Designing and Developing a Personalised Recommender System

lrfk99
dept. Computer Science
Durham University
Durham, United Kingdom

I. Introduction

A. Domain of application

The domain of this hybrid recommender system is restaurants, with a focus on trying new restaurants. This system is built around restaurants in a particular metropolitan area which serve a particular type of food.

B. Related work review

Hybrid recommender systems which combine knowledge-based (KB) and collaborative-filtering (CF) techniques are designed to combine the ability of KB techniques to give generic recommendations to new users with small profiles and the strength of CF to find peer users with unexpected shared preferences. This combination of techniques is particularly well suited to restaurants since a user may often be in the mood for a specific type of food, such as Italian or Chinese, so they only want to see suggestions of that type or restaurant, which is where the knowledge-based recommender is helpful. A collaborative-filtering algorithm will then be able to order all the restaurants which match this explicit criteria by how the user should rate them, based on the opinions of other similar users, helping them to discover new restaurants to try. These techniques are used in the Entree restaurant recommender system [1]–[4]. In particular, Burke's 1999 paper [1] is the first to discuss specifically using these two techniques in the same cascaded style that I implement here.

C. Purpose/Aim

The purpose of this application is to give suitable suggestions of restaurants for a user to go to. The recommended restaurants should be in the metropolitan area which the user specifies, and serve the type of food that the user asks for. As well as fitting this explicit criteria, the suggested restaurants should be similar to other restaurants that the user has rated highly in the past and which other users with similar preferences also like. The system will allow the user to add ratings of the restaurants they are recommended, and these ratings will affect future suggestions the system gives for that user and other similar users.

II. Methods

A. Data description

The data for my recommender system was generated from the Yelp Open Dataset [5]. This is a subset of all the businesses, reviews, and user data on Yelp, formatted as JSON files. The dataset includes approximately 8,000,000 reviews of 200,000 businesses by 2,000,000 users, as well as additional Covid-19 related features of each business such as whether they now offer takeout or delivery. There are 3 sections of the dataset, with 5 JSON files in the main dataset, 1 in the photos section, and 1 in the covid section. The $business.json$ file contains business data including location data, attributes, and cuisine. $review.json$ contains full review text data including the $user\_id$ that wrote the review and the $business\_id$ that the review is written for. The file $user.json$ holds every user's first name along with other data adn metadata which I don't use [5]. The data files $checkin.json$ and $tip.json$ are not used in my system. The dataset also includes approximately 200,000 photos but these are not relevant to my application, since the interface is command line only, so they were discarded. The covid related data from $covid\_features.json$ contains additional information about each business with regards to the measures they have taken due to the pandemic, such as offering delivery or takeout options or a special message for customers. Every business in the dataset is in 1 of 10 metropolitan areas in the USA and Canada (Montreal, Calgary, Toronto, Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison, and Cleveland) [5].

B. Data preparation and feature selection

The entire Yelp dataset contains a lot of data which is not relevant to my domain of restaurants, so much data preparation is required to save computation time and in order to provide more accurate predictions. The dataset is provided in JSON format, but in such a way that every entry is its own JSON on its own line, which does not lend well to fast lookups by the main program. Therefore it was necessary to convert every file I used in the system into a typical JSON file, such that it could be loaded as a python dictionary with the lookup key being either the uniquely identifying $business\_id$ or $user\_id$ depending on the case. This allows for extremely fast data lookups in the system. It was of course also necessary to filter the list of all Yelp businesses down to just the restaurants. I further narrowed the list of restaurants by selecting only those which fit into one or more of the top 12 cuisines in terms of number of restaurants (American, Mexican, Italian, Chinese, Seafood, Japanese, Canadian, Mediterranean, Indian, Thai, Middle Eastern, Vietnamese, in that order). Since the user selects one of these 12 options for cuisines they want to find, they will never be recommended a restaurant which doesn't fit at least one of these categories. I chose to use the top 12 cuisines because it's enough to provide a wide variety of options to the user without being so many that the user is overwhelmed by the amount. I also ensured that every restaurant was still open and had at least 1 check-in in 2019 (the last year that data was available). This left 20,565 restaurants which could be recommended to the user, which needed to be split into their respective metropolitan areas by using the latitude and longitude values of the businesses and of the centres of each city. All other datasets were filtered to only data pertaining to these restaurants, with duplicates in the covid dataset removed. The reviews were further filtered to only those which had 5 or more 'useful' votes by other users. This was to ensure I was only using high quality reviews which others found helpful, as well as to allow the recommender system to compute results in a reasonable amount of time (10 seconds on my machine). This left around 120,000 reviews, which were then converted to a CSV file ($reviews.csv$) with unnecessary data removed and structured as follows: ($User\_ID$, $Business\_ID$, $Rating$).

