Restaurant Recommender System[1]

Deshu Venkatesh, Vaibhav	Gomudurai Pandian, Tharun Niranjan	Iyer, Vignesh	Manikarnike, Mukund	Singh, Bharat	Syed Mohammad, Yousuf Hussain
Master of Science in Computer Engineering, ASU	Master of Science in Software Engineering, ASU	Master of Computer Science, ASU	Master of Computer Science, ASU	Master of Science in Software Engineering, ASU	Master of Science in Software Engineering, ASU
vdvenkat@asu.edu	tgomudur@asu.edu	vkiyer@asu.edu	mmanikar@asu.edu	bsingh21@asu.edu	syousufh@asu.edu

Table of Contents

[1] Source Code for the project is available at github.com/bks2009/RecommendationSystem.

List of Tables

List of Figures

ABSTRACT

As the number of users interacting with web based applications increase, the number of such applications increase and there is going to be a constant demand for fine-tuning how content is served up to each user with the aim of maximizing user satisfaction. This paper illustrates how such custom content can be served up to users by using a recommender system. Certain techniques like collaborative filtering, content based recommender are explored and their performance is compared by using measures like root mean square error and mean absolute error. The recommender system uses restaurant data from Yelp provided by Yelp as part of Round 7 of the Yelp dataset challenge [1].

Keywords

Recommender, Collaborative-Filtering, Content-based, Yelp-Data, Feature-Selection

1. INTRODUCTION

In the past decade and a half, the internet has exponentially grown, resulting in a large number of users and large amount of content generated by every user alone and also through inter-user interactions. As is the trend now, users look for a customized set of results to be output of each search query that they provide. Given the amount of content available and how it is predicted to grow, this would very soon become a common requirement on the internet. The sole purpose of a recommender system is to customize the content served to every user based on the requirements of each user. The following subsections talk about the dataset used for the purpose of recommendation and the organization of further sections of the paper.

Yelp Dataset Challenge [2]

Yelp collects a large amount of data for businesses that are registered on the platform. The data collected includes the kind of business run by each organization, the amount of check-ins at the particular place, number of reviews for the same, the user interest in the business and more such information. The Yelp dataset challenge has been ongoing since March 2013 in several rounds. The data used for this project is that made available for Round 7 of the challenge which started in February 2016 and goes on until June 2016. Some of the topics that the challenge hopes to be addressed through submissions are estimation of culture trends, location mining and urban planning, seasonal trends and many more such aspects. This project chooses to implement a restaurant recommender system using the provided dataset.

Yelp Dataset

The dataset consists of several components, which are shown in **Table 1-1**. Each of these components have several attributes that describe them.

Component	Record Size	Size on Disk (GB)
Business	206,534	0.064
Review	8,102,234	1.8
User	552,340	0.22
Check-in	55,570	0.024

Component	Record Size	Size on Disk (GB)
Tip	606,820	0.11
Photos	Not evaluated because it isn't used.	

Table 1-1 Dataset Description

Data items in each of these components cover data coming from all regions in the world. The following few subsections describe the structure of each of these components and how many data items each of these components contain. Each of these components are provided in JSON format. The data provided as part of Photos isn't used for recommendation purposes as part of this paper.

Business

This JSON contains information about businesses like their primary identification, the neighborhood they're in, the category they fall under, the average star rating given to the business by users and many more. The data item takes the format as shown in **Table 1-2**.

```
'type': 'business',
'business_id': (encrypted id),
'name': (business name),
'neighborhoods': [(hood names)],
'full_address': (localized address),
'city': (city),
'state': (state),
'latitude': latitude,
'longitude': longitude,
'stars': (star rating, round to 0.5),
'review_count': review count,
'categories': [(category names)]
'open': True / False (Closed/Open),
'hours': {
    (day_of_week): {
        'open': (HH:MM),
        'close': (HH:MM)
    },
    ...
},
'attributes': {
    (attr_name): (attr_value),
    ...
},
```

Table 1-2 Business JSON

Review

This JSON contains information about every review that every user has given every business that is on yelp. In addition to text reviews, it also contains other details like the star rating, number of votes for the particular review and so on. The data takes the format shown in **Table 1-3**

```
{
    'type': 'review',
    'business_id': (encrypted id),
    'user_id': (encrypted user id),
    'stars': (star rating, round to 0.5),
    'text': (review text),
    'date': (date, like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

Table 1-3 Review JSON

User

This JSON contains information about every user including the user ID, the number of reviews contributed by the user and many more details. The data item takes the format as shown in **Table 1-4**

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

Table 1-4 User JSON

Check-in

This JSON contains information about the check-ins made on Yelp to every business, the time at which it is done and more such information. The data item takes the form as shown in **Table 1-5**

```
{
    'type': 'checkin',
    'business_id': (encrypted id),
    'checkin_info': {
        '0-0': (number of checkins from 00:00 to 01:00 on all Sundays),
        '1-0': (number of checkins from 01:00 to 02:00 on all Sundays),
        ...
        '14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),
        ...
        '23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)
    }, # if there was no checkin for a hour-day block it won't be in dict
}
```

Table 1-5 Check-in JSON

Tip

Yelp contains a feature where users can provide tips to similar users based on what they're looking for. This JSON contains information about each tip that every user has provided. The data item is as shown in **Table 1-6**

