Agenda Endorsement System

Ankitha Premkumar
Computer Science
University of Southern California
Los Angeles
apremkum@usc.edu

Prathibha Muralidharan
Computer Science
University of Southern California
Los Angeles
prathibm@usc.edu

Reshma Malla
Computer Science
University of Southern California
Los Angeles
reshmama@usc.edu

Sharath Ravishankar
Computer Science
University of Southern California
Los Angeles
ravishas@usc.edu

ABSTRACT

Planning is a pivotal activity. Say a user has a list of tasks to be completed, there are various ways in which this can be accomplished. Here, a task could range anywhere from going to a restaurant, shopping or getting an errand done. There may be multiple options available to the user but doing it in a timely fashion and in a feasible manner is a challenge and solving that challenge is the goal of the system. The yelp dataset which contains a plethora of businesses is the data source for the engine based on which a plan can be proposed to a user, given his to-do list. A plan involving a list of the appropriate businesses for each of these activities is the output of the system.

1. INTRODUCTION

Volume is one of the 3V's of big data. With a large data set such as yelp's, options to carry out a task are far from limited - be it by location or by variety. Suggesting a location among those choices is a tough activity all-together. However the project utilises already existing state of the art recommendation libraries for prediction of ratings for each of the businesses. The task intensifies when the choices are suggested for a group of tasks. The optimality needs to be considered in entirety for a list of activities. In the market, there are several applications which cater to provide recommendations for a particular task. However in reality, there is more than one task to be done and additional factors need to be considered when done together. This forms the main objective of the agenda endorsement system.

Initially the businesses are grouped according to the activities that can be performed and preprocessed which is fed to the engine. The ratings of the businesses and the round trip distance of the overall plan are the two main factors involved in suggesting the plan for the agenda. The ratings component is predicted from the existing recommendation system library. By default the engine provides the plan with the shortest round trip distance, plan with highest rated businesses for each task and a plan which gives equal preference to both. The project presently focusses on these two factors but can be extrapolated further to consider other features such as user profile, friendship networks and their profiles.

For example, if the tasks that need to be done involve going to a restaurant, shopping and visiting a clinic. There are several businesses that a user can choose from to perform all of these activities. The engine proposed runs the standard recommendation algorithm on each of the businesses and estimates the user's rating for these businesses. After selecting the top businesses ordered by the predicted ratings, all possible combinations are computed with the respective businesses for the activities specified, each of which form a plan. Further, among these list of plans possible, three main plans are provided - 1. Optimal Plan - A plan which gives equal preference to ratings and round trip distance, 2. Shortest distance plan - Plan with shortest round trip distance 3. Best

Rated Plan - Plan with the highest average rating of the list of businesses suggested. The ratings for the businesses are predicted using state of the art mllib library. Moving forward, users can provide a weight which indicates their preference for a best rated plan versus a plan that compromises on rating but is shorter in terms of distance.

2. DATASET

We use the dataset provided by Yelp as part of their Dataset Challenge 2018 (Round 12) for training, prediction models and testing. The dataset includes data from about 188,593 businesses, 5,996,996 reviews, 280,992 pictures, 10 metropolitan areas. The dataset contains 6 files: business.json, review.json, user.json, checkin.json, tip.json, photo.json.

In order to build the recommendation system, we use business.json, review.json and user.json. Businesses have several fields- business_id, business name, latitude, longitude, stars, categories that are relevant to our system. The categories field of a business may contain tags that describe the business, for example, “Thai”, “Desserts”, “Beverages”. In review.json, we use user_id, business_id, stars fields to construct our collaborative filtering model.

3. SETUP

3.1 Pre-Processing

Handling the large dataset in an efficient manner entailed a set of pre-processing steps, which have all been implemented as Python scripts. First, the three data files namely, user.json, business.json and review.json are initially converted to csv format from the original json values using argparse, collections and json libraries. Second, in order to reduce the size of the data to be worked on, we restrict ourselves to the metropolitan with the highest number of businesses (Las Vegas - which has approximately 28K rows)(Depicted in Figure 3.1.1). Each of the businesses is provided with a list of categories that describe the business. These category terms are extracted to get a reduced set of all the unique tags across all businesses (2K category terms). This set is then cast into a set of Umbrella Terms by manual annotation. For example - the category terms Thai, Chinese, Italian are now annotated as Restaurant. Third, these umbrella terms are associated back with the businesses by parsing the category terms provided for each business. As a sub-step, redundant umbrella terms, discrepancies in terminology or spelling errors are eliminated while generating the annotated businesses file. Next, the top four business category umbrella terms are shortlisted along with their subcategories, i.e., the category Restaurant also has categories such as Restaurant-Coffee, Restaurant-Beverages, Restaurant-Pubs etc. Using this subset of umbrella terms, the businesses in Las Vegas are further filtered out (~20k rows). Finally, a list of users who have reviewed at least one of these businesses is extracted. This also reduces the numbers of users that need to be considered for the recommendation system. The fields of interest from the businesses csv file are business_id, business_name, umbrellaTerm_category, latitude and longitude of the business; from the reviews csv are user id, business id and the star rating.

