Yelp Food Recommendation System

Muhammad Tayyab and Shivam Bijoria

University of Texas at Arlington

muhammad.tayyab@mavs.uta.edu and shivambijoriya@gmail.com

Abstract

Our aim is to develop a recommendation system which helps users by recommending restaurants using data collected by yelp. Using Yelp's dataset, we extract customer, restaurant profiles and ratings given for suggesting recommendations. In particular, we implement naïve baseline, singular value decomposition, hybrid cascade of K-nearest neighbor clustering, weighted bi-partite graph projection and its variants with clustering or multi-step random walk or both. Using Root Mean Squared Error, we then evaluate and compare the algorithms' performances.

INTRODUCTION

A vast database of reviews, ratings, and general information provided by the community about businesses, Yelp provides consumers with a myriad of options and information even when searching for an especially specific service or goods niche. However, although all required information may be present to make an informed choice, it is often still difficult by just looking at the raw data. Reading all the reviews of a single business alone is time consuming and requires more effort than the average user is willing to expend. As a result, we believe users could greatly benefit from a recommendation system.

Recommendation systems have historically been created for various Machine Learning applications in numerous disciplines. One such example is social networking sites such as Facebook that utilize recommendation systems to suggest friendships to users. Music and media applications such as iTunes and Spotify also utilize similar machine learning and recommendation logic to suggest various songs, videos, movies, etc. to users based off their previous choices and taste. Given this general theme, our project focuses on creating a recommendation system for Yelp users in application to potential food choices they could make.

The rise of the popular review site Yelp has led to an influx in data on people's preferences and personalities when it comes to being a modern consumer. Recommendation systems that can identify a user's preferences and identify other similar users' and/or restaurants that match his/her preferences can make this problem easier. Specifically, we aim to build a recommendation system that will enable us to make sophisticated food recommendations for Yelp users by applying learning algorithms to develop a predictive model of customers' restaurant ratings.

Methods and Data

Data: We have used yelp data available at- https://www.yelp.com/dataset_challenge/
Data files which we have utilized for the project are business. json, review.json, user.json.

Evaluation Metric

After researching the kinds of evaluation metrics commonly used to evaluate recommendation systems, we chose to evaluate our system based on the root mean square error (RMSE) a through k-fold cross validation. The paper we chose also used similar metrics which made our comparison consistent. There is a research paper by Microsoft that explains how RMSE and MAE are historically the most famous evaluation techniques used for recommendation systems. Following is the formula:

$$\text{RMSE} = \sqrt{\frac{1}{|S|}\sum_{(u,f)\in S}(\hat{r}_{uf} - r_{uf})^2}$$

Where S is the set of all ratings given by user u to a restaurant f, $r_{u,f}$(cap) is the predicted rating and $r_{u,f}$ is the actual rating given(a tuple entry in review.json file).

Data Preprocessing

The yelp dataset provided on http://www.yelp.com/dataset challenge/. The data is divided into 5 sets: businesses, reviews, users, tips and check-ins. For our project we only used businesses, reviews and users datasets. To further reduce the size of data we only worked on restaurant businesses and neglected all the others. However there were still lot of problems with the dataset. Data was too sparse; more than 90 percent of data was empty. This was a major issue when building recommendation system which led to other problems such as cold start. Cold start is a situation when a recommender system doesn't have any historical information about user or item and is unable to make personalized recommendations. Another problem that surfaced was there were large number of grey sheep unpredictable ratings. There were many users whose profile could not match other users. So in order resolve these issues we spent lot of time on the preprocessing of data.

We first converted the entire data from JSON format to CSV format using a python script. Then we dealt with the missing values by using imputation techniques using average of the corresponding features and using it for the missing values. We used Weka tool for imputation. Since data was large it had what we call curse of dimensionality. Our next step was to select important features from the three datasets. By using techniques such as PCA, extract important features from the data. By combining all the features we were able to reduce data to 9 features (attributes) and 26140 instances.