```
{
    'type': 'tip',
    'text': (tip text),
    'business_id': (encrypted id),
    'user_id': (encrypted user id),
    'date': (date, like '2012-03-14'),
    'likes': (count),
}
```

Table 1-6 Tip JSON

Paper Organization

The following sections of this paper are organized as follows

- Section 2 provides a description of the approaches used to build a recommender system and the specifics of the implementation techniques used. It also describes how a subset of the dataset was chosen from the large dataset provided by Yelp.
- Section 3 presents certain theoretical analyses of all the algorithms and the results obtained from each of the methods used to carry out the process of recommendation.
- Section 4 provides a few conclusive points on the project including strong, weak points of the approaches and key takeaways from the project.
- Section 5 acknowledges all the people and sources that helped in the betterment of the results obtained for the project.
- Section 6 provides a few pointers on possible work that can be done in this regard in the future.

2. DESCRIPTION

2.1 Problem Formulation

The problem formulation of a typical recommender system is as shown in **Table 2-1**

	R_1	R_2	R_3	R_4
User 1	4.0	?	3.6	4.5
User 2	3.5	5.0	?	2.5
User 3	?	2.3	3.2	2.5

Table 2-1 Recommender System Problem Formulation

The problem here is, given a set of ratings for each user for each restaurant and given that a user hasn't rated a few restaurants, to predict the possible ratings a user would give for a restaurant which he/she hasn't rated yet. By doing so, one would be able to find the possibility of recommending the particular restaurant to the user. This is a simple presentation of the recommender problem and can become more and more complex as we add features through which each restaurant is described. This paper will describe all the approaches starting from the simple approaches to complex ones.

2.2 Recommender algorithms

This section describes the recommender algorithms used in this paper.

2.2.1 Benchmarking algorithm

In recommender systems, there isn't any ground truth available to compare with and hence evaluation of the predictions becomes difficult and sometimes not possible. Hence, the approach used to solve this problem is to use a few benchmarking algorithms. The usage of a benchmarking algorithm entails the following

1. Estimate error measures on the provided data using trivial algorithms by dividing the training set provided into a training set and a validation set.
