, videos, news, books, research articles, search queries and products in case of Amazon. Two most common methods to build a recommendation system are collaborative filtering and content-based filtering. Collaborative filtering methods use user's past behaviors and behaviors of similar users to find items which a user might like. Content-based methods use the features of the items liked by the user to suggest similar items. There are also hybrid recommendation system which combine both of these techniques.

2 Dataset and Analysis

2.1 Dataset

The original dataset described in the Yelp Dataset Challenge 10 [1] has 4.7M reviews and 1M tips by 1.1M users for 156K businesses spread across 12 cities. The data is given in json format which include business.json, review.json, user.json, checkin.json and tip.json.Each business has name, address, star rating and textual reviews. Each individual review data consists of anonymized IDs for the business, user and review, star rating, review type, review text and votes on how useful, funny or cool the review is.

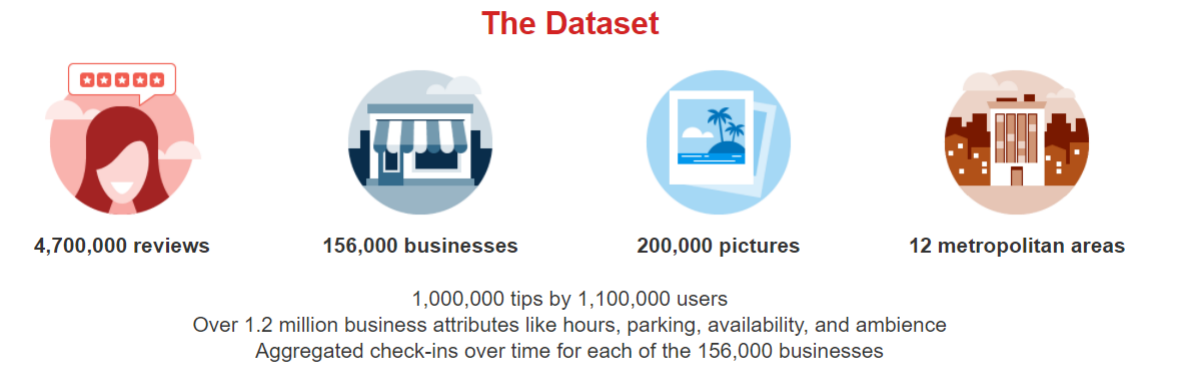

Figure 1: Dataset Details

2.2 Analysis

We did an initial analysis of the dataset. Below sections present our analysis.

2.2.1 User data

There are 1,183,362 total users whose reviews are present in the dataset. We plotted a histogram to understand the distribution of user reviews. Looking at the histogram, we can observe that most of the user have very few reviews and some top users have significant number of reviews. Majority of user have 25 or less reviews which is also shown by a mean of 23..72 and standard deviation of 80.5. The maximum number of reviews given by any user is 11656.

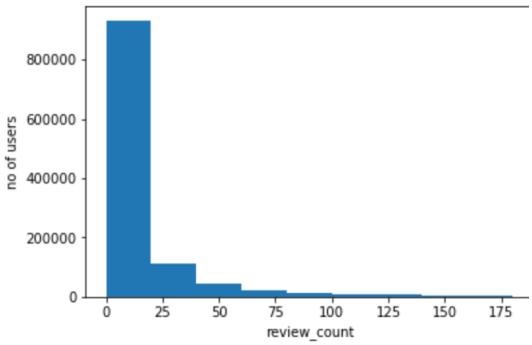

Figure 2: Review count per user

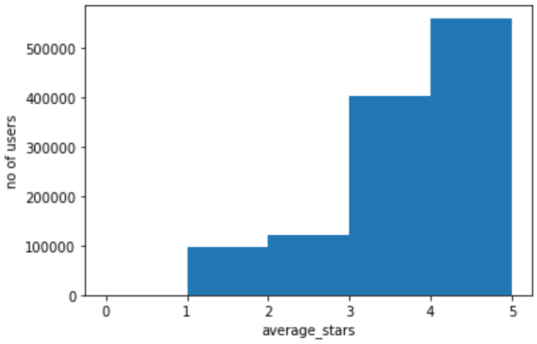

Figure 3: Rating per user

In addition to number of reviews, we also looked at distribution of star ratings given by a user. Looking at the histogram, we can observe that more users give higher rating which is shown by a median of 3.89 star rating. Mean and standard deviation for the same are 3.71 and 1.10 respectively. In order to group the reviews as positive, average and negative reviews, we have used the following method. We assume that if the rating lies in the range of (mean standard deviation, mean) which is 2.6 to 3.7, we will categorize it as average. Reviews lower than 2.6 will be considered as a negative review and anything greater than 3.7 will be considered as positive reviews with two extremes being 0 and 1.