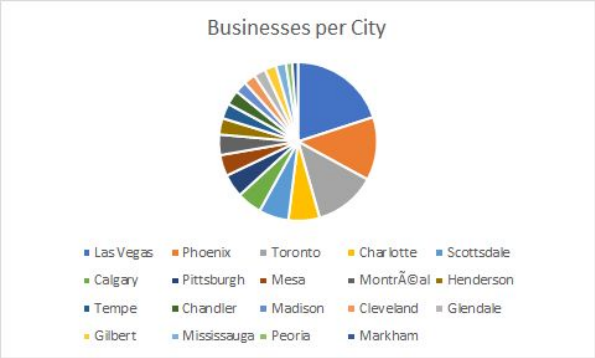


Fig 3.1.1

3.2 Building the Model

We build a collaborative filtering model using the ALS train model of the spark mllib library. The model is trained on the csv file which contains the businesses in Las Vegas, users who have rated them and the corresponding star rating. Before the dataset is fed into the train model, it is preprocessed to conform to the input requirements of ALS train. Since the dataset contains the user ids and business ids as strings, and the train model takes in parameters in int format, both these fields are required to be mapped to unique integers. This map as well as a reverse map (integer ids to strings) are stored separately as part of the model. These maps are used at a later stage while predicting the rating for an unseen user and business pair.

3.3 Endorsement Engine

The Endorsement Engine works on a user's task list and location, and produces three ordered plans of businesses the user should visit to finish their tasks. Each plan prioritizes a different attribute, such as average business rating, round trip distance and an equal weightage plan for business rating and round trip distance.

The user's task list, latitude and longitude (location) are read from a CSV file. Each task in the user list represents an Umbrella Term Category that the user's task belongs to. For every umbrella term in the list, we retrieve the businesses that pertain to it, along with the businesses' location and rating. We filter these businesses to pick out the ones which have been rated by at least one user, and construct user_id, business_id pairs for those businesses. We use our Collaborative Filtering Model to predict ratings for all these user-business pairs and normalize the ratings within the range 0 to 5.

Once this has been done for every task, we perform a cartesian product operation for business lists between tasks. Since the number of combinations of businesses could be a very large number, we restrict this number by only picking the top ten user-business pairs by rating. The number of combinations therefore is restricted to the nth power of ten, if n is the number of tasks specified by the user.

For each n-tuple combination of businesses, we compute the average rating of the plan and the round trip distance from user's location to one business after another, taken in order.

From these combinations, we choose one plan which provides a plan with the best average rating and one plan with the shortest round trip distance. We also compute an equally weighted average of rating and distance to provide an optimal plan. To evaluate our results, we compare these plans with a combination of businesses picked randomly based on the umbrella term. Average rating, distance and a weighted score is computed for the random plan.

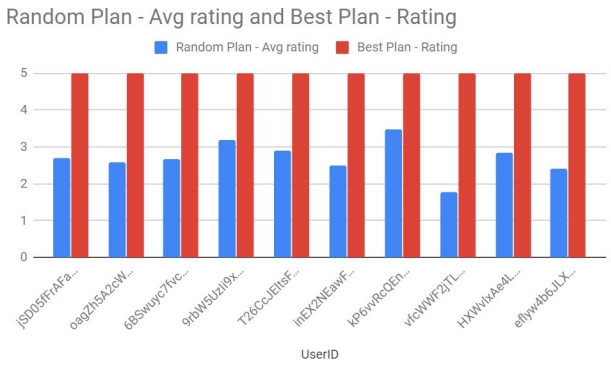
3.4 Challenges

In this section, the challenges encountered while doing the project are described.

1. The categories provided had to be grouped into broader umbrella terms. This involved manual annotation which is subjective to a person and maintaining consistency was a laborious task.
2. An individual business having multiple terms comes under multiple umbrella terms, there by activities, making the recommendation a bit hazy.
3. The main challenge was addressing the negative values in the rating predictions of recommendation system library. This was scaled to the appropriate range and handled.
4. Handling the cold start problem of a new business which has not been rated by any user. The recommendation system usually eliminates the rows which are not previously rated.

4. RESULTS

To test our endorsement system, ten randomly picked users are considered along with the list of tasks to be accomplished in their agenda. In this section the results obtained by the agenda endorsement engine are compared with the random plan for the three parameters, as depicted in the graphs below.



. Fig 4.1

In Fig 4.1, the scores of both the random plan and the best plan are plotted.

In fig 4.2, the round trip distances of 10 randomly picked users are compared with the results of random plan. In case of the last user, the random agent works slightly better than the agent proposed. This is because the plans constructed were combinations of top 10 businesses under every task category. The distance based plan has a bias towards better ratings. Therefore, there are better distance based plans that exist, which we do not explore in our system in order to reduce our search space.