2. These error measures can then be used as a benchmark against which all better algorithms are compared against which will provide a way to measure whether the recommender system does a good job or not.

There are of course other ways to measure the success of a recommender system like

1. User Satisfaction from the provided recommendations
2. User engagement with the content served up based on the recommendations provided.

However, the recommender system designed for the purpose of this paper has a static dataset provided by Yelp and has no means to measure user satisfaction on the results provided. Hence, a few benchmarking algorithms were employed. The overall summary of the benchmarking approaches are illustrated in **Figure 2-1**

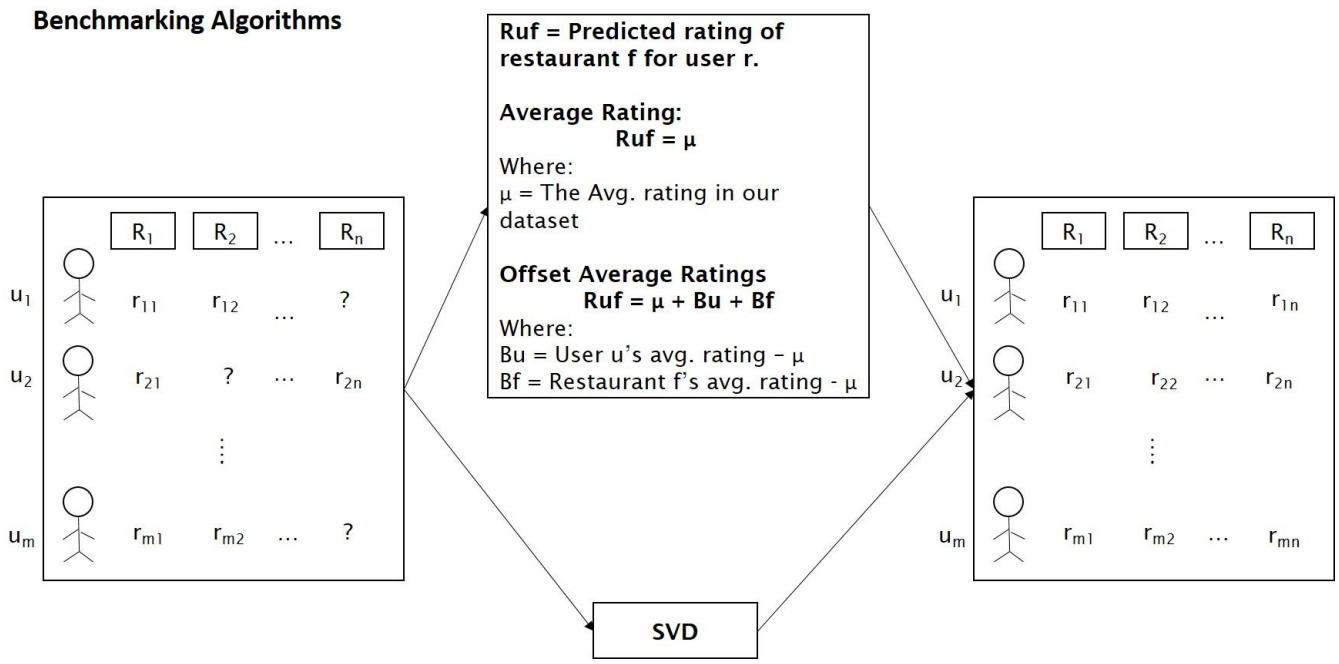

Figure 2-1 Benchmarking Algorithms

A summary of each of the benchmarking algorithms is as follows

1. Average Rating
    a. Predicts the average ratings of all available ratings for the ratings that are missing.
    b. This is a very basic approach since it doesn't take into consideration any user or restaurant parameters.
2. Offset Average Ratings[1]
    a. Computes average rating of all available ratings.
    b. Computes average rating per user and offset of the same from the whole average.
    c. Computes average rating per restaurant and offset of the same from the whole average.
    d. Predicts the missing ratings as a summation of the whole average and the corresponding offsets for a given user restaurant pair.
3. Singular Value Decomposition
    a. The user-restaurant matrix is a sparse matrix where each element represents the rating a user has given to a business. The restaurant average was replaced with the missing values i.e. businesses for which the user has not rated yet. After this each element in the matrix was normalized with the user average rating. This is similar to the approach discussed in [8].
    b. The existing numpy SVD algorithm [13] was used to reduce the dimensionality of the user-restaurant matrix. K values from 2-21, 25, 50, 100 were used. These matrices were then reconstructed, and error was calculated against the test set. The error rates with respect to k values for SVD is shown in **Figure 2-2.** The minimum error was achieved for k = 2.

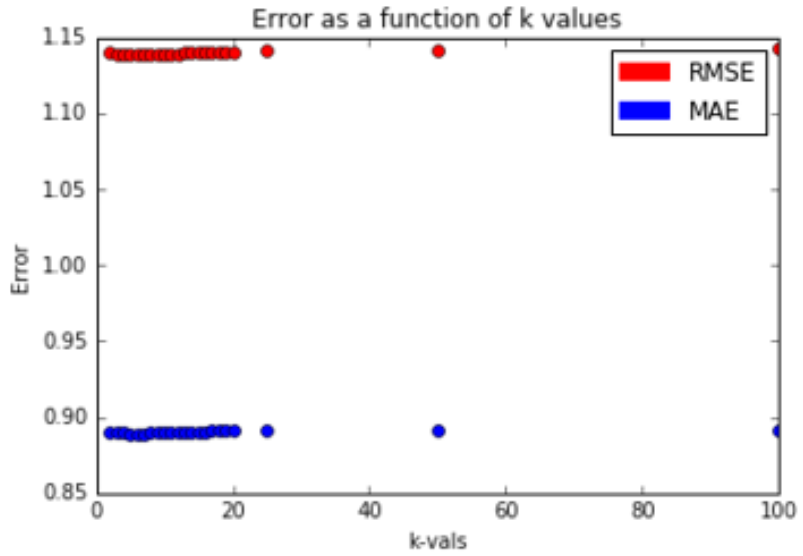

Figure 2-2 SVD Performance Variation

The results of each of the benchmarking methods, prediction using average rating, prediction using offset average ratings and SVD are detailed later in the paper.

2.2.2 Collaborative Filtering