We also did some analysis to see the user growth on yelp. User growth has started declining after an increase in users joining from 2005 to 2014 .

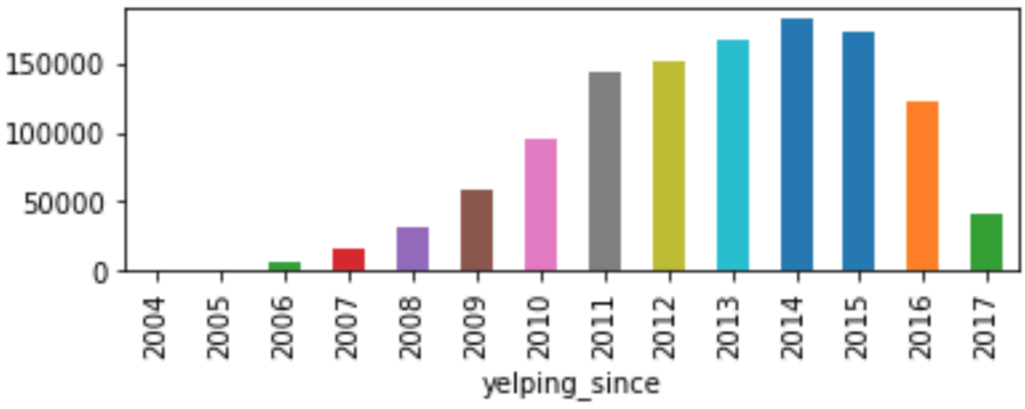

Figure 4: User Growth

2.2.2 Business Data

There are total 1,56,639 business in the dataset. We grouped business according to city and business category to determine popular cities and categories. Below pie-charts give idea about popular cities and categories.

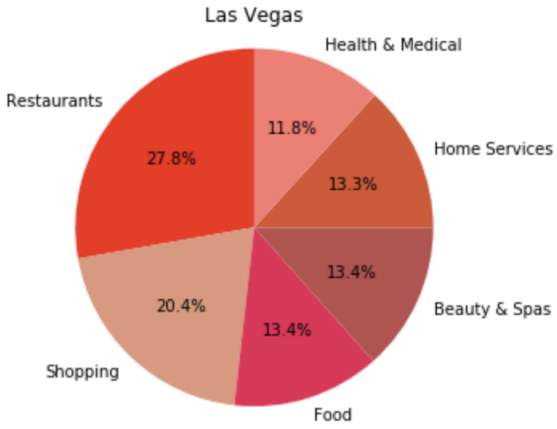

Figure 5: Top cities

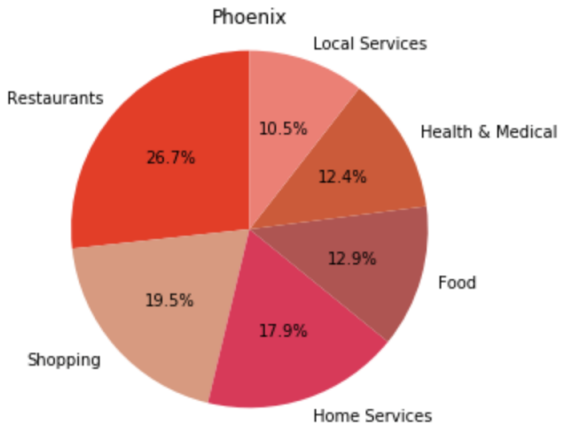

Figure 6: Top Business Categories

3 Methods

3.1 Clustering Based Approach

We use clustering to group users who have similar preferences. For clustering based approach, our inputs are user id, type of business and location of the user. The output of our system is top 20 recommendations of businesses for the above type of business.

Thus in order to build this recommendation system, we initially divide our training and testing data based on location. So, we only consider users and business from the given location. Our training data consists of user ratings for businesses from 2004 till 2015. Our testing data consists of user ratings for years 2016 and 2017.

