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 categories. $review.json$ 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 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 categories 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 categories of restaurants they want to find, they will never be recommended a restaurant

which doesn't fit at least one of these categories. 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. The list of reviews pertaining to these restaurants was then filtered down I also split by nearest city, and filtered only useful reviews This was done using the $categories$ attribute of the $business$ JSON file.

The business data was filtered down to businesses that contain 'Restaurants' as one of their listed categories getIDs filterReviews lastCheckin recentCheckin toCSV CovidDuplicates getUserIDs mostActiveUsers numberRestaurantReviews mostReviews filterMostActiveReviews uniqueRestaurants dropText splitCities splitIDs filterUsefulReviews filterRecentReviews splitReviews getCategories formatCovidFeatures closedCovid

The ratings data was (20,565 restaurants) Prepare data Of the 209,393 businesses in the dataset, 168,903 are still open. There are 43,965 restaurants which are open and contain 'Restaurants' as one of the categories. I then picked the top 12 most popular categories of restaurant: American, Mexican, Italian, Chinese, Seafood, Japanese, Canadian, Mediterranean, Indian, Thai, Middle Eastern, Vietnamese (in that order). When filtered for restaurants which fall into at least one of these categories, 20,565 had at least one check-in in 2019, but 333 of these were temporarily closed due to covid but should be open now, and 1 is still closed until June 2021. 19,862 restaurants had a review written about them in 2019, there were 503,979 reviews written in 2019 about these restaurants. There are 2,956,316 reviews of these restaurants. 481,073 tips for these restaurants. 1,968,703 users. None of these restaurants are temporarily closed according to the covid features data. 20,573 restaurants listed with covid features, 4 restaurants which make up 8 duplicate entries (4 have 3 entries), the only difference between their entries being in the "Virtual Services Offered" section. Allow to pick user ID of 10 users with most reviews made, to avoid cold start problem 4 users have over 10,000 reviews, the most is 14,455 (Victor, 8k3aO-mPeyhbR5HUucA5aA). I actually want to select the most reviews of the places I am accepting, restaurants. When taking the most reviews of restaurants in teh dataset, a threshold of 600 reviews gives a list of 10 users, with the most being 1747. I then checked each of these for how many different restaurants they had reviewed and the greatest number of times they had reviewed a single restaurant to check that none of them were just users who reviewed a small number of restaurants a huge amount of times each. All 10 users passed this criteria, with even the least active user having reviewed 496 unique restaurants, which will certainly avoid any cold-start problems. I split the businesses into 10 groups, by the city which they are closest to, I did this by using latitude and longitude rather than State or City since this does not give actual distance and restaurants close to a border could then be mis-categorised. Since there are 4,223,201 reviews of restaurants I filtered them by removing any which had zero 'useful' votes. When filtered to reviews with at least 1 'useful' vote, there are 1,735,732 and when at least 2 votes, 836,929 At least 3 votes, 462,412, at least 5 votes 190,902. 51,737 with at least 10 votes

Top categories: American, Mexican, Italian, Chinese, Seafood, Japanese, Canadian, Mediterranean, Indian, Thai, Middle Eastern, Vietnamese

The entire Yelp dataset is huge and much of it is not necessary for my domain of bars. As such I prepared the data by eliminating any data not relevant to my domain. I selected features such as user ratings and a particular user's average rating, since if a user typically rates places they go highly, but rates a particular bar low, this is more significant than a user who always rates places low.

C. Hybrid scheme

Which two algorithms A hybrid scheme is a good way to design a recommender system, since you can get the best of both algorithms if done properly. I am using a Cascade Hybrid Recommender System, since this has been shown to accurate predictions in my domain of restaurant recommendations [4]. The two systems are Collaborative Filtering and Content Based.

Meaning better recommendations than either algorithm could achieve individually. Cascade with knowledge-based and collaborative

D. Recommendation techniques/algorithms

The first recommender system is collaborative filtering. reference https://surpriselib.com/ I am using item-based collaborative filtering, since the number of users is larger than the number of items (1,968,703 users vs 209,393 businesses in the entire dataset), this gives greater accuracy of predictions. In this case the list of available items does not change, so a user-based method would be unnecessary. One of the requirements for this system is justifiability, and item-based methods make it much easier to explain why certain predictions were made. Item-item collaborative filtering looks for restaurants that are similar, in terms of how people rate them, to the restaurants that the user has already rated and recommend the most similar restaurants. Filtering reviews by city and then using CF won't work because the active user may not have ever reviewed a restaurant in that city, so there is no way to assess similar restaurants to ones they have already rated. Instead I carried out CF on all high quality reviews and produce an ordering of recommendations, which I then filter by city. Mention Netflix Prize and proposition of SVD, use lecture slides explanation and surprise documentation.

I am using a weighted mixed combination of these two systems in order to produce the results of my hybrid recommender system.

E. Evaluation methods

How to evaluate

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 idea is to find restaurants in a new city which are similar to those in another city that the user likes, just like Entree. I used a model-based approach to Collaborative Filtering based on matrix factorisation. The specific model is Single Value Decomposition (SVD), which was popularised by Simon Funk during the Netflix Prize. In my program, I fit the algorithm on a training set, and then for every restaurant of the specified category and in the specified city, I predict what the user would rate that restaurant. Then I sort those restaurants by their score and take the top 8 to present 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.

IV. Evaluation results

A. Comparison against baseline implementation

Compare vs generic suggestions

B. Comparison against hybrid recommenders in related studies

Read some papers

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 [6]. Implement the idea of particular categories being closer some than others. Specify price range and/or consider price when comparing similarities of restaurants.

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] 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.