The collaborative filtering approach tries to solve a typical recommender system problem based on the ratings as shown in **Table 2-1**.

Collaborative filtering can solve the recommender problem using User-based or Item-based similarity approaches. **Figure 2-3** illustrates the approach taken to solve the recommender problem using both approaches and provides the prediction model that is used to predict the rating of an item for a particular user.

2.2.3 Large Scale Collaborative Filtering

For the purposes of this project, the approach taken to perform collaborative filtering was an Alternating Least Squares with weighted λ regularization approach which is a parallelized collaborative filtering algorithm running on a distributed system. A summary of the algorithm is shown in **Figure 2-3**

Some of the challenges that one faces with collaborative filtering are

1. It is highly unlikely that a given user would have rated all restaurants in the entire dataset and hence the rating matrix would be extremely sparse.
2. It is quite possible that the distribution of ratings per user in the training set and test set may differ. So, it becomes difficult to predict ratings for a user who is sparsely represented in the training set.

ALS handles sparse data and scales very well [10]. The gist of ALS algorithm as explained in [10] is as follows

1. A matrix M is initialized with an average rating for every restaurant as the first row and randomized small values for the other entries in the matrix.
2. Similar to the EM algorithm, M is fixed and solved for U (the user feature matrix) which is by minimizing the objective function. The object function is nothing but the sum of squared errors.
3. Similarly, U is fixed and M is computed.

4. The process is repeated until convergence. The convergence condition can be a state where the change in RMSE is below a certain threshold.

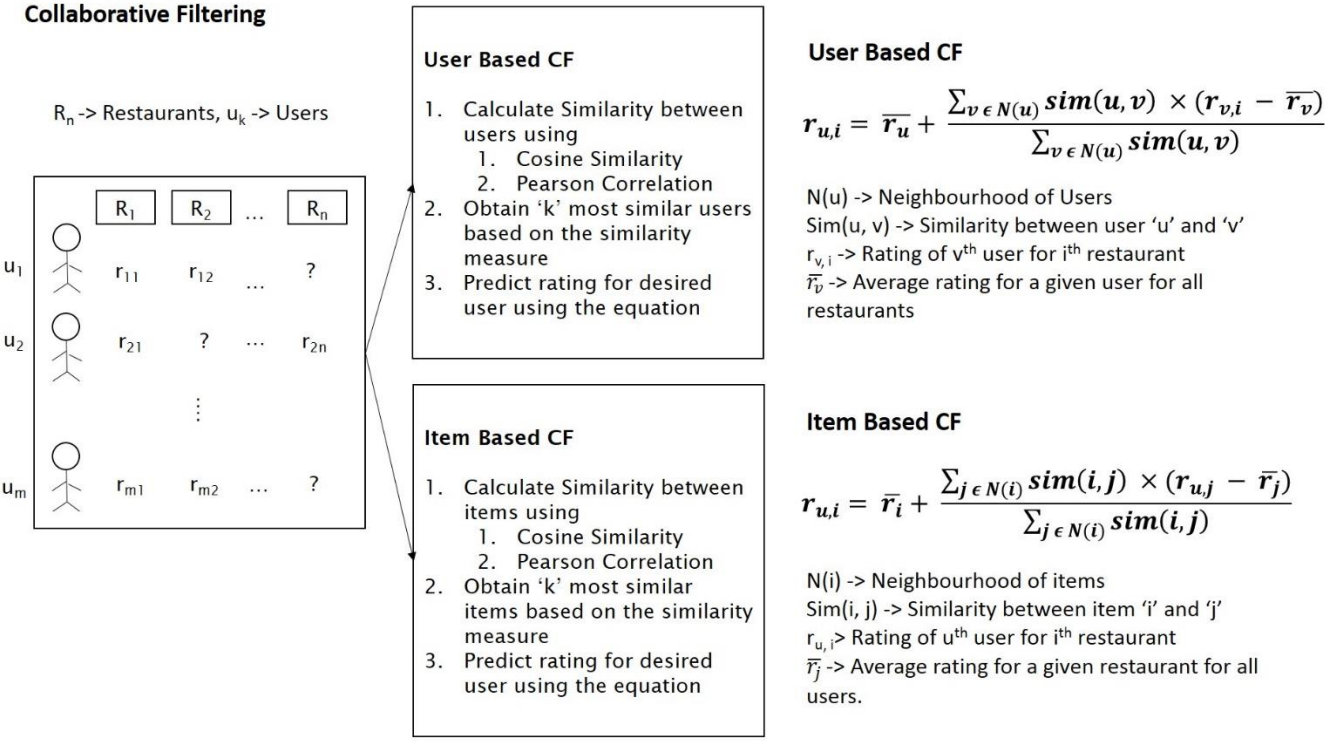

Figure 2-3 Typical Collaborative Filtering System

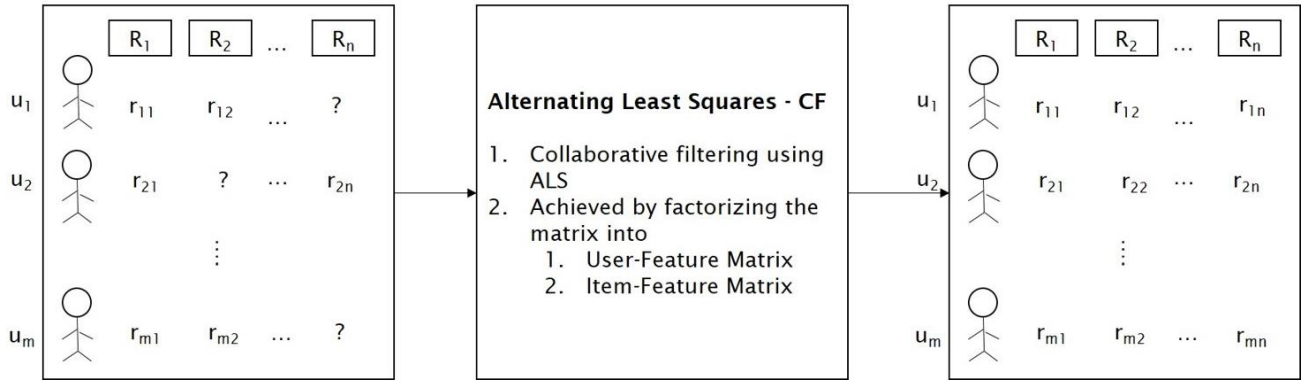

Figure 2-4 Parallelized Large Scale Collaborative Filtering

2.2.4 Content-based Filtering

The content based filtering mechanism takes the features available for an item into consideration while computing a prediction for an unknown rating. The summary of the approach is that, given a set of user ratings for all restaurants and a feature vector for each restaurant, it learns a parameter vector for each user

and predicts the rating as the dot product of the learnt parameter vector and the feature vector. The algorithm is as illustrated in **Figure** 2-4