We first get the preference of each user by using top 15 categories for the above type of business in the location provided in input. So assuming we have type of business as Restaurants, then top 15 categories could be Indian, American, Bar and Grill, Chinese, so on and so forth. We denote this in form of a vector with elements in it where each element corresponds to each of the top 15 categories. Then for each of those categories, we get all the reviews the user has given for it. We average out the review value and assign the output as a value for that category. If the user has not reviewed a particular category we give the category a value of 3.15 which is the average pf 2.6 and 3.7. 2.6 and 3.7 is the range of our average review as mentioned in the preprocessing part. Thus we create a user preference vector with 15 values.

Next, we create a business vector. This is because in order to recommend a business to a user, we need to create a similar preference vector as above for each business. we have the categories for which each business belongs to. Thus, we have a list of categories for each business. If at least 3 of the top categories are part of a business, then that business can

be recommended to a user. After filtering out valid business which we can recommend, we create a business vector exactly like the user's preference vector. It is important to know that index of any category in user preference vector should correspond to index of that category in business vector.

After this, we use Agglomerative Hierarchical Clustering[11] to cluster users. This method builds the hierarchy from the individual elements by progressively merging clusters using a distance metric. In our case we use Minkowski distance which is given by :

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

Thus we can calculate distance between user preference vectors using the formula above where n = 15. After calculating distances between every pair of user preference vector, we merge the two clusters with minimum distance and iteratively continue until we are left with only cluster. However if everything is in one cluster, then clustering does not help us that much. Thus we continue merging until we are left with 15 clusters where 15 denotes the number of categories we are using to create the preference vector. A sample dendogram after performing Clustering for Restaurants in the city of Stuttgart is given below:

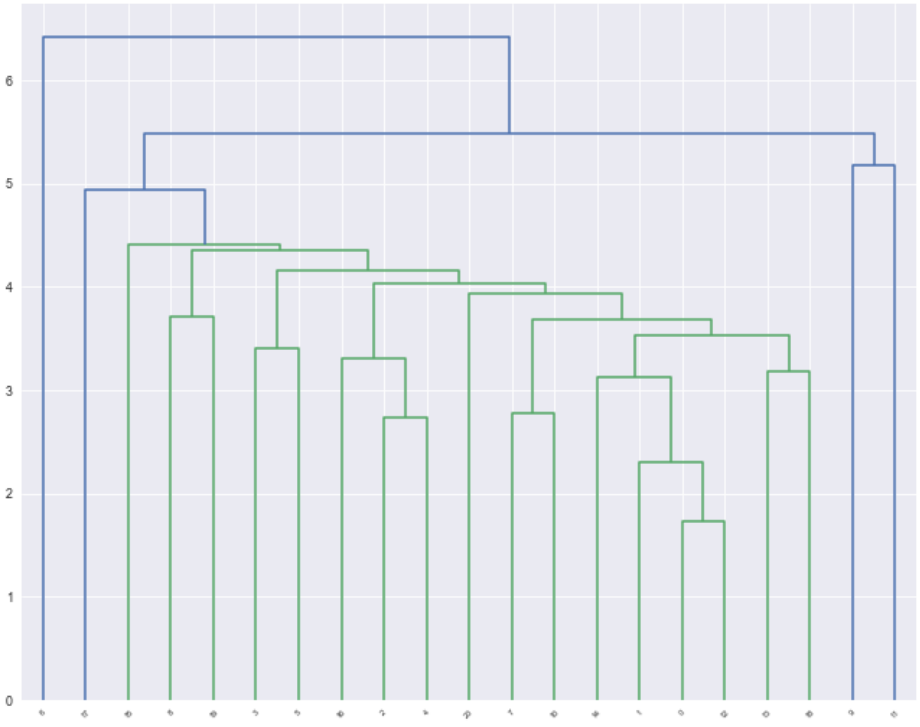

Figure 7: Dendogram of Clustering for Restaurants in the city of Stuttgart

X-Values of the dendogram represents the indexes of users and the y-axis of dendogram represents the measure of closeness between each user. After clustering, using the index we can determine which cluster our target user belongs to. Then we average out the preference vector for all users in the cluster and get an average preference vector. We then take dot product between the output average preference vector and each business vector to get similarity. Thus

if the dot product is 0 that means the two vectors are 90 degrees apart and disimilar. However, as the dot product increases, the two vectors converge and come close to each other. So, the business vector with which we get the highest dot product will be recommended first in our recommendation system.