Data before processing:

business

```
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)
        },
        ...
    },
    'attributes': {
        (attribute_name): (attribute_value),
        ...
    },
}
```

review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

user

```
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}
```

Data after preprocessing:

A	B	C	D	E	F	G	H	I	J
category	user_id	review_id	stars	biz_name	biz_stars	business_	biz_revie	user_rivew_count	
generalize	PUFPaY9K	Ya85v4eq	4	Mr Hoagie	3.5	5UmKMjU	7	60	
generalize	qiczib2fO	pVMIt0a_	3	Mr Hoagie	3.5	5UmKMjU	7	52	
generalize	qEE5EvV-f	AEyiQ_Y4	2	Mr Hoagie	3.5	5UmKMjU	7	26	
American	LWbYpcar	6w6gMZ3i	5	Emils Lour	4.5	mVHrayjG	26	29	
American	8fApIAMH	3Es8Gsjks	5	Emils Lour	4.5	mVHrayjG	26	34	
American	uK8tzraOp	KAkcn7oC	4	Emils Lour	4.5	mVHrayjG	26	46	
American	pdHC0oAc	zyn_Libz9	5	Emils Lour	4.5	mVHrayjG	26	23	
American	tAKjY3bQ	uf61rPucu	5	Emils Lour	4.5	mVHrayjG	26	128	
American	Kq8-FUG7	xY2gjy49d	4	Emils Lour	4.5	mVHrayjG	26	21	
American	a0ULPTNi	1W1vq7D	5	Emils Lour	4.5	mVHrayjG	26	37	

Naïve Baseline

For predicting baseline rating of user u for an unrated business b,we utilized the formula:

$$\hat{r} = \mu + b_u + b_f$$

Where mu is the average of all ratings given by any user to any business, b_u is the difference between average of all ratings given by user u and mu, b_f is the difference between average of all ratings received by restaurant f and mu.
Performance: We received average rmse of around 2.3.

Weighted Bipartite Graph Projection:

$$\hat{r}_{u,b} = \overline{r}_u + \kappa \sum_{j=1}^{n} sim(u,j)(r_{j,b} - \overline{r}_j)$$

$$\overline{r}_u = \frac{1}{B_u} \sum_{j \in B_u} r_{u,j}$$

$$rp(u,v) = \sum_{b \in B} \frac{r_{u,b}}{R_u} \frac{r_{v,b}}{R_b}$$

In this approach, we have two sets which comprise a graph. One set is set of all users and another of all restaurants.
Edges from one set to other are drawn with weights.
To find the probable rating of a user u for an unvisited restaurant b, we use above formula.

All users that have visited the restaurant b are considered. Similarity between each considered user v and user u is taken into account. This similarity is called recommendation power of any arbitrary user v on user u. It is summation of multiplication of two ratios. Each term in the summation is a restaurant rated by both

users-user u and v.each ratio is rating given by that user divided by total ratings given by that user.

Performance: We received average rmse of around 1.50.

Clustered Weighted Bipartite Graph Projection

It is a variation of above-mentioned algorithm. In the dataset we found out that most users have rated very few restaurants. So, probability that any two users have rated the same restaurant remains relatively low. So, what we have done, is that we have is divided the users in clusters, in same way we have done in knn approach. So, while estimating for a user's predictive rating, we consider only the user's present in that user's cluster because these users will definitely have common restaurants and also it makes more sense that eating habits of these users will be more similar than as compared to eating habit of a user in different cluster.
Performance: We received average rmse of around 1.17.

Multistep random walk Weighted Bipartite Graph Projection

In previous approaches we have considered recommendation power user v has on user u. This was a two step walk in our bipartite graph from one set to another. We can expand this approach by using 4,6,8 or more random walks. In this approach recommendation power v yields on u,goues through more than 1 restaurant and 1 or more than 1 user. Below is the formula given for the approach:

$$rp(u,v) = \sum_{b \in B} \frac{r_{u,b}}{R_u}\frac{r_{v,b}}{R_b} + \alpha\Big(\sum_{b_1,b_2 \in B} \frac{r_{u,b_1}}{R_u}\frac{r_{k_1,b_1}}{R_{b_1}}\frac{r_{k_1,b_2}}{R_{k_1}}\frac{r_{v,b_2}}{R_{b_2}}\Big) + \alpha^2\Big(\sum_{b_1,b_2 \in B} \frac{r_{u,b_1}}{R_u}\frac{r_{k_1,b_1}}{R_{b_1}}\frac{r_{k_1,b_2}}{R_{k_1}}\frac{r_{k_2,b_2}}{R_{b_2}}\frac{r_{k_2,b_3}}{R_{k_2}}\frac{r_{v,b_3}}{R_{b_3}}\Big) + \ldots + \alpha^n\Big(\sum_{b_1,\ldots,b_n \in B} \frac{r_{u,b_1}}{R_u}\frac{r_{k_1,b_1}}{R_{b_1}} \ldots \frac{r_{k_{n-1},b_n}}{R_{k_{n-1}}}\frac{r_{v,b_n}}{R_{b_n}}\Big)$$

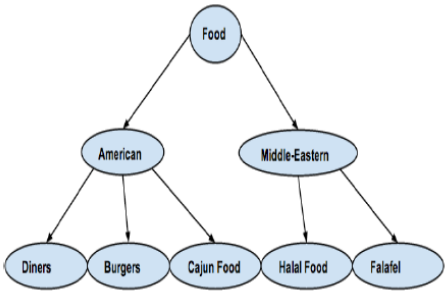
Performance: We received average rmse of around 1.25.