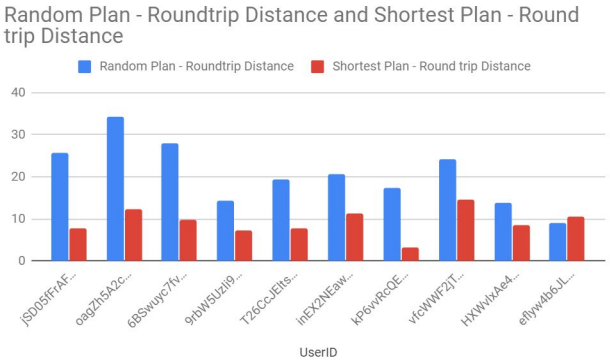


Fig 4.2

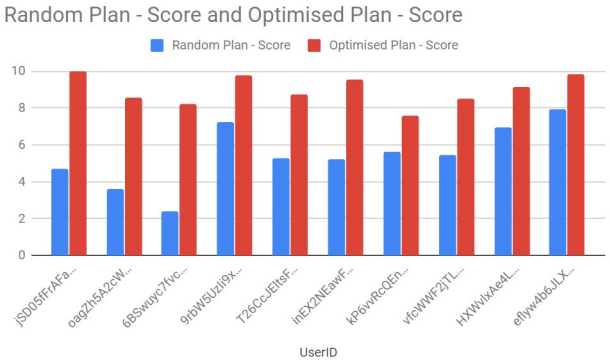


Fig 4.3

On similar lines , the score (weighted average of round trip distance and distance) is compared between the random plan and the optimised plan as shown in fig 4.3.

5. FUTURE WORK

Several factors apart from the ones used by the system could be considered for enhanced recommendations, such as-

1. Give priority to user's friends' rating towards businesses in the collaborative filtering model predictions.
2. Annotations could be more accurate in terms of co-relating the entire business description with an umbrella term rather than individual category terms in the description.
3. Take into account the hours of operation of businesses along with usually busy hours and suggest plans with enhanced optimality.
4. Handle missing values or negative rating predictions by assigning default ratings(overall average of all ratings) to businesses that may not appear as part of the training pairs.
5. The system can be further extended to handle businesses in more than just one metropolitan as well as to provide categories of businesses apart from the top categories considered currently (provides the user a wider variety of activities he can carry out).

6. CONCLUSION

This project involves building an endorsement engine which suggests the plan for performing all the tasks in the agenda. Presently the engine focuses on the businesses in Las Vegas and gives recommendation only in that location. Three plans are provided by default for the list of activities specified in the agenda. Further a provision for giving a weighted bias value towards either roundtrip distance or rating is given. The results of the performance of engine with the random plan are delineated with graphs. They also emphasise on how the model proposed does better than a random selection in terms of our two fundamental parameters (round trip distance and ratings). Down the line, the engine can be made more sophisticated by considering other features for proposing the plan.

7. REFERENCES

[1] "Yelp Dataset Challenge." Yelp. Web. 01 August 2018. https://www.yelp.com/dataset/challenge.

[2] "An Introduction to Yelp Metrics as of September 30 2018." Yelp. 26 May 2016. http://www.yelp.com/factsheet

[3] "Mining of Massive Data Sets" by Anand Rajaraman and Jeffrey Ullman

[4] "Distance between two points given latitude and longitude" https://www.movable-type.co.uk/scripts/latlong.html

[5] "Cartesian product between two lists in scala" https://rosettacode.org/wiki/Cartesian_product_of_two_or_more_lists

8. APPENDIX

Github link to code repository: https://bit.ly/2FSyxyt

Individual Contributions:

1. **Ankitha Premkumar**
   a. Wrote script to convert the JSON files to CSV format using simplejson.
   b. Write script to filter out users from data set.
   c. Wrote scala script to build a utility matrix of users vs business.
   d. Wrote a script in scala that accepts a Map data structure and generates the plans for a particular user.
   e. Script to calculate average rating for each of the proposed plans
2. **Prathibha Muralidharan**
   a. Uploaded data to GCP with BigQuery API to execute queries on the data for simplicity, and further insights.
   b. Write script to pick out businesses with the desired categories only.
   c. Script to pick random businesses from the relevant categories for tasks which the user wants to perform. Random task plan will be used for evaluation against the recommended plan.
   d. Script to calculate distance to be travelled for each plan.
   e. Script for comparison between proposed plan and random plan.
3. **Reshma Malla**
   a. Analyzed data points for each city and selected location with max data points
   b. Script to map umbrella terms back to businesses in Las Vegas.
   c. Wrote a script in Scala to parse input which is in JSON format to retrieve user's tasks. After this, all businesses that match the user's tasks are retrieved. This is repeated for each task.
   d. Wrote a script in scala to compute the possible combinations of plans that would be recommended to each user.
   e. Script to scale negative predictions from (-3.8, +3.8) to (0,5).
4. **Sharath Ravishankar**
   a. Wrote script to extract category field for each data point which is a string provided by the business itself to describe one or more categories it falls under.
   b. Duplicacy in category terms spotted during annotation leading to redundant categories. A script to resolve this issue.
   c. The recommendation system required train-test split on the yelp data and wrote a script to do the same. Included analysis over the RMSE for the various test-train data splits.
   d. Random selection of the plan and predicting it's value from the model and its comparison with our best selected plan.
   e. Visualization for random plan vs Proposed plan based on three parameters- ratings, distance and score.