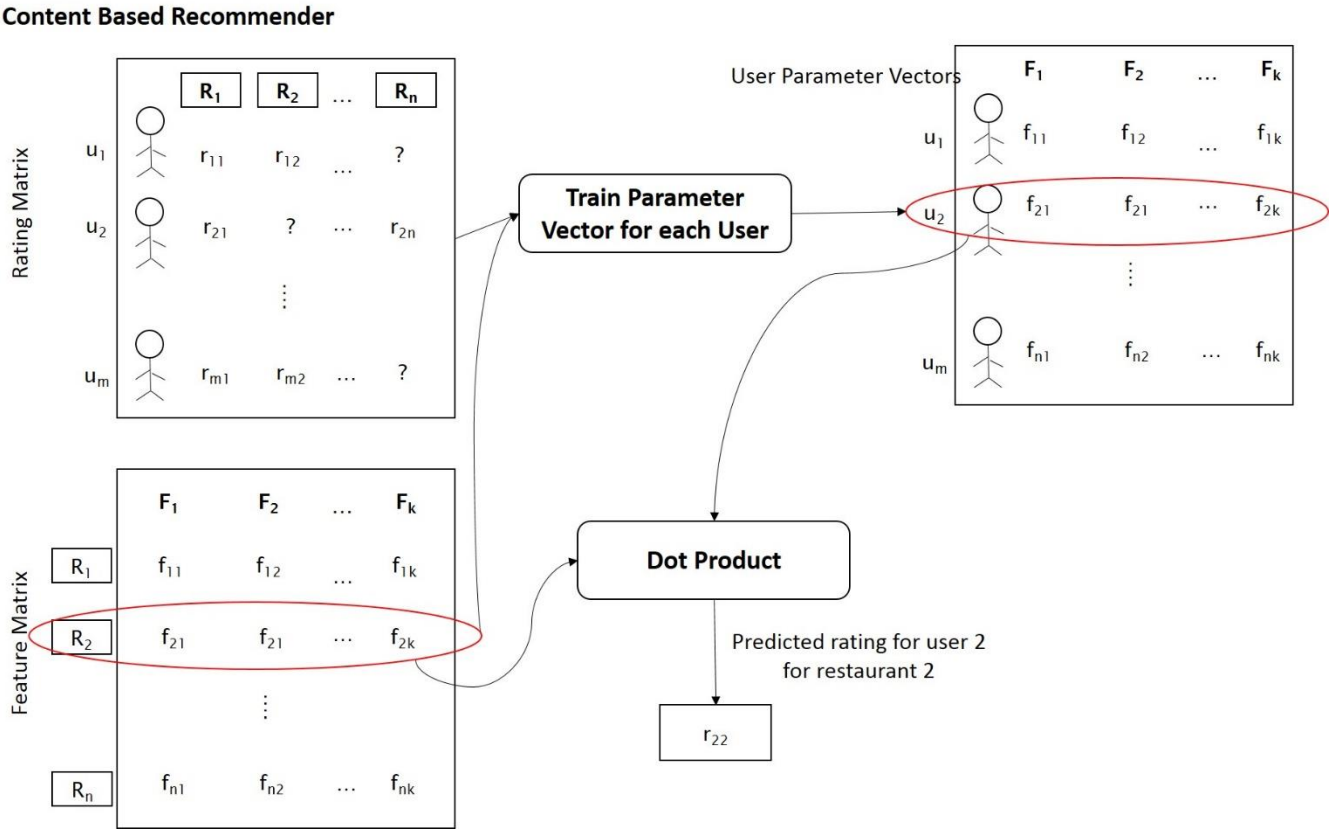

Figure 2-5 Content Based Recommender System

2.2.5 Location Aware Recommender System

The location aware recommender system is a variation of the algorithm mentioned in [5]. This system acts as an add-on to one of the recommender algorithms mentioned above by providing top k nearest restaurant recommendations. The algorithm is as illustrated in **Figure 2-6.**

The summary of the algorithm is as follows

1. Given a user latitude and longitude, the algorithm extracts the top-k nearest restaurants and their distances.
2. The algorithm optimizes the performance by using a k-d tree to extract top-k nearest restaurants.
3. These distances are later used as travel penalties for the particular restaurant and to recommend restaurants that are closest to a user.
4. The variation in the algorithm proposed by [5] and the algorithm implemented for this project is that this algorithm barely uses the distances scaled to a range of 0-1 as penalties which will be subtracted from the rating predicted using other algorithms.

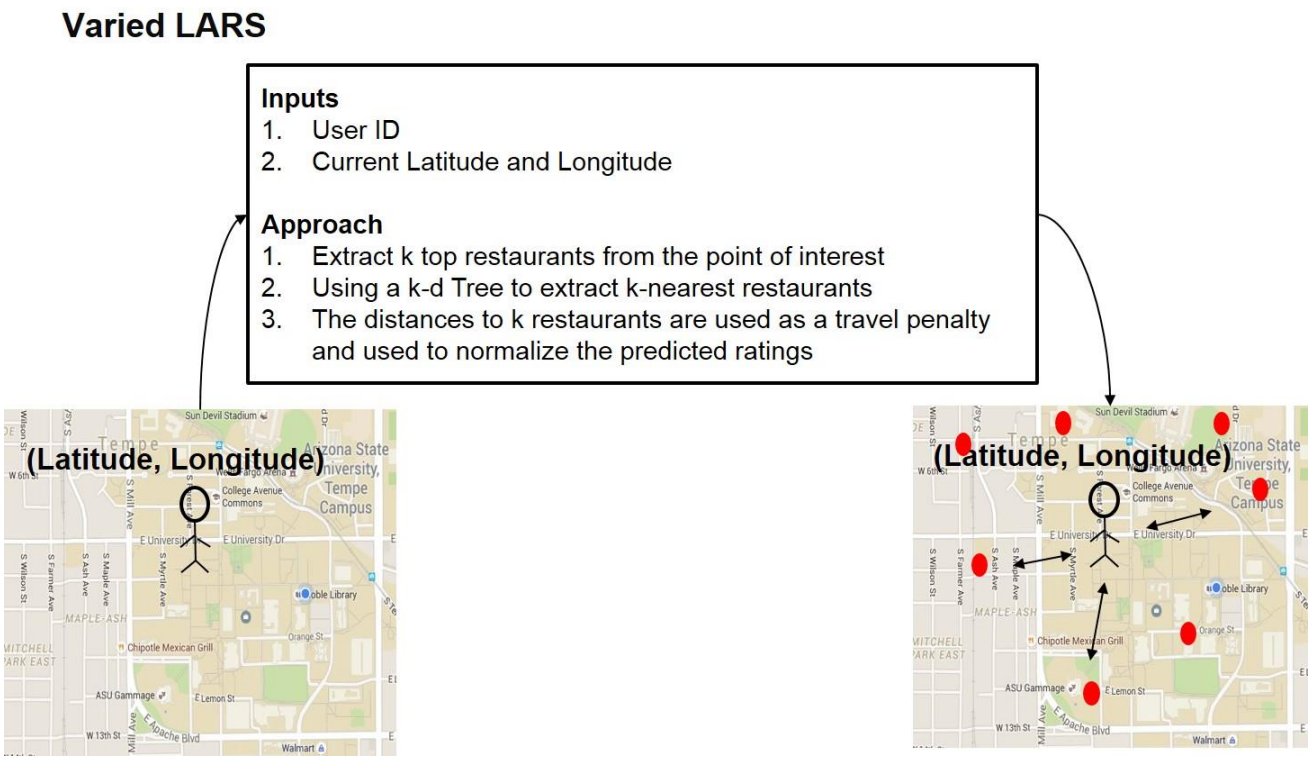

Figure 2-6 Varied Location Aware Recommender System

2.3 Recommender Lifecycle

This section includes a brief description of each of the steps carried out as part of this project. An overview of the system designed using the recommender algorithms described in the previous section is shown in **Figure 2-7**

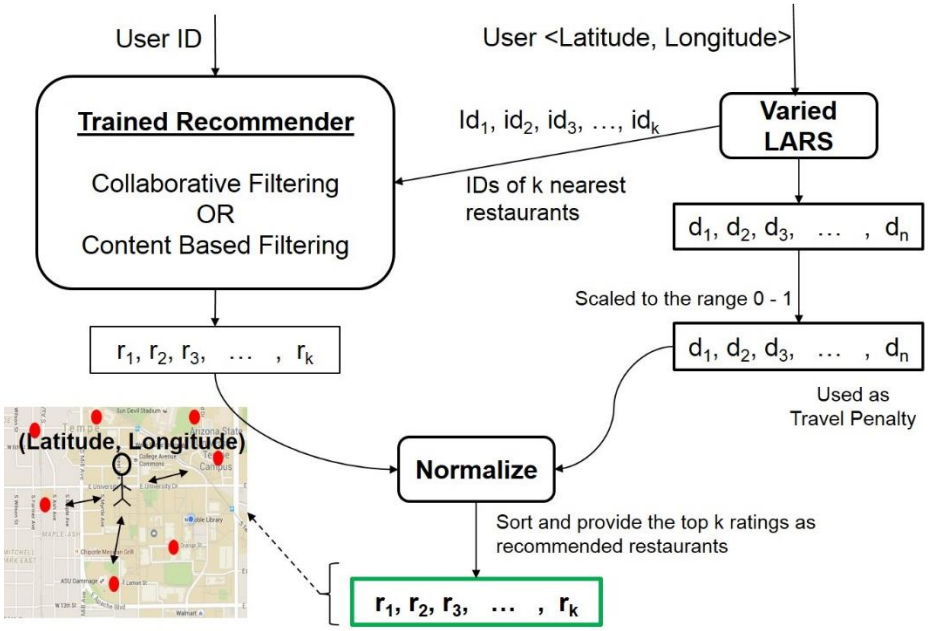

Figure 2-7 Restaurant Recommender Lifecycle

The following is a summary of the steps followed in the complete recommender lifecycle

1. Data Pre-processing
   - The Yelp Dataset was provided in the form of JSONs. In order to process easily and make it readable, they were converted CSVs.
   - The Yelp Dataset provided contained data points all over the globe. In order to make the predictions make more sense, the dataset was reduced to only restaurants in Phoenix.
   - By doing so, the dataset was reduced to about 3,000 restaurants and 65,000 users.
   - The **Figure 2-8** provides a summary of the ratings dataset in the Phoenix area in the form of a histogram and the **Figure 2-9** provides a geographical plot of all the restaurants in the Phoenix area.

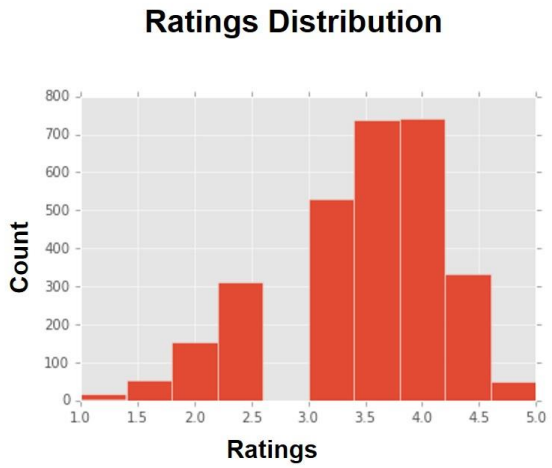

Figure 2-8 Ratings Distribution

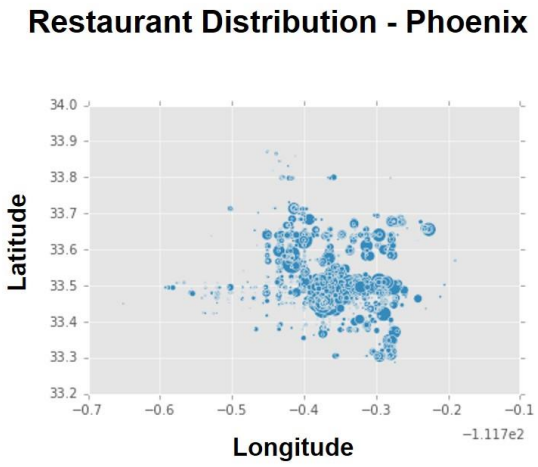

Figure 2-9 Restaurant Geographical Plot

2. Feature Selection
   - The feature selection process was carried out by running a random forest regressor on the data to find the influence of the features on the ratings. The top 10 features were used for the content based recommendation algorithm. The influence of these features as obtained is shown in **Table 2-2**