C. Hybrid scheme

The two personalised recommender systems I implement are knowledge-based and collaborative-filtering, and the hybrid scheme is cascade. A cascade hybrid scheme works particularly well in my domain of restaurants with an emphasis on providing accurate predictions of restaurants which a user may like in a city they have never been to before, based on the restaurants they like in their home-city. This is because the knowledge-based system can filter the restaurants by location and by the type of food they serve, while the collaborative-filtering system is able to find restaurants which are similar to the ones the user has previously rated highly. So when these two systems are combined in a cascade style with the knowledge-based system as the primary recommender and the collaborative-filtering system as the secondary the hybrid system can find hidden similarities between restaurants in different cities very accurately. This method of hybrid recommender system (HRS) has been implemented as a restaurant recommender in the Entree system, giving very accurate results [1]–[4].

D. Recommendation techniques/algorithms

The first prediction method used in this hybrid system is a knowledge-based (KB) approach, which relies on knowledge about each of the restaurants based on its location and the type of food that it serves. This approach completely eliminates the cold-start problem, a new user will still receive relevant recommendations for the city they are in and the type of food they in the mood for when using a KB system. The second prediction method is a collaborative-filtering (CF) approach, specifically Single Value Decomposition (SVD) matrix factorisation. This approach was chosen because it scales well with large datasets, which is important since my dataset has 120,000 entries. The CF technique here also introduces an element of variability in the recommendations, such that the user may not receive the same ordering of suggestions when they run the system multiple times with the same KB specifications, helping the user to try new things. This system allows the user to make ratings through the user interface, providing the ability to experiment with the system in terms of how new ratings affect the current user's recommendations as well as those of other similar users.

E. Evaluation methods

The evaluation of the system was done using offline experiments. The accuracy of ratings predictions was evaluated using Mean Absolute Error (MAE), which is calculated by the following equation:

$$MAE = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |\widehat{r}_{ui} - r_{ui}| \tag{1}$$

This metric is appropriate for this RS since it can handle large errors on a few items, which may occur in this system due to some rare skewed ratings. The MAE of this system is around 0.99. Measuring the accuracy of usage predictions is done using precision, calculated by:

$$\frac{TP}{TP + FP} \tag{2}$$

where TP and FP correspond to True Positive and False Positive respectively. This is because the number of recommendations is predefined and not very large (8 items). The precision of this system is 0.125, since 1 out of the 8 items is a true positive. Novelty, diversity, coverage

III. Implementation

A. Input interface

The system has a command line based interface, so all user input of the input is given via the command line. The system recognises the active user by their unique user ID, some example IDs are provided in the 'README.txt' file for testing and demonstration purposes. Only explicit user data is gathered in order to make the system more explainable and transparent, and users are made aware of which data is collected, how it is collected, and for which purposes. This data includes the user's nearest city (of the 10 available) and the type of food that they are in the mood for. The user is able to change these choices from the main menu, as well as switch to a different user for testing purposes. When the user is presented with a list of recommendations, they can then choose one that they want to see more information about and from here they can leave a rating, return to the list, or return to the main menu. User profiles can be updated by giving a rating of a restaurant through the system. Any rating given will affect future recommendations given by the system to that user and to other similar users.