3.2 Collaborative Filtering

Collaborative Filtering is the method to implement recommendation system. It is the way to recommend item to user 'u1' by collaborating the choice of item of other users similar to user 'u1'. We create a n X m matrix, where n is the number of users and m is the number of business, with each cell representing the rating given by a user to that particular business. Each row in matrix represent the vector of star rating given by user for all the businesses.

business_id user_id	-MhfebM0QIsKt87iDN-FNw	-cYOKJ5kbVZqzSYQlzZcqA	-lC6gIVhl7vY6W_dnw08YA	-pV9kWNoA9vyHfM_auYecA	03SYJLErY8XpNfY-qiDZcw	0nyM
---1lKK3aKOuomHnwAkAow	NaN	NaN	NaN	NaN	NaN	
--Nnm_506G_p8MxAOQna5w	NaN	NaN	NaN	NaN	NaN	
--P-Qvza7AED8gnDrZkMgA	NaN	NaN	NaN	NaN	NaN	
--ZNfWKj1VyVElRx6-g1fg	NaN	NaN	NaN	NaN	NaN	
-00MbjbaOlSrcuV7jOVRlg	NaN	NaN	NaN	NaN	NaN	

5 rows × 227 columns

Figure 8: UserVsBusiness - Stars Rating Value

While doing this, we face two issues 1. As expected, there are a lot of missing values in matrix(for items which user has not given rating). We treat missing values as the average computed after following the second step - which will always be zero. 2. There is an issue of handling the rating given by soft users and hard users i.e some users may rate the business they like with a 3 star ratings and there are some users who may rate the business they don't like with a 3 star ratings. To normalize these ratings for each user we use the centered cosine similarity. We normalize the ratings by subtracting the row mean for each user.[8]

To recommend new businesses to a target user 'target', we find the cosine similarity between all other users and 'target'. Top 'k' businesses rated with positive average star rating, by users of having cosine similarity greater than 1, will be recommended to user 'target'.[9]

3.3 Deep Learing Method

We have using Neural Collaborative Filtering [7] (NCF) approach suggested by Xiangnan He and team. We looked at various deep learning approaches towards building recommendation systems and choose this work as it was generally applicable compared to other techniques which either target music or video recommendations specifically. NCF uses multi-layers percpetron to learn use-item interaction function. It is generic and tries to generalize matric

factorization under its framework. This work focuses on the implicit feedback which indirectly reflects users behaviours.

3.3.1 Notation and Problem Formulation

Let M and N denote the number of users and items, respectively. User-item interaction matrix $Y \in \mathbb{R}^{M \times N}$ from user's implicit feedback is defined as,

$$y_{ui} = \begin{Bmatrix} 1, & \text{if interaction (user u, item i) is observed} \\ 0, & otherwise \end{Bmatrix}$$

The recommendation problem with implicit feedback is formulated as the problem of estimating the scores of unobserved entries in Y, which are used for ranking the items.

3.3.2 Neural Collaborative Filtering

A multi-layer reprentation is used to model user-item interaction y_{ui} as shown in below figure.

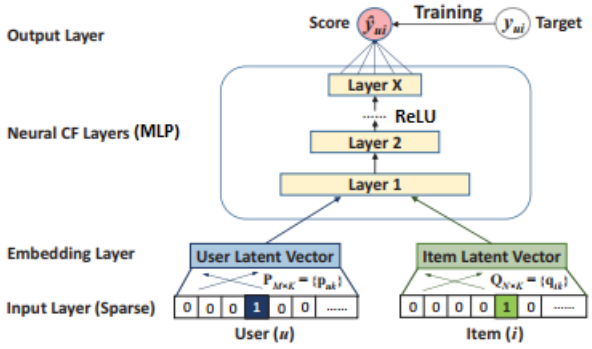

Figure 9: Neural Collaborative Filtering Model

The bottom input layer consists of two feature vectors v_u^U and v_i^I that describe user u and item i, respectively. As it is purely colloborative model, only user and item ids are taken as input in form of one-hot vectors. Above the input layer is the embedding layer; it is a fully connected layer that projects the sparse representation to a dense vector. The user embedding and item embedding are then fed into a multi-layer perceptron architecture to map the latent vectors to prediction scores. The final output layer is the predicted score $\hat{y_{ui}}$, and training is performed by minimizing the pointwise loss between $\hat{y_{ui}}$ and its target value y_{ui}

4 Experiments and Results

4.1 Clustering Based Approach

4.1.1 Calculating Similarity and Generating Recommendations

Based on recommendations received from our cluster, we use two measures to analyse our recommendation system namely Hit-Ratio (HR) and Normalised discounted cumulative gain (NDCG). If our user visits any of the recommended business in our testing data, we check what rating has he given to that business. If the rating falls in the positive review range as

mentioned in the pre-processing part, then we increment the hit score by 1. If the rating falls in the negative range, we decrement the score by 1. If the rating is in average range then it does not matter as it can be positive or negative. Thus using the above metric we calculate the Hit ratio which is given by:

$$\frac{\sum_{i=1}^{n} HitScore}{count}$$

Now this hit ratio is calculated for one particular user. We average out the hit score for all the users in the test set. Thus we get an average hit score for recommendations for all the users.