Feature	Influence
Alcohol	0.0974
Price Range	0.0787
Delivery	0.0744
Parking Lot	0.0686
Outdoor Seating	0.0682
Waiter Service	0.0664
Good For Groups	0.0618
Good For Lunch	0.0564
Takes Reservations	0.0556

Feature	Influence
Parking Street	0.0536

Table 2-2 Feature Selection Influences

3. Putting the recommenders together
   - The available ratings along with the selected features are provided to the recommender algorithm of choice such as Collaborative or Content Based Filtering.
   - The distances of k-restaurants obtained as part of the LARS algorithm are normalized to obtain travel penalties.
   - The travel penalty is subtracted from the predicted ratings obtained using Content based or Collaborative filtering approach.
   - These ratings when sorted in descending order provide the location aware recommendations for a particular user.

3. IMPLEMENTATION

3.1 Details

This section talks about implementation details of each component described in **DESCRIPTION**. A summary of the implementation and their details are available in **Table 3-1**.

Item of interest	Tools/Technologies Used	Description
Data pre-processing	Python, SciKit Learn, Numpy, Pandas	Pre-processing was needed to construct a use-business matrix. This was done by aggregating data from business, review and user jsons using pandas library. **Related Files** Utilities/ matrix_creation.py
Feature Selection	Python, SciKit Learn	Select-From-Model along with Random-Forest-Regressor in SciKit Learn was used to obtain the top 10 features. **Related Files** Feature Selection/feature_selection.py

Item of interest	Tools/Technologies Used	Description
Benchmarking Algorithms	Python, SciKit Learn, Numpy, Pandas	Numpy linear algebra implementation of SVD was used with the *full_matrices property* set to false. Matplotlib was used to plot results. **Related Files** Baseline estimation/Baseline.ipynb Baseline estimation/ SVDBaseline.ipynb
Large Scale Collaborative Filtering	Apache Mahout, Hadoop	In order to carry out this algorithm, Mahout documentation [11] and Recommender Blog [12] were used. The environment setup for running Collaborative Filtering on Mahout was as follows 1. A Hadoop cluster of 3 machines was setup, each running Ubuntu. 