B. Recommendation algorithm

The collaborative-filtering aspect of the algorithm is carried out by first converting the ratings data to a matrix with $User\_ID$ as the rows and $Business\_ID$ as the columns. If a user has not rated a restaurant then the rating in that cell is set to 0. The prediction $\widehat{r}_{ui}$ (user u's rating of item i) is set as:

$$\widehat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{3}$$

Where the parameters are learnt by minimising the following regularised squared error:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \widehat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2) \tag{4}$$

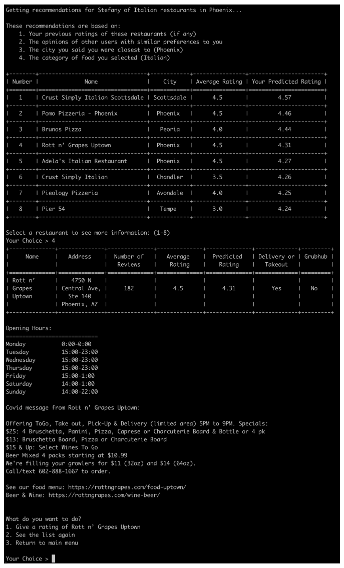
This minimisation is performed by stochastic gradient descent:

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned} \tag{5}$$

where $e_{ui} = r_{ui} - \widehat{r}_{ui}$. These steps are performed over all the ratings of the trainset and repeated 20 times [6]. The knowledge-based aspect of the algorithm produces recommendations based on the explicit location and cuisine preferences of the user. These are then cascaded down to the CF algorithm, which assigns each a predicted rating based on the ratings data. The 8 recommendations with the highest predicted rating are then presented to the user.

C. Output interface

Recommendations and prediction scores are presented to the user via the command line interface and using an ASCII table. I used a python module called texttable for this, since it makes the output very clear and visually appealing. I chose to output the top 8 recommendations since I think this is enough variety and choice for the user to find a restaurant to try, without there being too many options which might overwhelm the user. If I presented more than 8 recommendations at a time, the table wouldn't fit in a single terminal output window and would require the user to scroll around to read different parts of the output. Since the user will be interacting with the system on a laptop or desktop computer through a command line interface, an ASCII table is the most visually attractive way to present information, and the white text on a (typically) black background will help accommodate users who are colourblind or who require a high-contrast interface. The interface is clear enough to understand that users of all ages are able to use the system, and if a user requires the text to be larger then they can increase the text size of their terminal. Along with the recommended restaurants, an explanation of how these results are calculated is also provided, as well as the prediction values themselves so the user can understand why they are being shown these suggestions. This, along with an option in the main menu to view information about the data the system uses and how it is used, further adds to the explainability and transparency.



```
Getting recommendations for Stefany of Italian restaurants in Phoenix...

These recommendations are based on:
    1. Your previous ratings of these restaurants (if any)
    2. The opinions of other users with similar preferences to you
    3. The city you said you were closest to (Phoenix)
    4. The category of food you selected (Italian)

+--------+---------------------------------+------------+----------------+-----------------------+
| Number |              Name               |    City    | Average Rating | Your Predicted Rating |
+========+=================================+============+================+=======================+
|   1    | Crust Simply Italian Scottsdale | Scottsdale |      4.5       |         4.57          |
+--------+---------------------------------+------------+----------------+-----------------------+
|   2    | Pomo Pizzeria - Phoenix         |  Phoenix   |      4.5       |         4.46          |
+--------+---------------------------------+------------+----------------+-----------------------+
|   3    | Brunos Pizza                    |   Peoria   |      4.0       |         4.44          |
+--------+---------------------------------+------------+----------------+-----------------------+
|   4    | Rott n' Grapes Uptown           |  Phoenix   |      4.5       |         4.31          |
+--------+---------------------------------+------------+----------------+-----------------------+
|   5    | Adela's Italian Restaurant      |  Phoenix   |      4.5       |         4.27          |
+--------+---------------------------------+------------+----------------+-----------------------+
|   6    | Crust Simply Italian            |  Chandler  |      3.5       |         4.26          |
+--------+---------------------------------+------------+----------------+-----------------------+
|   7    | Pieology Pizzeria               |  Avondale  |      4.0       |         4.25          |
+--------+---------------------------------+------------+----------------+-----------------------+
|   8    | Pier 54                         |   Tempe    |      3.0       |         4.24          |
+--------+---------------------------------+------------+----------------+-----------------------+

Select a restaurant to see more information: (1-8)
Your Choice > 4
+--------------+--------------+--------------+--------------+--------------+-------------+---------+
|     Name     |   Address    |  Number of   |   Average    |  Predicted   | Delivery or | Grubhub |
|              |              |   Reviews    |    Rating    |    Rating    |   Takeout   |         |
+==============+==============+==============+==============+==============+=============+=========+
| Rott n'      |    4750 N    |              |              |              |             |         |
| Grapes       | Central Ave, |     182      |     4.5      |     4.31     |     Yes     |   No    |
| Uptown       |   Ste 140    |              |              |              |             |         |
|              | Phoenix, AZ  |              |              |              |             |         |
+--------------+--------------+--------------+--------------+--------------+-------------+---------+

Opening Hours:
============================
Monday          0:00-0:00
Tuesday         15:00-23:00
Wednesday       15:00-23:00
Thursday        15:00-23:00
Friday          15:00-1:00
Saturday        14:00-1:00
Sunday          14:00-22:00

Covid message from Rott n' Grapes Uptown:

Offering ToGo, Take out, Pick-Up & Delivery (limited area) 5PM to 9PM. Specials:
$25: 4 Bruschetta, Panini, Pizza, Caprese or Charcuterie Board & Bottle or 4 pk
$13: Bruschetta Board, Pizza or Charcuterie Board
$15 & Up: Select Wines To Go
Beer Mixed 4 packs starting at $10.99
We're filling your growlers for $11 (32oz) and $14 (64oz).
Call/text 602-888-1667 to order.

See our food menu: https://rottngrapes.com/food-uptown/
Beer & Wine: https://rottngrapes.com/wine-beer/


What do you want to do?
1. Give a rating of Rott n' Grapes Uptown
2. See the list again
3. Return to main menu

Your Choice > █
```