Clustered Multi-step Random Walk Weighted Bipartite Graph Projection

In this variation of Walk Weighted Bipartite Graph Projection, we use both the variation mentioned above simultaneously-clustering and multi-step random walk. This was the best approach in all the 4 algorithms-the Weighted Bipartite Graph Projection and its three variations.
Performance: We received average rmse of around 1.10.

Hybrid Cascade KNN Clustering

The sparseness of the ratings makes it very unlikely that two users rated the same restaurant. So in order to overcome this we first cluster the restaurants based on the general food category they belong to . This clustering is based on knowledge technique.



The figure above shows an example how business clustering is done. The food is divided into different types American, Middle Eastern which are again subdivided into categories like diners burgers Cajun food etc. The way we cluster restaurant is we take the common category for e-g, American and map all the restaurants that serve food related to this category into one cluster.

Next we cluster the users based on which category of restaurants the reviewed the most. For example all the users who reviewed American restaurants the most fall in one group or cluster, similarly all users who reviewed restaurants belonging to Mediterranean category are grouped into one cluster.

After clustering users and restaurants we map the test user to the user cluster it belongs. Then we find k nearest users with in that users that are most similar to our test user. We tried doing this by applying knn algorithm but as the number of users were in such large number that the complexity was very high and we did not have enough computational power. So to solve this issue, we randomly searched for k users within user cluster and then applied knn algorithm on those users. The result of this gave us the most similar users to our test user.

We then find restaurants that similar users have rated the best and test user has not. The ratings of these restaurants are calculating by averaging all the ratings given by all the users within that cluster to that restaurant. The following formula shows this:

$$\hat{r}_{u,f} = \frac{1}{|R_{C_u,C_f}|} \sum_{k \in R_{C_u,C_f}} k$$

In the expression above, RCu, Cf is the set of all ratings by users in Cu of businesses in Cf. The rmse result we get after implementing this algorithm is 1.55.

SVM

The yelp business data gives large number of attributes that describe every business, in our case every restaurant. So selecting users who reviewed at least 20 restaurants, we developed another dataset. For simplicity we selected user that had reviewed 416 Restaurants and used it as our test case. These restaurants have attributes like has-wifi,open on Sundays, has tv, parking etc. and the classification label as ratings.

So we apply SVM to get predicted rating of the restaurants whose attributes we know for our test user. For each user, using features on the restaurants they reviewed, we learnt a hyperplane that reflects their preferences. Hence, how the user rates the restaurant corresponds to the distance of the restaurant to their hyperplane. We then recommend restaurants whose predicted ratings classified by SVM are highest. The implementation of this algorithm was done using Weka . We got rmse value of 0.78 after implementation

Results

Naïve baseline

Accuracy of recommendation:

Total examples	Correct prediction	Accuracy=correct pred./total ex.
281	121	.4306
278	112	.4028
139	70	.5035
217	100	.4608
98	45	.50
200	80	.4
304	130	.4276
368	151	.41032
1885	809	.4292

KNN

Predicted values and actual values

```
r_cap:  2.375
r:  2.0
r_cap:  3.5454545454545454
r:  4.0
r_cap:  4.248648648648649
r:  4.0
r_cap:  3.85
r:  3.0
r_cap:  4.227941176470588
r:  5.0
r_cap:  3.522727272727273
r:  4.0
r_cap:  3.090909090909091
r:  5.0
r_cap:  4.449704142011834
r:  5.0
r_cap:  4.25
r:  4.0
r_cap:  4.001317523056653
r:  5.0
r_cap:  4.1866666666666665
r:  4.0
r_cap:  4.203703703703703
r:  4.0
r_cap:  3.7938931297709924
r:  5.0
r_cap:  3.2
r:  1.0
r_cap:  2.5
r:  1.0
```

Weighted Bipartite graph Projection

```
^[[Ashivam@shivam:~/Desktop/wbp$ python program.py
the root_mean_square_error is 1.50
----------------------------------------
the root_mean_square_error is 1.52
----------------------------------------
the root_mean_square_error is 1.49
----------------------------------------
the root_mean_square_error is 1.49
----------------------------------------
the root_mean_square_error is 1.51
----------------------------------------
the root_mean_square_error is 1.50
----------------------------------------
the root_mean_square_error is 1.51
----------------------------------------
```