2. The user restaurant rating matrix was loaded to HDFS. 3. The data was split into training set (80%) and test set (20%) using the command mahout splitDataset -i hdfs://master:54310/input/user_business_rating_anonymized.csv -o hdfs://master:54310/split -t 0.8 -p 0.2 4. The ALS algorithm was run using the command mahout parallelALS --input hdfs://master:54310/split/trainingSet/ --output hdfs://master:54310/alstraining/ --numFeatures $itr --numIterations 5 --lambda 0.065 5. RMSE on probe set was calculated using mahout evaluateFactorization --input hdfs://master:54310/split/probeSet --output hdfs://master:54310/rmse/ --userFeatures hdfs://master:54310/alstraining/U/ --itemFeatures hdfs://master:54310/alstraining/M/
Content Based Filtering	Python, SciKit Learn, Numpy	The following are the implementation specifications for content based filtering 1. A user restaurant rating matrix was created using a Numpy Array

Item of interest	Tools/Technologies Used	Description
		2. The matrix was split into training and validation matrix. 3. User Parameter matrix was learnt using linear SGD Regressor. 4. Predicted rating matrix was computed using user parameter matrix and restaurant feature matrix. **Related Files** Scripts/Content_based_filtering.py
Varied Location Aware Recommender System	Python, SciKit Learn, Numpy	The aim was to find the nearest k restaurants from a given set of co-ordinates which was done as follows 1. KDTree algorithm with leaf size 2 in SciKit Learn was used to get nearest k restaurants in an efficient manner. **Related Files** Recommender/Restaurant_Recommender.py

Table 3-1 Implementation Details

3.2 Contribution Summary

A brief summary of the contributions from each team member is listed in 2 parts as part of **Table 3-2** and **Table 3-3**.

Common Contributions from all team members
1. Literature Survey 2. Dataset Usage Study

Table 3-2 Common Contributions

Team Member	Contribution	Percentage
Deshu Venkatesh, Vaibhav	Location Aware Recommendation Contributed to other design decisions	16.67%
Gomudurai Pandian, Tharun Niranjan	Content Based Filtering Contributed to other design decisions	16.67%
Iyer, Vignesh	Collaborative Filtering	16.67%

Team Member	Contribution	Percentage
	Contributed to other design decisions	
Manikarnike, Mukund	Creation of Reports and Slides Contributed to all design decisions	16.67%
Singh, Bharat	Content Based Filtering Contributed to other design decisions	16.67%
Syed Mohammad, Yousuf Hussain	Baseline approaches Contributed to other design decisions	16.67%

Table 3-3 Contribution Summaries

4. RESULTS

A sample execution of the recommender system which uses a content based recommender and varied location aware recommender together is as shown in **Figure 4-1**