Fig. 1. System output

IV. Evaluation results

A. Comparison against baseline implementation

Cross validation with 5 folds was carried out on both the SVD system and a baseline implementation. The 5 Mean Absolute Errors for the SVD system had a mean of 0.9962 and a standard deviation of 0.0033. The MAE results for the baseline implementation had a mean of 1.0091 and a standard deviation of 0.0056. This shows that when compared to a baseline implementation, the SVD system does perform better, generating more accurate predictions, however the difference is not particularly large.

B. Comparison against hybrid recommenders in related studies

The Entree restaurant recommender system is an example of a cascade style hybrid recommender system which uses both knowledge-based and collaborative-filtering methods to help users find a restaurant they might like in a city they've never been to before [1]–[4]. This is very similar to the way my system works, with the biggest difference being the vastly greater amount of knowledge engineering that has gone into Entree. This additional knowledge is used in Entree, and other FindMe style systems, to give more accurate similarity assessments of restaurants by considering factors such as price and quality of experience as well as location and cuisine, and by arranging these factors hierarchically. For example, Entree arranges these as: cuisine, price, quality of experience, others, but other systems allow the user to specify this order themselves. This is part of the reason why Entree would outperform this system.

C. Ethical issues

People's personal data

V. Conclusion

A. Limitations

What can it not do?

B. Further developments

What could I do in the future [7]. Implement the idea of particular cuisines being closer some than others. Specify price range and/or consider price when comparing similarities of restaurants. Include demographic information

References

[1] R. Burke, "Integrating knowledge-based and collaborative-filtering recommender systems," in *Proceedings of the Workshop on AI and Electronic Commerce*, 1999, pp. 69–72.

[2] ——, "Knowledge-based recommender systems," *Encyclopedia of library and information systems*, vol. 69, no. Supplement 32, pp. 175–186, 2000.

[3] ——, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[4] ——, "Hybrid web recommender systems," *The adaptive web*, pp. 377–408, 2007.

[5] Yelp. Yelp open dataset. Website. [Online]. Available: https://www.yelp.com/dataset

[6] N. Hug. Matrix factorization-based algorithms. Website. [Online]. Available: https://surprise.readthedocs.io

[7] L. Martinez, R. M. Rodriguez, and M. Espinilla, "Reja: A georeferenced hybrid recommender system for restaurants," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3, 2009, pp. 187–190.