In NDCG, we first calculate gain for each recommendation. The gain refers to the actual rating user gives to each business. Cumulative gain refers to summation of each gain for each recommendation. After that we discount each gain by assigning a weight to it. Thus if the recommendation is at the top of list, the gain for that recommendation will have more weight than if the recommendation was at the bottom of the list. Thus our Discounted cumulative gain is calculated based on the formula given below:

$$DCG = \sum_{i=1}^{n} \frac{ReviewRating}{i}$$

In this case 'i' refers to the index of recommendation. In our case we are recommending top 20 business so 'i' will range from 1 to 20 inclusive. After calculating Discounted Cumulative Gain (DCG) we calculate Ideal Discounted Cumulative Gain (IDCG) which refers to for every recommendation we give a rating of 5. Thus our NDCG becomes:

$$NDCG = \frac{DCG}{IDCG}$$

After calculating NDCG for one user, we calculate NDCG for every user in the test and average it out.

Below is a table showing calculation of Hit Ratio and NDCG for top 15 cities mentioned in pre-processing.

4.2 User-User Collaborative Filtering

For the initial setup, we have worked on 100,000 rows of reviews.json file. We have created a matrix of 78276 users and 4224 businesses.We randomly choose one user to find the set of similar users of count 200. Based on positive average star ratings given by 200 users, 531 set of businesses are recommended to users.

5 Future work

1. Modify original collaborative deep learning implementation[10] of paper [**?**] to analyze our dataset.

2. All the algorithms are run on a small subset of data. Look into distributive processing framework(Spark), to perform the same on all the data or leverage the Google Cloud Platform.

Table 1: Evaluation Metrics: Hit Ratio and NDCG of top 15 Cities for Restaurants

City	Hit Ratio	NDCG
Las Vegas	0.65	0.77
Toronto	0.65	0.89
Phoenix	0.70	0.82
Charlotte	0.63	0.86
Edinburgh	1.0	0.87
Pittsburgh	0.73	0.84
Montral	0.63	0.88
Scottsdale	0.71	0.88
Cleveland	0.80	0.87
Madison	0.74	0.84
Stuttgart	0.5	0.66
Mesa	0.5	0.76
Tempe	0.48	0.79
Henderson	0.69	0.93
Mississauga	0.35	0.68
Average	0.65	0.83

3. Finalize evaluation criteria. The general approach that we plan to take would be:
   (a) Divide training and test data based on the timeline of reviews - to get past activity and future activity.
   (b) Define users in future activity as target users. Use the past activity of the target users to make recommendations.
   (c) Evaluate the recommendation as correct, if the recommendation shows up in the users future activity.
   (d) Get the total accuracy based on the same and evaluate our model!
4. Handle the problem of cold start - where a new business or a user is added. The general approach here would be to recommend the most popular business to the new user. Have to search for a way to tackle new businesses.

References

[1] Yelp Dataset Challenge `https://www.yelp.com/dataset_challenge`

[2] Recommendation System Wiki `https://en.wikipedia.org/wiki/Recommender_system`

[3] Collaborative filtering `https://en.wikipedia.org/wiki/Collaborative_filtering`

[4] Collaborative filtering Techniques `https://www.youtube.com/watch?v=h9gpufJFF-0`

[5] Movie Recommenation System `https://beckernick.github.io/matrix-factorization-recommender/`

[6] Paul Covington, Jay Adams, Emre Sargin *Deep Neural Networks for YouTube Recommendations*, ACM 2016.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua*Neural Collaborative Filtering*, ACM 2017

[8] Lecture on CF - Stanford `https://www.youtube.com/watch?v=h9gpufJFF-0&t=854s`

[9] Recommendation System Netflix `https://www.youtube.com/watch?v=bLhq63ygoU8&t=5598s`

[10] CDL Implementation `https://github.com/hexiangnan/neural_collaborative_filtering`

[11] Szkely, G. J.; Rizzo, M. L. (2005). "Hierarchical clustering via Joint Between-Within Distances: Extending Ward's Minimum Variance Method". Journal of Classification