```
Enter the latitude: Like 33.4 something 33.402772
Enter the longitude: Like -112.0 something -111.976086
Enter User ID:U4AAjQYM8tKs-PzHmfIp5g
1 ->  Pizza Hut -> 4708 S 48th St
Phoenix, AZ 85040

2 ->  Whataburger -> 4610 South 48th Street
Phoenix, AZ 85040

3 ->  Taco Bell -> 4430 South 48th Street
Phoenix, AZ 85040

4 ->  Quiznos -> 4001 E Broadway Rd
Ste B12A
Phoenix, AZ 85040

5 ->  Sonic Drive-In -> 4740 Baseline Rd
Phoenix, AZ 85040
```

Figure 4-1 Sample Execution

4.1 Experimental Results

The performance of each algorithm used for this project in terms of Root mean square error and Mean absolute error is shown in **Table 4-1**.

Algorithm	MAE	RMSE
Benchmarking Algorithms		
Average Ratings	1.042869	1.284065
Offset Average Ratings	0.950065	1.181828
SVD	0.889324	1.138830
Core Algorithms		
Collaborative Filtering	Not Measured	1.349099
Content Based Filtering	1.14450	1.343062
Varied Location Aware Recommender System	No evaluation result available because there is no validation set available with the location attribute being taken into account.	

Table 4-1 Algorithm Performance Measures

4.2 Theoretical Analysis

4.2.1 Evaluation Techniques

In a recommender system, the evaluation of what you might predict for unknown ratings is not an easy problem because of the lack of availability of ground truth. Hence, the evaluation technique used in general in this paper is to split the training set into a training and a validation set and then evaluate the root mean square and mean absolute errors for the ratings that are known already. This gives one a measure of how well the model does. Once this measure is known, one could say that the prediction might on an average deviate by the amount of error obtained.

Table 4-1 shows a list of these errors for each algorithm that was implemented. An error of 1.343062 as obtained for Content based filtering means that the predictive algorithm might on an average deviate from the true value by 1.343062.

4.2.2 Analysis of Results

The RMSE and MAE of Benchmark algorithms are lower than that of the core algorithm because of the following two reasons

1. Most of the features used to perform the prediction are boolean and do not considerably influence the user preference for a particular restaurant.
2. A linear regressor model was used due to lack of availability of a high computing machine to handle the large amount of data.

The RMSE and MAE of Content Based Filtering algorithm is supposed to decrease with a higher-order regressor. There is no Test data to evaluate the performance of LARS with our recommender system. The test runs of the developed recommender shows that restaurants that are near with favorable features are recommended highly to the user.

5. CONCLUSION

The task of a recommender system is to recommend items to a user based on his/her previous history which is typically represented as a profile in any social network. In order to do this, the problem is treated as predicting how a user would rate an item that he or she hasn't rated yet. By predicting how he or she would rate such an item, in effect we would be solving the recommender system problem. Performing

predictions isn't a difficult problem because it most often is trying to fit a curve to a set of data points which is a traditional optimization problem. Solving optimization problems are straight-forward and can be done through any of the gradient descent methods. However, the bigger problem when it comes to recommender system problem is evaluation of these recommendations. The questions one would like to answer when building a recommender system are how good the recommender system is and how one would evaluate the goodness of the recommendations provided. These problems are solved in this project by using benchmarking algorithms and cross validation. Benchmarking algorithms are algorithms of very little complexity. Having such algorithms ensures that we measure the performance for a very simple model to start with and that provides one with a better comparison in terms of models as the model complexity improves. Cross validation helps avoiding the overfitting problem by training the model using different sets of data.

5.1 Key Learnings

Some of the key learnings from this project that we, as a team gained from this project were

- One of the main learnings from this project is that in order to have a good recommender system, it has to train on a geographically focused dataset because usage of ratings of a restaurant in a geographically set apart area would produce extraneous results. Hence, data pre=processing is an important stage of any recommender system project.
- In the area of handling large datasets, we were able to get a clear picture on how to split the ratings dataset into a training and validation set and what percentages work well.
- In the area of recommendation algorithms to be used, we tried gradient descent and stochastic gradient descent approaches for the content based recommender approach. We found that stochastic gradient descent takes a lot less computation time. However, gradient descent performs better in terms of achieving a much accurate solution to the optimization problem. So, the choices of algorithms to be used will have to be appropriately made according to the requirements of the recommender system and the dataset available.

6. ACKNOWLEDGEMENTS

We would like to express our sincere thanks to Dr. Hanghang Tong for his efforts in discussing interesting topics in class that motivated us to do well in this project.

During the course, we also referred to several of Dr. Tom Mitchell's video lectures available on his CMU Website and the lectures of Dr. Andrew NG on Coursera which we found extremely useful and would like thank them for their insightful lectures which were truly thought provoking.

We would also like to thank the team at Yelp that made this project possible by making the dataset available publicly as part of a challenge.

7. FUTURE WORK

The Yelp dataset contains a lot more data than what was considered for this project. The recommender was tested only on restaurants in Phoenix. The items that weren't considered for recommendations were

1. Businesses other than restaurants.
2. Businesses in areas other than the Phoenix area.
3. Photos associated with each business.

4. User reviews in textual form for each business.
5. User tips to other users in textual form about each business.

The future work for this project includes

1. Comparing regions with each other to predict cultures for new regions.
2. Use computer vision APIs to extract a much richer feature set from the photos for each business.
3. Use Natural language processing to extract richer features from the textual data provided.

8. REFERENCES

[1] Sumedh Sawant, Gina Pai, Yelp Food Recommendation System, Stanford University

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

[3] Yelp Engineering Blog – Dataset Challenge
http://engineeringblog.yelp.com/2013/10/yelp-dataset-challenge-winners-round-two-now-live.html

[4] Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, Social Media Mining An Introduction

[5] Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, Mohamed F. Mokbel, LARS: A Location-Aware Recommender System

[6] A recommender systems article at Carleton College, Northfield, MN
http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/svd.html

[7] Kirk Baker, Singular Value Decomposition Tutorial

[8] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, John T. Ried, Application of Dimensionality Reduction in Recommender System -- A Case Study, GroupLens Research Group / Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota

[9] Yehuda Koren, Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model, AT&T Labs – Research

[10] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber and Rong Pan, Large-scale Parallel Collaborative Filtering for the Netflix Prize, HP Labs

[11] Introductions to ALS Recommendations with Hadoop
https://mahout.apache.org/users/recommender/intro-als-hadoop.html

[12] Building Recommender engines with Apache Mahout
https://datasciencehacks.wordpress.com/2013/11/03/building-recommender-engines-with-apache-mahout-part-i/

[13] Numpy Linear Algebra SVD Algorithm
http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.linalg.svd.html

[14] Feature selection documentation from SciKit Learn
http://scikit-learn.org/stable/modules/ensemble.html#forest