Multistep random walk Weighted Bipartite graph Projection

```
shivam@shivam:~/Desktop/wbpc$ python program.py
the root_mean_square_error is 1.28
----------------------------------------
the root_mean_square_error is 1.24
----------------------------------------
the root_mean_square_error is 1.22
----------------------------------------
the root_mean_square_error is 1.26
----------------------------------------
the root_mean_square_error is 1.29
----------------------------------------
the root_mean_square_error is 1.26
----------------------------------------
the root_mean_square_error is 1.24
----------------------------------------
the root_mean_square_error is 1.22
----------------------------------------
the root_mean_square_error is 1.23
----------------------------------------
the root_mean_square_error is 1.25
----------------------------------------
```

```
Test mode:    10-fold cross-validation

=== Predictions on test data ===

    inst#     actual  predicted        error
        1          5      3.808       -1.192
        2          4      3.714       -0.286
        3          3      4.247        1.247
        4          4      3.283       -0.717
        5          4      4.061        0.061
        6          3      3.807        0.807
        7          4      3.503       -0.497
        8          4      4.036        0.036
        9          3      3.935        0.935
       10          4      3.728       -0.272
       11          3      3.83         0.83
       12          4      3.994       -0.006
       13          4      3.619       -0.381
       14          4      3.571       -0.429
       15          3      3.687        0.687
       16          4      3.262       -0.738
       17          4      3.651       -0.349
       18          4      3.942       -0.058
       19          3      4.155        1.155
       20          4      3.392       -0.608
```

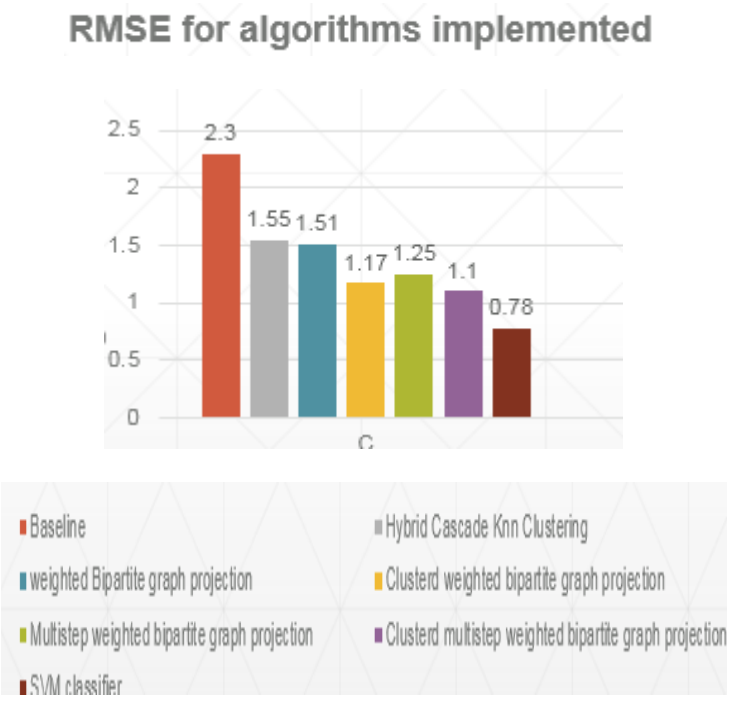
SVM

Predicted and actual ratings
And rmse and mae values.

```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.1427
Mean absolute error                      0.6253
Root mean squared error                  0.78
```

Comparison



Conclusion

If we compare the result, the SVM classifier gives the least root mean square error.But in SVM We did not take into account the case where users reviewed the same restaurant multiple times.
In hybrid cascaded KNN clustering we are limited to predictions within the category where user has previously reviewed the most.
In case of four weighted bipartite algorithms, the multistep clustered weighted bipartite graph projection gives the least rmse of around 1.10.it is because it includes both the improvements-clustering and multistep walking.

References

- https://www.yelp.com/dataset_challenge/
- https://en.wikipedia.org/wiki/Support_vector_machine
- https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- http://cs229.stanford.edu/proj2013/SawantPai-YelpFoodRecommendationSystem.pdf
- Gunawardana A., Shani G, Evaluating Recommendation Systems, <http://research.microsoft.com/pubs/115396/evaluationmetrics.tr.pdf>.